十六式いろは煌（きらめき）

第 33 回世界コンピュータ将棋選手権
アピール文

メンバー

末吉竜介、宇井絢音、市川翠、加藤凜、 Phan Xuan Hoa 、
有賀宏樹、 Panwar Abhishek 、大熊琉斗、茂木海輝

「十六式いろは 煌」の由来

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

以下、 2022 年当時の由来の記載です。

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

ソフトの概要

採用予定

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

ソフトの説明（予定）

- dlshogi のモデルを軽量なものに変更
- やねうら王での評価関数の学習
- 定跡ファイルの作成
- floodgate 、 AobaZero の棋譜の利用
- 詰将棋エンジンを含めて合議制の見直し

2023.03.29

第33回
世界コンピュータ将棋選手権
なのはアピール文書

2023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 東京都在住
- とあるメーカーに勤務(ちょっとAWS使う)
- 好きな食べ物は焼肉、しゃぶしゃぶ、寿司
- 好きなアニメは魔法少女リリカルなのは、りゅうおうのおしごと！、痛いのは嫌なので防御力に極振りしたいと思います。、くまクマ熊ベアー、とある科学の超電磁砲、ラブライブ！、五等分の花嫁
- 将棋ウォーズ 1級
- 囲碁は日本棋院 初段(アマチュア)

■開発環境

- こんなPCで開発しています

- 出場PC

  Lenovo Thinkpad T14

  CPU：AMD Ryzen 7 PRO 4750U

  メモリ：16GB

  OS：Windows11

  プレゼントでいただきました

■なのはの構成

- MSYS2のg++で開発
- Visual Studio Community 2022(C++)にてビルド
- 手生成では歩、角、飛の不成も生成
- 探索はStockfish使用
- Bitboard未使用(盤情報は配列)
- 定跡部は実戦での出現数および勝率を考慮して手を選択
- 評価ベクトルは直前に学習
  - 周回遅れの3駒関係による評価 → 間に合えばNNUE型を使うかも？

…と、数年前に見たような平凡な構成

■なのはの特徴は？

- 特に主張するような特徴はありません(今回、詰めルーチン未搭載)

  →詰めルーチンを間に合わせたい…

■意気込み

- 0次予選突破！
- 出来れば勝ち越したい
- Wi-Fiなので回線が切れないことを祈っています

■使用ライブラリについて

- なのはmini　Stockfishをベースに独自に将棋化しました。

■使用データについて

- 評価ベクトルにはやねうらおさんが一般に流布した110億教師データを使う予定

■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

アピール文

2023.3.4(作成)

花井　祐

プログラムの名称：いちびん

プログラムの特徴：

１．概要

NNUE を基本としています。自作の教師データを作成し、やねうら王のラーニング機能を使って思考部分を作成しています。探索は、やねうら王の探索ルーチンを、自作のプログラム（JAVA）を用いて、動作をコントロールしています。

１-１.教師データ

数年前に一世を風靡した３駒関係に基づく評価関数（当時、最強だったDorphin?）をもとにして、当時のやねうら王（確か、Ver.4　or 5）の教師データの生成を行っています。その際に、ソース・コードを改変して、１手あたり最大で１００万局面、探索枝狩りをゆるくして教師データを作成しています。

1－２.評価関数

ここ数年間で、私は、1.1で述べた教師データ（深さ１２～１６）を約１５億局面ほど作成しています。今回は、その集大成として、全データを用いて、完全なヴァージンから評価関数を作成しました。ターゲットとした評価関数はNNUEで、その学習は、やねうら王のラーニング機能を使って作成しています。

1－３.探索評価

やねうら王の探索ルーチンもとにして、それを自作のプログラム（JAVA）を用いて、動作をコントロールしています。JAVAでコントロールする項目は、通信内容、時間管理、評価値の異常値検査（突然、評価が楽観読みに変わったときに、思考時間操作）、勝負手探索（対人間しか役立たない。評価値が一定以下のマイナス評価の際に使用）、みっともない負け方をしない機能（相入王が期待できない局面になったら、さっさと投了信号を送る）などです。

２．最後に（作者からのコメント）

毎年、アイデアのネタ切れから、今年を最後にしようと思うのですが、１２月末にコンピュータ将棋事務局の方から、お誘いのメールをもらうと、ついつい参加申し込みをしていまいます。ここ何年か、コロナの影響で、皆が集合しての大会が、無しか、あっても小規模だったので、皆さん、会場に是非とも行きましょう。ワイワイガヤガヤ、会場の雰囲気はとても楽しいですよ。今年は、最終日に懇親会もあるそうです。（なんでもアルコール飲料は無いとのこと、残念。）

（以上）

第 33 回世界コンピュータ将棋選手権 参加ソフト

ねね将棋 アピール文書

日高雅俊

2023/04/16

方針

iPad 上で深層学習ベースの将棋 AI と NNUE ベースの将棋 AI の合議を行い、iPad の能力をできるだけ引き出して高い棋力を目指す。

使用予定ライブラリ:

- やねうら王（NNUE 評価関数により主に CPU で思考）
  - https://github.com/yaneurao/YaneuraOu
  - 水匠 5 評価関数を使用 https://twitter.com/tayayan_ts/status/1478647171419611142
- ふかうら王（やねうら王の一種。深層学習系評価関数により主に機械学習専用チップ Neural Engine で思考）
  - dlshogi 系評価関数を使用（書籍「強い将棋ソフトの創りかた」サンプルコードにより学習した 20 層 192 チャンネルの CNN）
- cshogi（Mac 上での合議に利用）
  - https://github.com/TadaoYamaoka/cshogi

合議

使用ハードウェアである iPad(第 9 世代)では、NNUE のほうが DL（深層学習）より強い（勝率 80%程度）ため、NNUE の指し手を主体として、DL は補助として使用することとした。

評価値はエンジンごとにスケールが異なるため、合議の前に、両者の評価値を勝率に変換する。変換関数は、自己対戦により評価値と勝敗のペアを収集し、シグモイド関数のフィッティングにより求めた。

NNUE では MultiPV により候補手を 2 手出力し、DL では候補手を 5 手出力する。NNUE では複数の候補手を出すことは若干の棋力低下につながる一方、DL では影響がない。NNUE の候補手が DL 側にも存在した場合、それらの勝率を係数 0.75, 0.25 で重みづけ平均したものをその候補手の勝率とみなす。この処理を行ったうえで NNUE の候補手のうち最も勝率が高いものを選択する。すなわち、NNUE の候補手間の勝率差がわずかな場合、DL の出力により順序が逆転する場合がある。

時間管理は特に行わず、両者の指し手がそろった時点で合議処理を行う。Ponder は未実装。

ただし、64 手目以降は、詰み付近の処理に強いと考えられる NNUE のみで、MultiPV を用いずに指し手を決定する。

実装

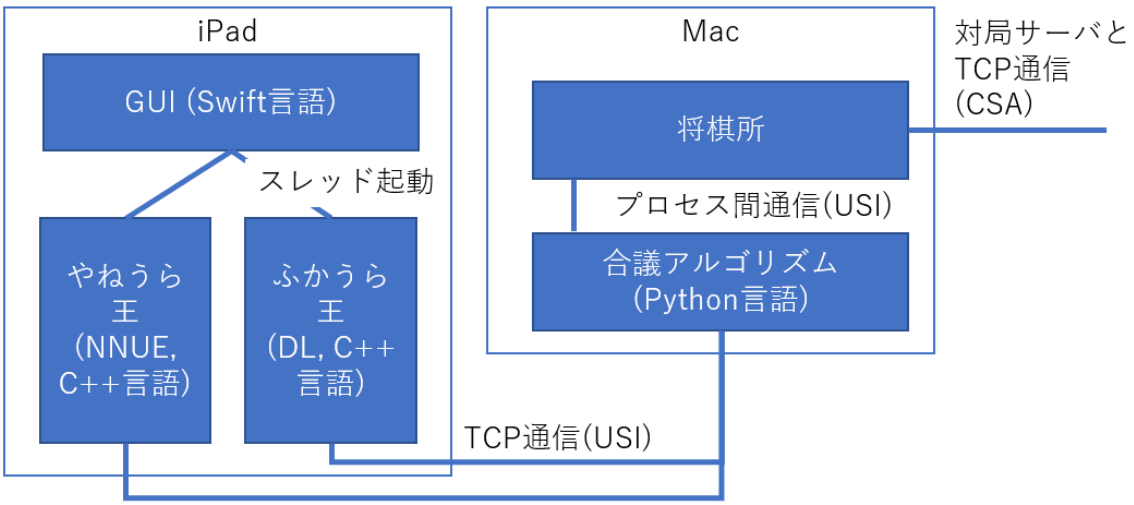
アプリ内で複数のプロセスを立てられない iPad 環境において、2 つのエンジンを同時に動作させる必要がある。そのため、特殊なビルドを実施した。

- 同じ名前の関数を、NNUE/DL でコンパイルオプションを変えてビルドし、リンクすることとなる。通常はリンカエラーとなるため、すべての関数を名前空間内に格納し、名前空間の名前自体をプリプロセッサマクロで置換した。
- 同一の標準入出力に NNUE/DL 双方の出力が混合されてしまう問題がある。出力を分離するため、標準入出力を呼び出したスレッドが属するエンジンを特定し、異なるソケットに対して入出力するようにした。

詳細はブログに掲載する。

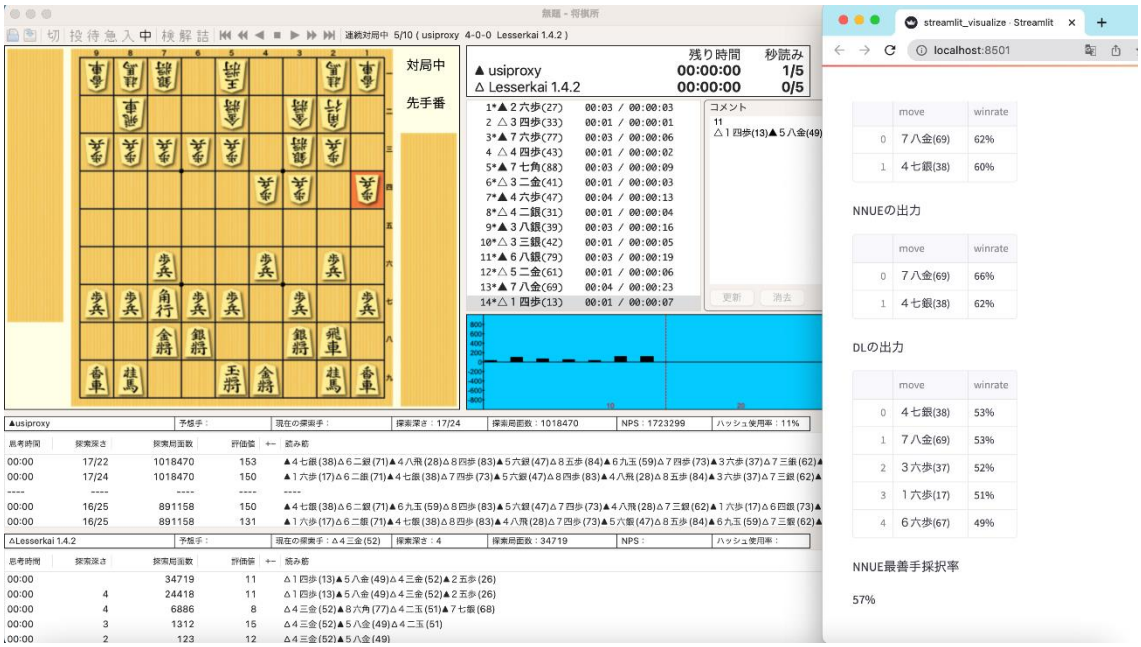
https://select766.hatenablog.com/archive/category/%E3%82%B3%E3%83%B3%E3%83%94%E3%83%A5%E3%83%BC%E3%82%BF%E5%B0%86%E6%A3%8B

合議および CSA プロトコルでの通信の実装は、Mac 上で行う。計算コストはわずかであり理論上は iPad 上で実現可能であるが、開発期間の都合により Mac で動作させる。



システム構成図。

合議の内容は Mac 上で可視化する。



局面と合議内容を表示した Mac の画面。

手抜きについて

手抜きチーム *

2023 年 3 月 28 日

本稿は第 33 回世界コンピュータ将棋選手権における手抜きのアピール文書です。

1　手抜きについて

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

1.1　使用ライブラリ

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

1.2　特長

- NNUE
- α β 探索
- D 言語

1.3　Q&A

Q1. なぜ D 言語なのですか

A1. 1. 関数プロトタイプがいらない 2. 配列をスタック領域に置ける 3. ガベージコレクションがある 4. メモリ安全である 5. 速い からです

1. 関数プロトタイプがいらない

C/C++ では使っている関数のシグネチャをコンパイラに教えるために、プログラマが関数プロトタイプを書かなければなりません。これは DRY ではありません。関数のシグネチャは関数本体

* 鈴木太朗 @hikaen2, 玉川直樹 @Neakih_kick

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

に書いてあるのですから、本来はコンパイラが関数本体を見に行けばよいはずです。D（や Rust やその他の多くの言語）ではそのようになっているため、プログラマが関数プロトタイプを書く必要がありません。

2. 配列をスタック領域に置ける

Java や C#などの高級な言語では配列やオブジェクトはヒープ領域に置かれます。そのため、たとえば探索で配列やオブジェクトを作るとそのたびにヒープアロケーションが発生して時間がかかります。配列をスタック領域に置くのであれば、スタックポインタを進めるだけなのでとくに時間がかかりません。D では配列やオブジェクトをスタック領域に置くことができます。

3. ガベージコレクションがある

普通のプログラム（システムプログラムでないプログラム）を書くのであればガベージコレクションがあったほうが便利だと思います。C や C++ や Rust はシステムプログラミング言語なのでガベージコレクションがありません。D はガベージコレクションがあります。

4. メモリ安全である

C/C++ は配列の境界チェックをしないためメモリ安全ではありません。C/C++ で配列の境界を超えたアクセスは未定義動作であり、わかりにくく再現性のないバグを引き起こします。D は配列の境界チェックをチェックをするのでメモリ安全です。

5. 速い

ldc2[*2]でコンパイルした D のコードは、clang でコンパイルした C のコードと同等の速さが期待できると思います。

Q2. Rust でもよいのではありませんか

A2. Rust でもよいと思いますが、普通のプログラムを書くのであればガベージコレクションがあったほうが便利だと思います。

2　付録 1：私のための Minimax 入門

本章では私にむけて Minimax の解説をします。

関連する前提知識：再帰関数、木構造、ゲーム木、深さ優先探索、Ruby など

また、次の内容は本解説の対象外です：合法手生成、静止探索、静的評価、反復深化、置換表 など

2.1　Minimax

2.1.1　やりたいこと

図 1 のゲーム木で局面 a の評価値を求めたい。どうすればよいか。ただし先手番の局面が△、後手番の局面が▽とする。△、▽の中に先手から見た評価値を記入するものとする。

[*2] `https://github.com/ldc-developers/ldc`

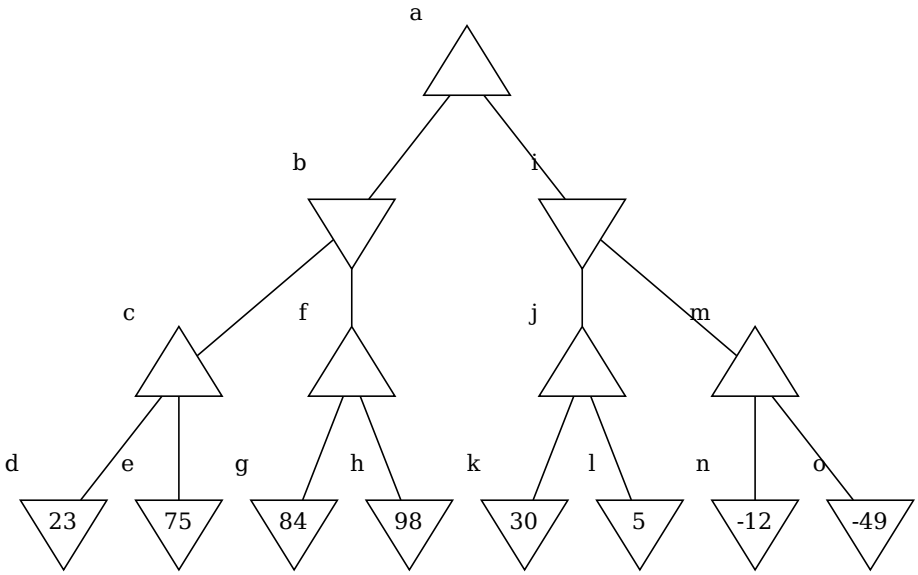


図 1

2.1.2 手順

まず c と f の評価値を求める。c と f は先手番なので子局面の中からなるべく評価値の大きいものを選ぶ。すなわち：

$c = \max(d, e) = \max(23, 75) = 75$

$f = \max(g, h) = \max(84, 98) = 98$

つぎに b の評価値を求める。b は後手番なので子局面の中からなるべく評価値の小さいものを選ぶ。すなわち：

$b = \min(c, f) = \min(75, 98) = 75$

同じように j と m と i の評価値を求める：

$j = \max(k, l) = \max(30, 5) = 30$

$m = \max(n, o) = \max(-12, -49) = -12$

$i = \min(j, m) = \min(30, -12) = -12$

最後に a の評価値を求める：

$a = \max(b, i) = \max(75, -12) = 75$

以上の結果を記入すると図 2 になる。

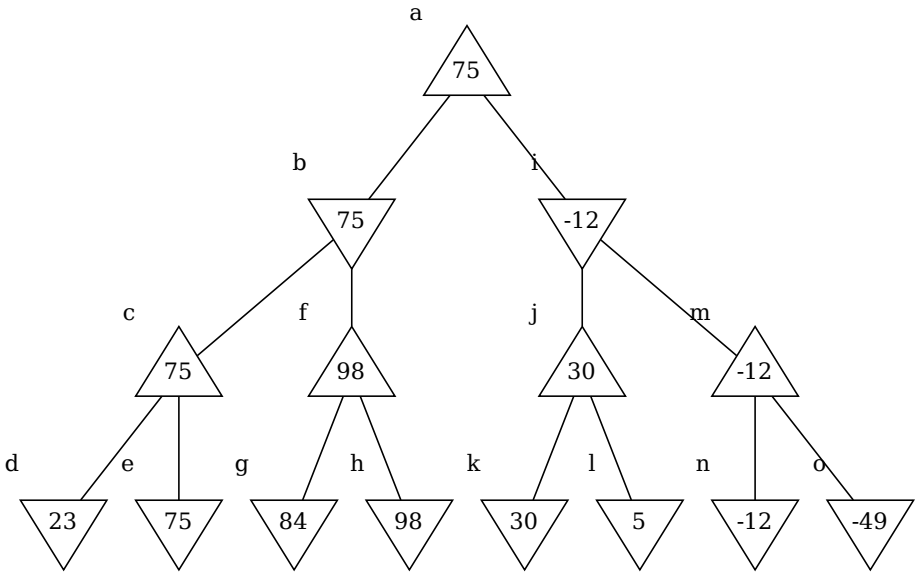


図 2

また、以上の結果を一つの式にまとめると：

$$
\begin{aligned}
a &= \max\Big(\min\big(\max(d,e),\max(g,h)\big),\min\big(\max(k,l),\max(n,o)\big)\Big) \\
&= \max\Big(\min\big(\max(23,75),\max(84,98)\big),\min\big(\max(30,5),\max(-12,-49)\big)\Big) \\
&= \max\Big(\min\big(75,98\big),\min\big(30,-12\big)\Big) \\
&= \max\Big(75,-12\Big) \\
&= 75
\end{aligned}
\tag{1}
$$

2.1.3 コード (Ruby)

リスト 1: minimax.rb

```
tree =
  {side: :MAX, children: [
    {side: :MIN, children: [
      {side: :MAX, children: [
        {side: :MIN, value: 23},
        {side: :MIN, value: 75},
      ]},
      {side: :MAX, children: [
        {side: :MIN, value: 84},
        {side: :MIN, value: 98},
      ]},
    ]},
    {side: :MIN, children: [
```

```
14          {side: :MAX, children: [
15            {side: :MIN, value: 30},
16            {side: :MIN, value: 5},
17          ]},
18          {side: :MAX, children: [
19            {side: :MIN, value: -12},
20            {side: :MIN, value: -49},
21          ]},
22        ]},
23      ]}
24
25  def minimax(p)
26    return p[:value] if p[:children] == nil
27    return p[:children].map{|q| minimax(q)}.max if p[:side] == :MAX
28    return p[:children].map{|q| minimax(q)}.min if p[:side] == :MIN
29  end
30
31  puts minimax(tree) # => 75
```

1 行目で定義している tree が図 1 のゲーム木である。

25 行目で minimax 関数を定義している。p は Position（局面）の頭文字である。q は p の子局面である。

26 行目は終端局面のときのコードである。（静的）評価値を返している。

27 行目は先手の場合のコードである。子局面の minimax 値のうち最も大きいものを返している。

28 行目は後手の場合のコードである。子局面の minimax 値のうち最も小さいものを返している。

2.1.4 Q&A

Q1. なぜ Ruby なのですか

A1. 簡潔で表現力が高いので説明に適していると考えたからです。疑似コードと違い実際に実行できるという利点もあります。一方、C や D や Rust に比べて実行速度は遅いので、実戦には向きません。

Q2. 式 (1) からどのようにしたらリスト 1 が得られるのですか

A2. 式 (1) は min と max が再帰的に連なっています。そこでどうにかして式 (1) を次の漸化式にします：

$$\mathrm{minimax}(p) = \begin{cases} \text{value of } p & (p\text{が終端局面のとき}) \\ \max\limits_{q \in \text{children of } p} \mathrm{minimax}(q) & (\text{上記以外で}p\text{が先手番のとき}) \\ \min\limits_{q \in \text{children of } p} \mathrm{minimax}(q) & (\text{上記以外で}p\text{が後手番のとき}) \end{cases} \tag{2}$$

これを Ruby で書くとリスト 1 が得られます。

2.2 Negamax

リスト 1 では先手か後手かで場合分けしている（27 行目と 28 行目）。Negamax ではこれを先後の区別なく共通化する。子局面の評価値の最小を求めることは、子局面の評価値の正負を反転して最大を求め、結果の正負を反転することと同じである。すなわち $\min(x, y) = -\max(-x, -y)$。これを用いて式 (1) を変形すると：

$$
\begin{aligned}
a &= \max\Big(\min\big(\max(d,e),\max(g,h)\big),\min\big(\max(k,l),\max(n,o)\big)\Big) \\
&= \max\Big(-\max\big(-\max(d,e),-\max(g,h)\big),-\max\big(-\max(k,l),-\max(n,o)\big)\Big) \\
&= \max\Big(-\max\big(-\max(23,75),-\max(84,98)\big),-\max\big(-\max(30,5),-\max(-12,-49)\big)\Big) \\
&= \max\Big(-\max\big(-75,-98\big),-\max\big(-30,-(-12)\big)\Big) \\
&= \max\Big(-(-75),-12\Big) \\
&= 75
\end{aligned}
\tag{3}
$$

2.2.1 コード (Ruby)

tree の定義はリスト 1 と同じなので省略する。以下同様。

リスト 2: negamax.rb

```
1  def negamax(p)
2    return p[:side] == :MAX ? p[:value] : -p[:value] if p[:children] == nil
3    p[:children].map{|q| -negamax(q)}.max
4  end
5
6  puts negamax(tree) # => 75
```

Minimax が常に先手から見た評価値を返すのに対して、Negamax は手番がある側から見た評価値を返す（表 1）。たとえば後手番の局面で Negamax が正の値が返したら、それは後手が有利（先手が不利）という意味である。そのために 2 行目では、後手番であれば評価値の符号を反転して返している。言い換えると、Negamax では静的評価関数が手番のある側から見た評価値を返す必要がある。

3 行目には -negamax(q) という式がある。マイナスが付いているのは相手の言い分を反転させるためである。たとえば後手が「わたしは +100 点である」と言うのなら、それは先手にとっては −100 点の意味だからである。

2.2.2 Q&A

Q1. Minimax を Negamax にするとどのくらいよいですか

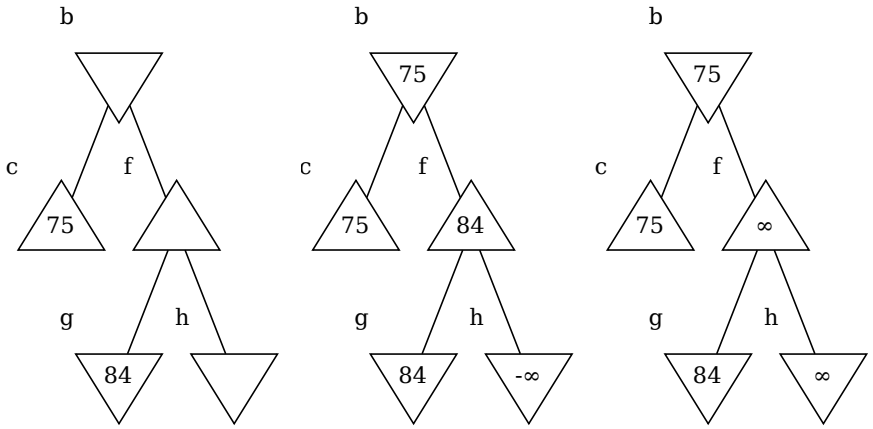
表 1: Minimax 値/Negamax 値の正負

局面	Minimax 値	Negamax 値
先手有利 かつ 先手番	正	正
先手有利 かつ 後手番	正	負
後手有利 かつ 先手番	負	負
後手有利 かつ 後手番	負	正

A1. 性能は変わりません。先手と後手の場合分けが一つなくなるのでコードが簡潔になります。これは後述する Alpha-Beta Pruning を適用したときにとくにうれしいです。

2.3 Alpha-Beta Pruning

Minimax では局面 a の値を求めるために子局面の全ての値を計算したが、実際のところ全ての値を計算する必要はない。たとえば図 3a の h はどのような値であっても b に影響を及ぼすことがない。仮に h に $-\infty$ と ∞ を入れた結果を図 3b と図 3c に示す。どちらも b は 75 であり、h の影響がない。このことから h は探索しなくてよいことがわかる。

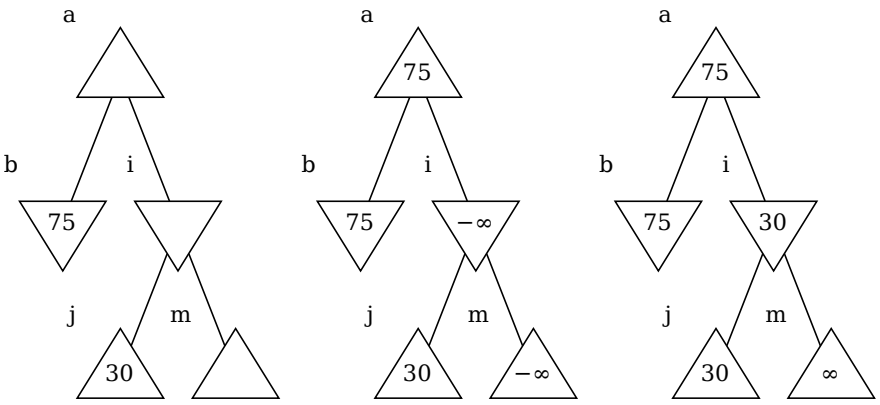


(a) 評価中の b の周辺　(b) h に $-\infty$ を入れた　(c) h に ∞ を入れた

図 3

なぜこのようになっているのかを説明する。① $c = 75$ で、なおかつ b は後手番なので $b = \min(75, f)$ であり、f が 75 を下回らない限り b に採用されないことがわかる。②一方 $g = 84$ で、なおかつ f は先手番なので $f = \max(84, h)$ であり、h がどのような値であっても f は 84 を下回らないことがわかる。①、②より f は h の値に関わらず b に採用されないことがわかる。したがって h を探索する必要がない。

同様に図 4 では局面 m の値を計算する必要がない。



(a) 評価中の a の周辺　(b) m に $-\infty$ を入れた　(c) m に ∞ を入れた

図 4

2.3.1 コード (Ruby)

リスト 3: alphabeta.rb

```
1  def alphabeta(p, alpha, beta)
2    raise 'assertion error' unless alpha < beta
3    return p[:value] if p[:children] == nil
4    if p[:side] == :MAX
5      p[:children].each do |e|
6        alpha = [alpha, alphabeta(e, alpha, beta)].max
7        return beta if beta <= alpha
8      end
9      return alpha
10   else
11     p[:children].each do |e|
12       beta = [beta, alphabeta(e, alpha, beta)].min
13       return alpha if beta <= alpha
14     end
15     return beta
16   end
17 end
18
19 puts alphabeta(tree, -Float::INFINITY, Float::INFINITY) # => 75
```

alpha と beta は探索する評価値の範囲を表す窓である。alpha が窓の下限、beta が窓の上限であり、alpha < beta である必要がある（2 行目）。初期値は $-\infty$ 〜 $+\infty$ である（19 行目）。窓は開区間である。すなわち alpha $< x <$ beta の x が窓に収まる値である。

探索中に評価値が窓に収まらないことがわかった場合は、その局面は実現しないため探索を打ち切ることができる。探索を打ち切る場合の評価値は、alpha 以下の適当な値や、beta 以上の適当な値を返しておけばよい。このとき、ちょうど alpha や beta を返す方法を fail-hard という。alpha 以下や beta 以上の、より真の値に近い値を返す方法を fail-soft という。

探索中に先手番は alpha を押し上げながら探索を進める（6 行目）。このとき、alpha が beta 以上になれば、その局面は窓の範囲外なので beta を返して終了する（7 行目）。これを beta-cutoff という。

探索中に後手番は beta を引き下げなから探索を進める（12 行目）。このとき、beta が alpha 以下になれば、その局面は窓の範囲外なので alpha を返して終了する（13 行目）。これを alpha-cutoff という。

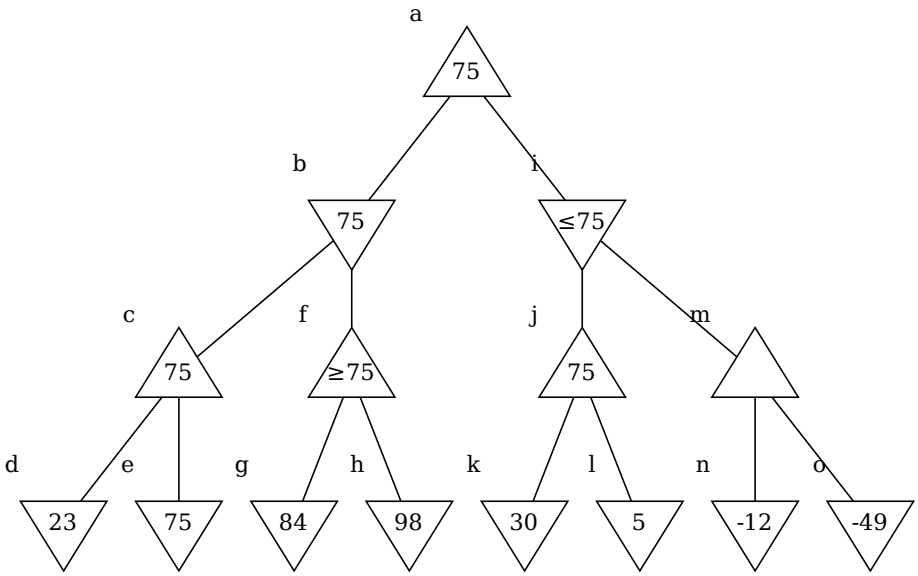


図 5

2.3.2 Q&A

Q1. Minimax を Alpha-Beta Pruning するとどのくらいよいですか

A1. ゲーム木の 1 局面あたりの枝の数を d (degree)、ゲーム木の高さを h (height) とすると、Minimax の計算量が $O(d^h)$、Alpha-Beta Pruning の計算量が最良で $O(d^{h/2})$ なのでだいぶよいです [1]。

Q2. 7 行目の beta <= alpha を beta < alpha にしたらどうなりますか

A2. これは alpha が窓の上限を越えたかどうかを判定する式です。窓は開区間なので alpha == beta のときにすでに alpha は窓の上限を越えています。したがって判定式を beta < alpha にすると、ちょうど alpha == beta のときに alpha が本当は上限を越えているのに上限を越えていないと判定されることになり、枝刈りが弱くなります。

Q3. 7 行目の return beta を return alpha にしたらどうなりますか

A3. beta-cutoff を fail-soft にするということになります。ここでは beta 以上の値を返せばよく、このときの alpha は beta 以上なので alpha を返しても問題ありません。ただし評価値が alpha より下だったときに 9 行目で fail-hard しています。したがって、評価値が beta より上のときは fail-soft して、評価値が alpha より下のときは fail-hard するという一貫性のないコードになってしまいます。全体を fail-soft で実装する方法については 2.5 節で説明します。

2.4 Negamax Alpha-Beta Pruning

Negamax を Alpha-Beta Pruning することができる。Negamax Alpha-Beta Pruning はただの Negamax と同様に、手番のある側から見た評価値を返す。

2.4.1 コード (Ruby)

リスト 4: nega-alphabeta.rb

```
1  def nega_alphabeta(p, alpha, beta)
2    raise 'assertion error' unless alpha < beta
3    return p[:side] == :MAX ? p[:value] : -p[:value] if p[:children] == nil
4    p[:children].each do |e|
5      alpha = [alpha, -nega_alphabeta(e, -beta, -alpha)].max
6      return beta if beta <= alpha
7    end
8    alpha
9  end
10
11 puts negaalpha(tree, -Float::INFINITY, Float::INFINITY) # => 75
```

5 行目の -beta, -alpha では窓をひっくり返している。たとえば窓が $-100 \sim +200$ であれば、ひっくり返した窓は $-200 \sim +100$ となる。これはたとえば先手が持つ $-100 \sim +200$ の窓は、後手から見ると $-200 \sim +100$ の窓だからである。

2.4.2 Q&A

Q1. Minimax Alpha-Beta Pruning と Negamax Alpha-Beta Pruning はどちらがいいですか

A1. 性能は同じです。Negamax Alpha-Beta Pruning のほうがコードが簡潔なのがいいです。

2.5 fail-soft Alpha-Beta

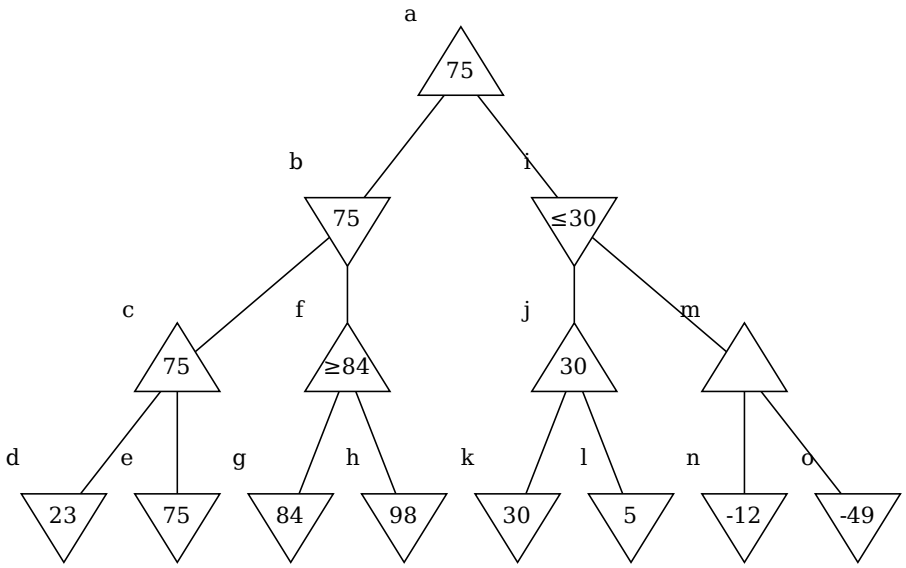


図 6

2.5.1 コード (Ruby)

リスト 5: failsoft-alphabeta.rb

```ruby
def failsoft_alphabeta(p, alpha, beta)
  raise 'assertion error' unless alpha < beta
  return p[:side] == :MAX ? p[:value] : -p[:value] if p[:children] == nil
  score = -Float::INFINITY
  p[:children].each do |q|
    score = [score, -failsoft_alphabeta(q, -beta, -alpha)].max
    alpha = [alpha, score].max
    return score if beta <= alpha
  end
  score
end

puts alphabeta_failsoft(tree, -Float::INFINITY, Float::INFINITY) # => 75
```

2.5.2 Q&A

Q1. fail-hard と fail-soft はどっちがいいですか

A1. 性能は同じです。コードは fail-hard のほうがちょっと簡単です。本稿では扱いませんが置

換表に評価値を保存している場合は fail-soft のほうが真の値に近いのでよさそうな気がします [2]。

2.6 Principal Variation Search

2.6.1 コード (Ruby)

リスト 6: pv-search.rb

```ruby
def pv_search(node, alpha, beta)
  return p[:side] == :MAX ? p[:value] : -p[:value] if p[:children] == nil
  score = -Float::INFINITY
  first = true
  p[:children].each do |q|
    if first
      v = -pv_search(q, -beta, -alpha)
      first = false
    else
      v = -pv_search(q, -alpha - 1, -alpha)
      v = -pv_search(q, -beta, -alpha) if alpha < v && v < beta
    end
    score = [score, v].max
    alpha = [alpha, score].max
    return score if beta <= alpha
  end
  score
end
```

2.6.2 Q&A

Q1. 8 行目を first = false if alpha < v としている実装がありますが、どっちがいいですか

A1. わかりません

3 付録 2：関数についての考察

高階関数は、関数を受け取る関数や、関数を返す関数と説明される。しかしこの説明には関数とは何であるかの説明が足りていない。そこで本章では関数について考察する。

歴史的経緯により関数には次の 2 通りの定義がある [3]：

1. 古典的定義：相伴って変化する数（あるいは量）；すなわち関数は数である
2. 近代的定義：（一意）対応、あるいは対応関係；すなわち関数は対応関係である

ここで数と対応関係は明らかに異なるものであることに注意されたい。たとえば 1 や 2 や 3 が数である。一方、対応関係とは x に対して x^2 が対応するといった関係である。

古典的には関数はたとえば $y = x^2$ とか $f(x) = x^2$ のように書かれ、それぞれ「y は x の関数で

ある (y is a function of x)」、「$f(x)$ は x の関数である ($f(x)$ is a function of x)」、あるいは単に「y は関数である (y is a function)」、「$f(x)$ は関数である ($f(x)$ is a function)」と呼ばれる。

ここで y や $f(x)$ は、x に相伴って変化する数である。したがって関数は数である。ここでは f は関数を表すためのただの記号である。

一方、近代的定義では関数は数ではなく、対応関係であると解釈される。そこでは f こそが関数であり、$f(x)$ は「関数 f の x における値 (value of a function f at x)」と呼ばれる。したがって関数は対応関係である。

以上の違いを表 2 に記す。

表 2

	古典的定義では	近代的定義では
f は	関数を表す記号	関数（すなわち対応関係）
$f(x)$ は	x の関数（すなわち数）	関数 f の x における値

古典的定義においては f はただの記号だが、近代的定義においては f こそが関数である。そこで近代的定義においては f を数式で表す。たとえば式 $f(x) = x^2$ において、f は矢印表記を用いると $f : x \mapsto x^2$ と表せる。ラムダ式を用いると $f = \lambda x.x^2$ と表せる。

関数に近代的定義を導入すると、関数の対応関係そのものを論ずることができる。冒頭で、高階関数とは、関数を受け取る関数や、関数を返す関数であると述べた。ここで関数とは対応関係のことである。すなわち高階関数とは、対応関係を受け取る関数や、対応関係を返す関数と言うことができる。これは高階でない関数が、数を受け取り、数を返すのと比べて、明らかに異なる。

たとえば、受け取った関数を 2 回適用する関数を返す高階関数 apply_twice をラムダ式を用いて次のように定義できる：apply_twice $= \lambda f.\lambda x.f(f(x))$

ここで $\lambda f.\lambda x.f(f(x))$ は f を受け取り (λf)、$\lambda x.f(f(x))$ を返す関数を意味する。返す値である $\lambda x.f(f(x))$ もまた関数である。したがって $\lambda f.\lambda x.f(fx)$ は関数を返す関数であり、高階関数である。

参考文献

[1] Donald E. Knuth and Ronald W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, Vol. 6, pp. 293–326, 1975.

[2] John P. Fishburn. Another optimization of alpha-beta search. *SIGART Newsl.*, Vol. 84, pp. 37–38, 1983.

[3] 公田藏. 近代日本における, 函数の概念とそれに関連したことがらの受容と普及. 数理解析研究所講究録, Vol. 1787, pp. 265–279, 2012.

WCSC-33 PR文章

本資料はWCSC-33PR文章とします。
きのあ将棋の最近の研究としては将棋AIの棋力向上も重視しているものの、より注力＆目指しているのは人に役に立つ将棋AIの研究です。

あわせて、生産性向上などの目的のため GI-49発表のスライド資料を転用しますことをご容赦くださいませ。大半はそのままですがスライドの一部を調整しましたことをここに記します。

作成 2023/03/31

きのあ / Qinoa

ユーザに好まれる
将棋局面の機械生成について研究

ユーザ投稿評価をもとに
将棋 AI を最適化＆機械学習したユーザ満足度の高い局面生成方法

WCSC-33 PR文章
(GI-49発表資料転用)

山田元気

結論から書くと

ユーザに好まれる
将棋局面の機械生成について研究

	good率	投稿数
平手	74.3%	102615
全指定局面	76.0%	88073
詰め将棋(実戦、詰め逃れ含む)	86.0%	36778
機械生成(最適化のみ)	88.9%	3710
機械生成(最適化＆goodbad機械学習評価関数利用)	89.6%	3771

ユーザ満足度について

きのあ将棋サイトにて 対局アンケートを常時設置。
アンケートでは good,badボタンを押すだけ。

満足度は下記のようにgood率とする。
[good投稿数] ÷ ([good投稿数]+[bad投稿数])

good率(満足度)計算例

この局面からの対局は、面白いまたは練習などになりましたか？

good 6 / bad 1　　good / bad

・good投稿数 6
・bad投稿数　1
　　の場合

・6/(6+1)
　good率⇒86%

指定局面を工夫したい

平手	74.3%	...投稿数102615
指定局面すべて	76.0%	...投稿数88073
詰将棋	86.0%	...投稿数36778

・工夫した局面は good率が高い。
・雑な機械生成の局面では good率が低い。

⇒悪い局面が指定局面全体のgood率を押し下げ。
⇒よい局面を人間が生成するのは高コスト。

おおよその局面生成方針

将棋は 勝つとうれしい。
勝ちのときのgood率のほうが高い！

good率は、　勝ち84%　負け69%

…GPW-22 にて、きのあ発表時のデータを引用

確実に勝つ練習は上達に役立つ可能性も高い。
広い意味では詰将棋などもその一環である。

⇒勝てる局面を機械生成する方針に

簡易的な局面機械生成

・将棋AIによるxの計算量において自己対局を実施。
・終局からy手戻す。
・戻した局面にて勝った側から対局。

⇒こんな簡単な方法でいいの？
⇒パラメータはどれくらいがいいの？

パラメータの最適化

自己対局の思考量	詰み20手戻し	詰み30手戻し	詰み40手戻し	詰み50手戻し	投了20手戻し	平均
候補手生成 0.2M	74%	88%	81%	91%	99%	87%
候補手生成 1.0M	66%	91%	93%	100%	84%	87%
候補手生成 2.0M	74%	93%	88%	87%	84%	85%
候補手生成 4.0M	72%	84%	92%	94%	83%	85%
平均	71%	89%	88%	93%	87%	86%

2020年9月23日実験開始。2021年1月15日集計

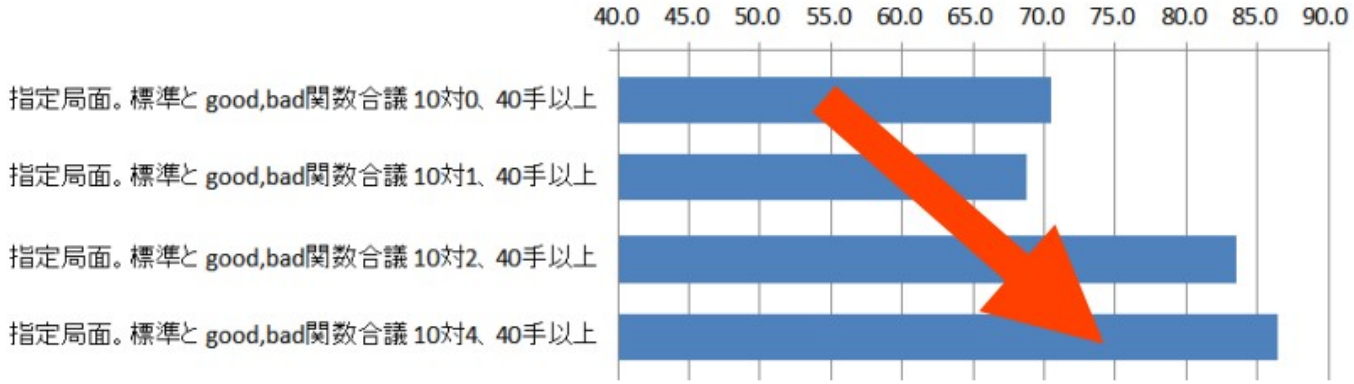
おおよそ、赤枠の中がよさそうだ！

good,bad評価関数

きのあ将棋サイトで、
good,bad投稿を押すと押したところまでの
ユーザ体験局面(ほぼ棋譜)が記録される。
この局面とgood,bad種別を機械学習した評価関数。

⇒good,bad評価関数を局面生成に活用！

good,bad評価関数検証



通常評価関数に対しgoodbad評価関数の重み割合ごとに将棋AIを
4パターン用意した。ユーザは2段階の指定した棋力から、ランダムで選ばれ
たAIとサイト上で対局。パターンごとにgood投稿率を集計検証した。
（ただしgoodbad表関数重みが大きいほど棋力は低下する。）

検証の結果、good率評価関数の重みが大きいAIほど
good率向上の傾向を確認。特に10対4とすると71%→86%と大幅上昇。

このページの集計は40手フィルタを入れて集計しています。

このスライドの
データは、
GPW-22発表の
自己引用です。

局面機械生成研究の結論を再掲載

	good率	投稿数
平手	74.3%	102615
全指定局面	76.0%	88073
詰め将棋(実戦、詰め逃れ含む)	86.0%	36778
機械生成(最適化のみ)	88.9%	3710
機械生成(最適化＆goodbad機械学習評価関数利用)	89.6%	3771

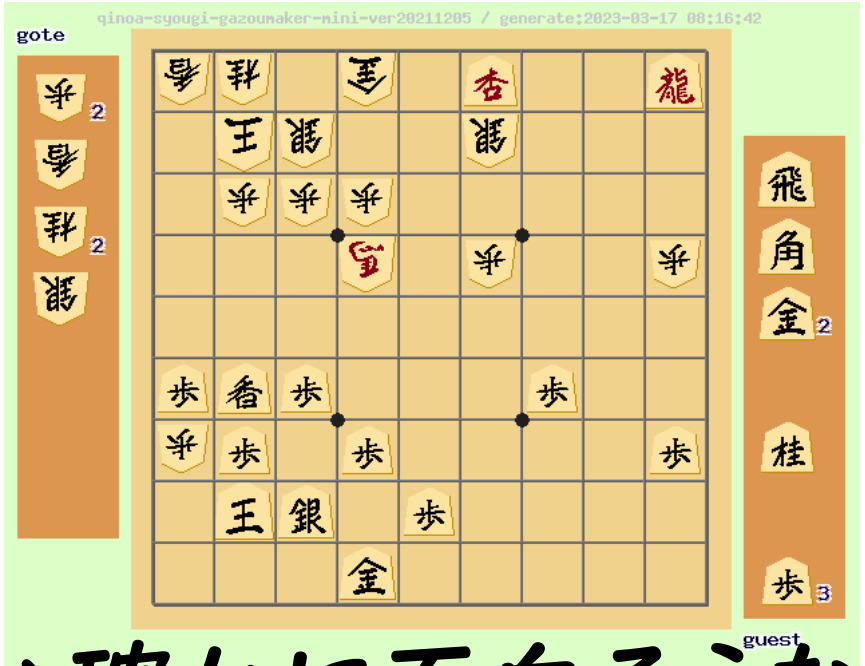
最適化&good,bad評価関数利用の詳細分析

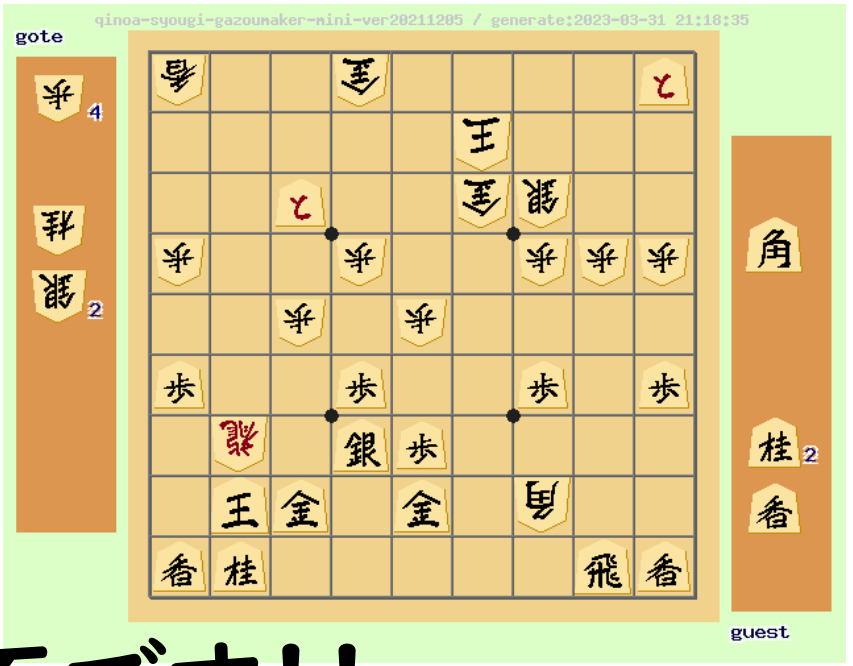
	局面ごとのgood率平均	投稿数
機械生成(最適化のみ)	88.5	3710
10手以上、機械生成(最適化のみ)	87.7	2626
20手以上、機械生成(最適化のみ)	88.5	2309
40手以上、機械生成(最適化のみ)	90.7	1443
80手以上、機械生成(最適化のみ)	87.6	385

	局面ごとのgood率平均	投稿数
機械生成(最適化＆goodbad機械学習評価関数利用)	89.8	3771
10手以上、機械生成(最適化＆goodbad機械学習評価関数利用)	89.6	2686
20手以上、機械生成(最適化＆goodbad機械学習評価関数利用)	90.7	2323
40手以上、機械生成(最適化＆goodbad機械学習評価関数利用)	93.1	1435
80手以上、機械生成(最適化＆goodbad機械学習評価関数利用)	89.7	370

⇒good,bad評価関数利用でよりよくなる。
⇒局面を味わうほど差が出る！？

最適化&good,bad評価関数利用の生成局面





⇒確かに面白そうな局面です!!

※GI-49発表スライドでは右側局面は最適化のみでの生成局面でした。
本資料ではこの誤りを訂正すため局面を差し替えました。
大変ご迷惑をおかけしました。

謝辞



本研究は多数のきのあ将棋サイトのユーザ様の協力の結果得られたものです。
ここに、お礼申し上げます。

ありがとうございました!!

CGP アピール文章

2023/3/27
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
・現在のところ去年からの変更点はほぼなしです。
・大会までにはアピール文書を書き直すくらいに開発が進んでて欲しいです。
・一般に流布している定跡データや、一般に流布している「局面と評価値のセット」、読み筋等は使用しておりません。無駄なこだわりだとは思いますが。

・１から作成

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

・非ビットボード型，利き等を保持

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

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

・評価関数

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

・SIMD等の活用

高速化のためSIMDを活用しております。SIMDは現在評価値の算出、指し手の生成、オーダリング、構造体のコピーが主な使用箇所です。

【第33回世界コンピュータ将棋選手権】

コンピュータ将棋ソフト『元気もりもりニンニクパワー』アピール文

都賀町えいだ

【第33回世界コンピュータ将棋選手権】
『元気もりもりニンニクパワー』アピール文(1)

・Vtuber 都賀町えいだが作成したソフト

・名前は公募＋アンケートによって命名
　名付け親は牛車さん(@gyuusha3)

【第33回世界コンピュータ将棋選手権】
『元気もりもりニンニクパワー』アピール文(2)

・「強い将棋ソフトの創りかた」
　著：山岡忠夫、加納邦彦

これをもとに言語(C#)で作成し
python版を超える棋力になること

ライブラリ「cshogi」は使用しない

【第33回世界コンピュータ将棋選手権】
『元気もりもりニンニクパワー』アピール文(3)

・WCSC32からの改善点
　前回までに作成できなかった部分を追加
　(主にponder関連)
　バグの修正、速度改善など

【第33回世界コンピュータ将棋選手権】
『元気もりもりニンニクパワー』アピール文(4)

・特定の戦型を定跡なしでできるようにする
2023/03/31時点で対応中
間に合わないかも。

大会終了後にできた部分、できなかった部分
について追記します。

山田将棋について（2023 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

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

・探索はαβ中心で、LMRとかFutilityとかの至って普通の技術を使ってます
→なんか評価関数をいじったらLMR効かなくなったのでどうするか（20180331）
→バグをとったら効いたので入ってます(2018大会時点)
→なんか評価関数をいじったら明確に弱くなったどうしよう(20190301)
→なんか静止探索をいじったらまれに動かなくなったどうしよう(20190301)
・いわゆるボナンザメソッドを使ってます、もうちょっと改造できないかな
・オンライン学習を勉強してみたかったので、平均化パーセプトロンを試しています
→今更ながらこれ本当に平均化パーセプトロンなのか・・・？って気もしてきた（20180331）
→怪しかったので試すのをやめました(20190301)
・なんと打ち歩詰めのバグが未だにあったので修正しました(20200330)
・なんと王手されている時に受ける手に未だにバグがあったので修正しました(20200330)
→めったに起きないので強さにあまり影響がない、というかむしろ探索が遅くなって弱くなりました。。。なぜ。。。けどさすがに外せない
・探索がもうちょっといい感じになる予定です(20200330)
・今年は正直何も手を入れられていないです・・・健康はいいぞ(20210331)
・2020年バージョンから探索を丸っと作り直し、評価関数も学習しなおしたら、ちょっとだけ強くなりました。
もっと強くなると思ったのになぜ同じくらいに収束するんだ・・・(20220329)
・今回は誤差レベルでしか強くなってない・・・正直まったく強くなってないと思う・・・(20230331)
以上です。

アピールについては以上なので、思い出話と参考URLを貼っておきますね。

といいつつ、過去のアピール文書をコピーしたところも多いですが。。。

毎年なんだかんだ起きるので、参考URLは増える一方です。

■始めての参加

始めての参加は第17回の時で、選手としてではなく、アルバイトとして小谷研から徴集されました。

場所はかずさアークでして、近くのホテルに泊まるとバイト代から足が出るという訳の分からない状態だったのですが、

世界で初めて？パズルで博士を取ることになる先輩にそそのかされ（なぜか今同僚）、

君津のネカフェに3泊4日し、毎朝となりのマックでご飯を食べてました。

初日は全く寝ることができず、すごくつらい思いをしたことを覚えています。

もう二度とネカフェで3泊4日はしたくありません。

第17回大会

<http://www2.computer-shogi.org/wcsc17/>

かずさアーク

<http://www.kap.co.jp/>

自遊空間君津店

<https://jiqoo.jp/shop/9931865/>

マクドナルド君津店

<https://map.mcdonalds.co.jp/map/12031>

■始めての出場と、（恐らく大会史上初の）プログラム名リジェクト

実は当初プログラム名は、「(´・ω・`)」にしていたのですが、

運営の方から「読めません」と言われリジェクトされ、今のとても強そうな名前になりました。

最近の事例を見る限り、読み方をつけておけばリジェクトにはならなかったぽいので、ちょっと後悔をしています。

デビュー戦はなかなかに熱かったです、白砂将棋さんと対局させていただきました。

あの負け方（王手千日手負け）は二度と忘れることは無いでしょう、悔しかったw

実は二次予選に行って20位でしたすげえ。

第21回大会

<http://www2.computer-shogi.org/wcsc21/>

■始めての賞罰

・独創賞受賞（解説記事の生成）

・（恐らく大会史上初の）アピール文書リジェクト

2012年に独創賞いただきました、ﾔｯﾀｰ

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

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

第22回大会

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

■会場に行かずに近くで遊ぼう

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

まだ試行回数が足りないと思うので、今年も行きたいと思います。

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

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

行った場所は下のほうにまとめておきました。

2016年には罰ゲームでうまるちゃんやりました。

その際は、（確か菅井先生だったかな）プロ棋士の方が「なんか変な人いるんですけど大丈夫なんです

か？」と運営の方に相談されていたそうです、すみません、ぼくは大丈夫な人です。
また、千田先生からは「ちょっと身長が高すぎるかもしれませんね」とご指導いただきました、ありがとうございました。
この話を最近（2018年夏ころ？）小谷研の後輩さんにお話ししたところ、「マジっすか、ご褒美じゃないですか！？」と言っていただいたので、今後ぼくの人生におけるご褒美イベントの一つとして大切にすることにします。
ところでコスプレは罰ゲームだと思っていたのに、たぬきさんとか自発的にコスプレされているので、コスプレは罰ゲームじゃなかったんだなあと思いました。
もうやりたくありませんが、とりあえず一式は取ってありますので、うまるちゃんになりたい人がいればお貸しすることは可能です。
2018年は永瀬先生のお父様のお店である川崎家に行き、ネギチャーシューラーメンを食べました、辛みがアクセントになって非常においしかったです。
家系ラーメンは大好きですし、きっと二日目は暇になるので（え、今年もぜひ行きたいと思います、もしよかったら皆さん行きましょう！
(20200330追記)
去年はラーメン食べている間に対局が始まってしまい会場入りが間に合わなかったのですが、きちんと対局がスタートしていました。
自動化されているって素晴らしいなって思いました、もう会場いかなくてずっと川崎家でラーメン食べてもいいですね。
あと今年は二次予選に進出しないと二日目に川崎家いけないので、是が非でも初日に川崎家に行っておきたいと思います。
(20220329追記)
地元のおいしいレストラン（とある弁護士事務所のそばだったりする）でランチを食べることを楽しみに対局してました。
だけど5/3は月曜日で確か空いてなかったのよな・・・。5/5に食べに行った記憶がある。
今年は川崎家だ・・・行きたい・・・

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/>

停車場

<https://tabelog.com/chiba/A1202/A120202/12000477/>

■まさか自宅から大会に参加することになるとは・・・

2020年はコロナの影響もあり、まさかのリモートでの大会となりました。

割と暇な時間が長かったので、大会中にずっとゲームをやっていたのは内緒です。

下記のゲーム、渋滞を起こして街が死ぬってパターンが多い気がする。

シティーズスカイライン

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

■送りバントはいいぞ

自分のプログラム名について一応拾っておきますか。

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

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

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

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

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

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

アピール文書を書きながらジャイアンツの開幕戦（2023年）を観ているのですが、村神様が初打席でホームラン打ったそうです。

WBCでは味方になってくれましたが、敵になった村神様は恐怖でしかないですね、早くFAでジャイアンツに来ないかな。

■目標

まったりゆうちゃんを倒して師匠超えしたいです。

最近は直対すると勝てるケースが多いのですが、順位で勝てないケースがたまにあるので、ぜひ勝ちたいですね。

あとできれば勝ち越ししてみたいですし、久しぶりに2次予選にも進出してみたいです・・・！

去年(2021年)は二日目暇になることを見越して、あらかじめ桃鉄やる約束を取ってしまってました。

その時他のメンバーにボコられて悔しかったので感想戦をやったのですが、「1年目の5月のムーブが悪い」と言われました、なんて厳しいゲームだ。

桃太郎電鉄 ～昭和 平成 令和も定番！

<https://www.konami.com/games/momotetsu/teiban/>

―昨年は惜しかったんですよね・・・王手ラッシュして逃げられるっていう負け方しました。。。あの対局勝ってれば上に上がれたらしい・・・。
同じ方に去年当たったのですが、レベルめっちゃ上がってて、手も足も出ませんでした・・・。

今年も去年同様に、CSA運営の皆さんも頭を抱えられたことと思います。
まだ予断を許さない状況でもありますが、川崎での開催に向けて進めてくださっている運営の皆様には心より御礼申し上げます。

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。

昨年と変わらず。昨年のアピール文書を参照ください。
https://www.apply.computer-shogi.org/wcsc32/appeal/KatsudonShogi/katsudonshogi_apeal_wcsc32_v2.pdf

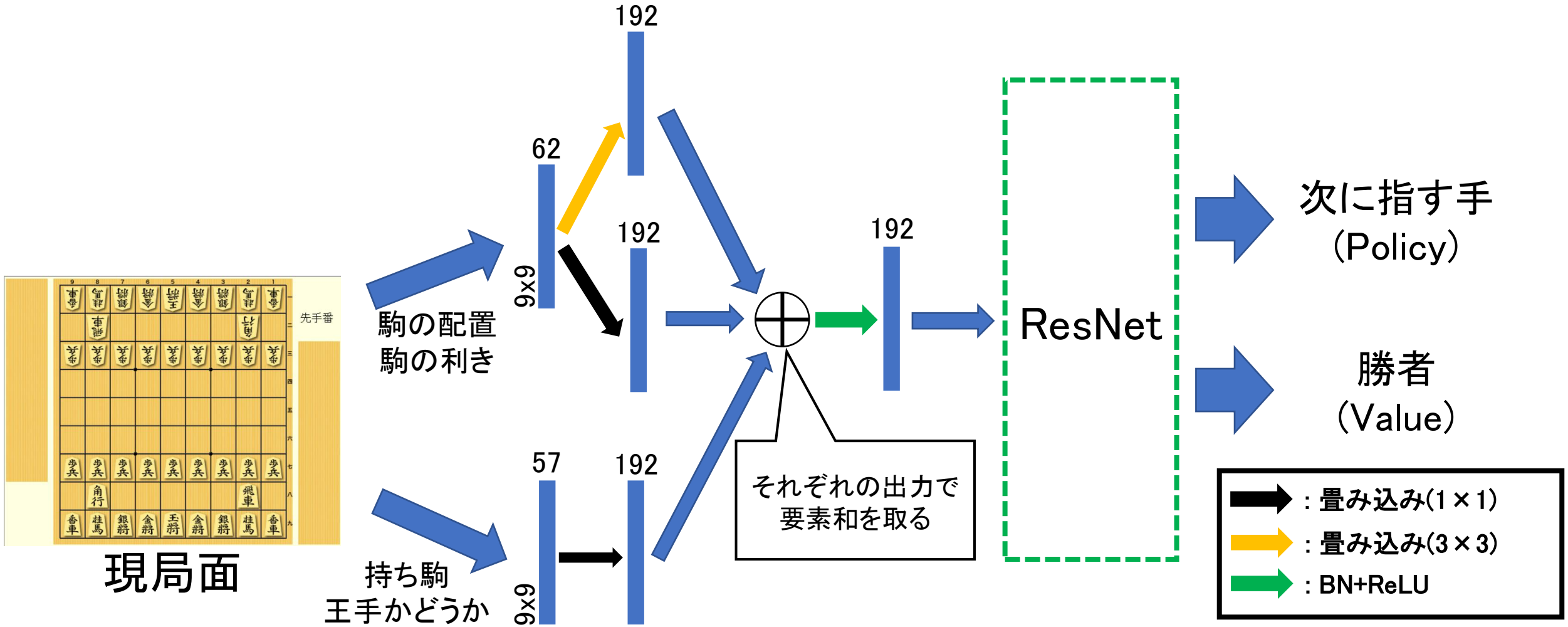
ponkotsu アピール文書(仮)

概要

- Deep Learningを使用した将棋AI
- ネットワーク構造にDenseNetを採用
- 5月開催の世界コンピュータ将棋選手権に向け強化中

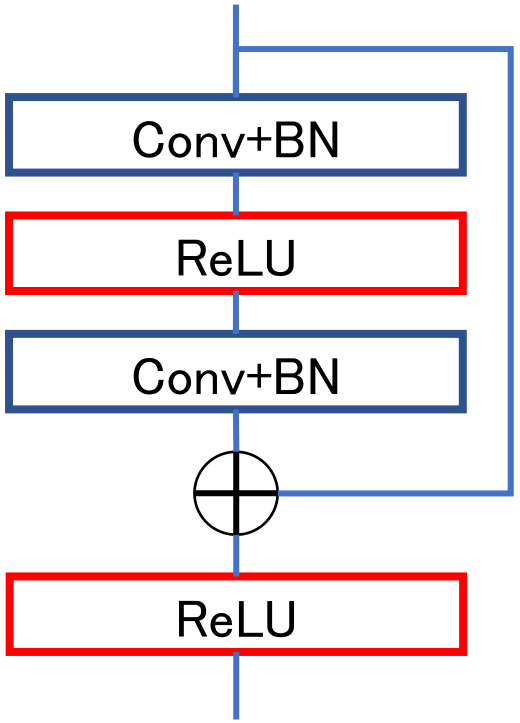
Policy Value Network

- ほとんどのAIでネットワーク構造にResNetを採用



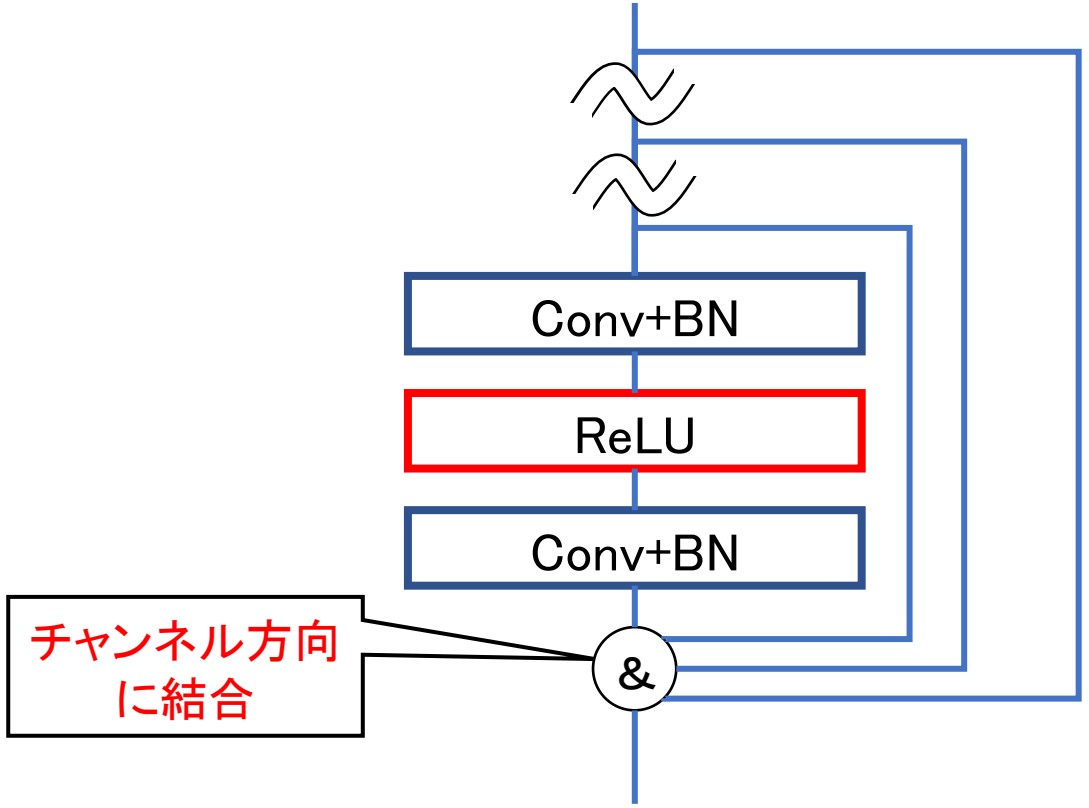
DenseNet

ResNet Block



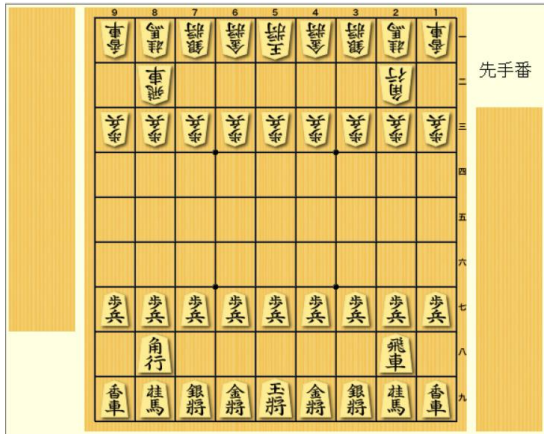
- 直前の入力と出力の残差を学習
- 入力と出力で要素和を取る

Dense Block



- 直前の層までのすべての層とスキップ接続する
- 入力と出力でチャンネル方向に結合する

提案手法(ネットワーク全体)



現局面

駒の配置
駒の利き

持ち駒
王手かどうか

62
9x9
192
192
57
9x9
192

それぞれの出力で
要素和を取る

DenseNet

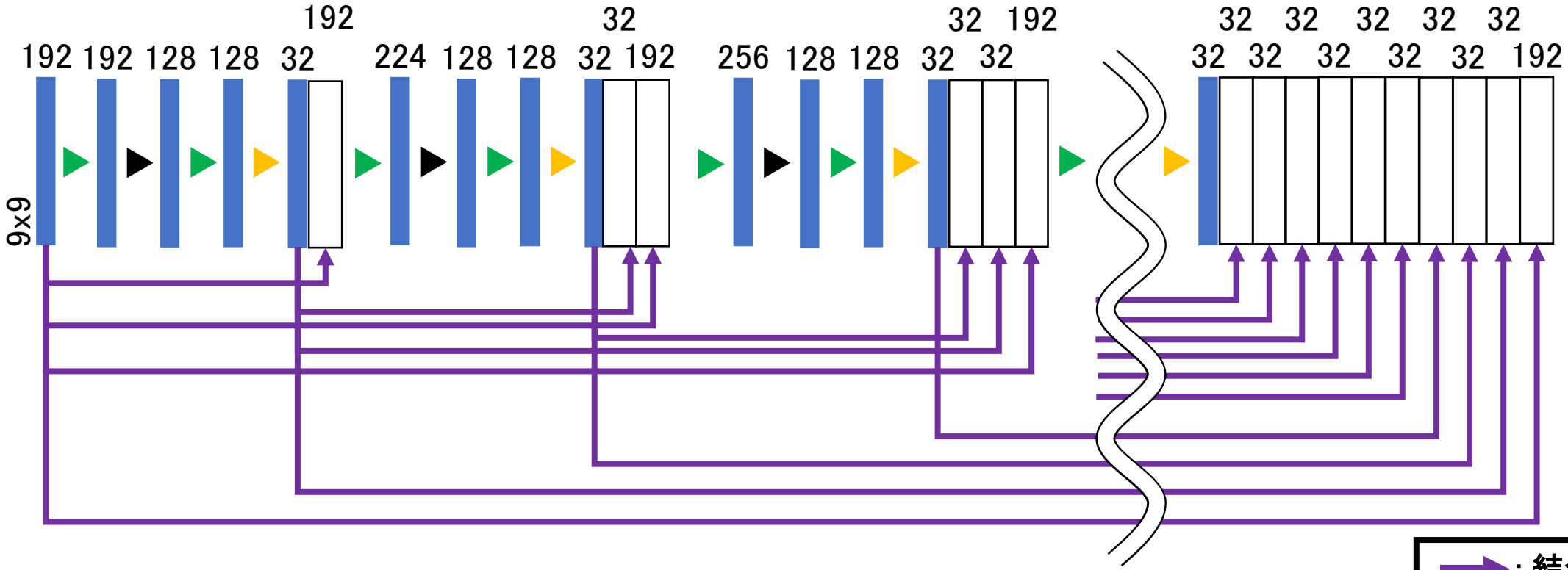
次に指す手
(Policy)

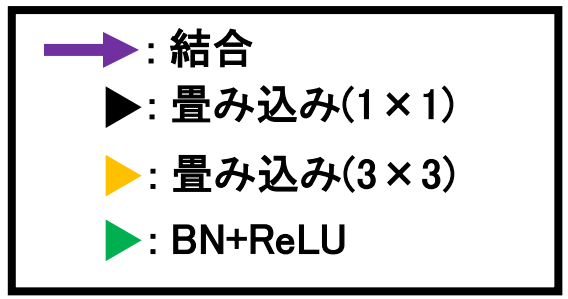
勝者
(Value)

: 畳み込み(1×1)
: 畳み込み(3×3)
: BN+ReLU

提案手法(DenseNet)

- Dense Blockのレイヤ数は10





学習方法

学習データ	• floodgate(将棋AI対局サイト)の2019年～2021年5月の棋譜 • 水匠3改の棋譜 • dlshogi with GCTの棋譜
テストデータ	floodgateの2017年～2018年の棋譜
最適化手法	SGD (lr=0.01)
損失関数	Policy: Cross Entropy Value: Binary Cross Entropy
Epoch数	144
バッチサイズ	1024

結果(識別精度)

- テストデータ(floodgateの2017年～2018年の棋譜)による Policy(次に指す手)とValue(勝者)の精度
- Policy・Valueともに提案手法のほうが高精度

Model	Policy Accuracy(%)	Value Accuracy(%)
Resnet10(32 channel)	39.5	70.7
DenseNet10(提案手法)	**43.8**	**72.8**

学習方法

最適化手法	SGD (lr=0.01)
評価値の重み	0.333
学習率	0.2から開始 10、30、50エポック目に1/10ずつ小さくする
重みの平均を取る間隔	250学習ステップごと
損失関数	Policy: Cross Entropy Value: Binary Cross Entropy
Epoch数	144
バッチサイズ	1024

結果(識別精度)

- テストデータ(floodgateの2017年～2018年の棋譜)による Policy(次に指す手)とValue(勝者)の精度
- Policy・Valueともに学習テクニックを使用したほうが高精度

Model	Policy Accuracy(%)	Value Accuracy(%)
DenseNet10	43.8	72.8
DenseNet10+ ブートストラップ+ 学習率スケジューラ+ SWA+AMP	**44.5**	**73.7**

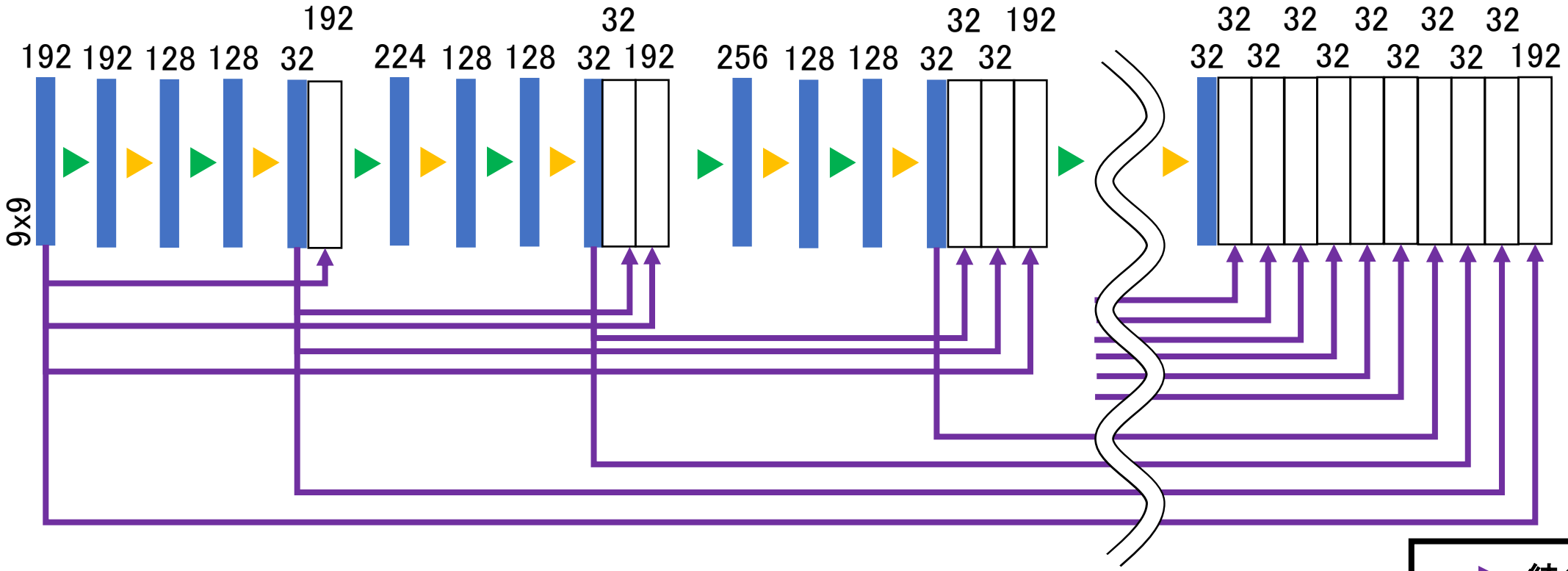
結果(戦績)

- floodgate(将棋AI用の対局サイト)で204勝118敗、
  レーティング3803を達成した
  (参考: 昨年の世界大会一次予選突破ラインが約3100)
- 少なくともプロ棋士よりは強い(対プロの期待勝率は93%)
  (プロ棋士のレーティングは3300と言われている)

rating (Δday, Δweek)	approximated rating in shogi club 24	wins (Δday)	losses (Δday)	%
3803 (+0, -12)	3737	204 (+0)	118 (+0)	0.634

ネットワークの改良(DenseNet)

- Dense Blockのレイヤ数は10
- 1×1の畳み込みをすべて3×3に変更



: 結合
: 畳み込み(3×3)
: BN+ReLU

結果(識別精度)

- テストデータ(floodgateの2017年～2018年の棋譜)による Policy(次に指す手)とValue(勝者)の精度
- Policy・Valueともに改良後のほうが高精度

Model	Policy Accuracy(%)	Value Accuracy(%)
DenseNet10(改良前)	44.5	73.7
DenseNet10(改良後)	**46.9**	**74.7**

事前学習

課題点

- 振り飛車のような出現頻度の低い戦型に対して正確な局面評価ができない(相手が飛車を振ると400点近い評価値を出す)

→学習データの戦型が(居飛車に)偏っている

多様な序盤を指すAIのデータで事前に学習を行う

学習方法(事前学習)

学習データ	• AobaZeroの棋譜 • elmoの棋譜 • GCTの棋譜 • 水匠の相入玉局面の棋譜
テストデータ	floodgateの2017年～2018年の棋譜
学習率	0.2から開始 10、30、50エポック目に1/10ずつ小さくする
損失関数	Policy: Cross Entropy Value: Binary Cross Entropy
Epoch数	106
バッチサイズ	1024

学習方法(本学習)

学習データ	• floodgate(将棋AI対局サイト)の2019年～2021年5月の棋譜 • 水匠3改の棋譜 • dlshogi with GCTの棋譜 • 水匠の相入玉局面の棋譜
テストデータ	floodgateの2017年～2018年の棋譜
学習率	0.0002
損失関数	Policy: Cross Entropy Value: Binary Cross Entropy
Epoch数	144
バッチサイズ	1024

結果(識別精度)

- テストデータ(floodgateの2017年～2018年の棋譜)による Policy(次に指す手)とValue(勝者)の精度
- Policy・Valueともに事前学習を行ったほうが高精度

Model	Policy Accuracy(%)	Value Accuracy(%)
DenseNet10 (事前学習なし)	46.9	74.7
DenseNet10 (事前学習あり)	**47.3**	**74.9**

結果(戦績)

- floodgate(将棋AI用の対局サイト)で54勝9敗、
  レーティング3973を達成した
  (参考: 昨年の世界大会一次予選突破ラインが約3100)
- ネットワークの改良・事前学習の実施によりレーティングが
  170上昇(前のバージョンに対する期待勝率は73%)

rating (Δday, Δweek)	approximated rating in shogi club 24	wins (Δday)	losses (Δday)	%
3973 (+3973, +3973)	3899	54 (+9)	9 (+3)	0.857

使用ライブラリ

- dlshogi

使用データ

- 「強い将棋ソフトの創り方」付属データ
- GCT学習データ
  (URL: https://tadaoyamaoka.hatenablog.com/entry/2021/05/06/223701
  2023年3月29日アクセス)

参考文献

- 「強い将棋ソフトの創り方」（山岡忠夫著、マイナビ出版）
- 「Densely Connected Convolutional Networks」
  （Gao Huang、Zhuang Liu、Laurens van der Maaten、Kilian Q. Weinberger）

まったりゆうちゃんのアピール文書 2023

去年とほとんどおなじである。複数のシステムを動作させて、その間でデータをやり取りして並列動作をする方法を検討している。以下は前回とほぼ同様の説明である。

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

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

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

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

第33回世界コンピュータ将棋選手権

baron

アピール文書

2023年4月18日　湯川和雄

【開発者の言葉】

参加は2回目になります。wcsc32はチームで参加いたしました。

今年は個人で参加いたします。

昨年は会場で参加いたしまして、開発者の方からディープラーニングの学習方法や教師データなど、ご教示いただき参考にさせていただきました。

今年はオンラインで参加いたします。

baronの棋力としてはfloodgateのレートで、昨年は2000前後、今年は3200前後かと思われます。

第3回世界将棋AI電竜戦本戦から大きな違いはありません。全く同じというのも忍びないので、定跡部分は最新の棋譜データなどを使い、手を加えてみようと考えております。

少し足掻いてみようと思い、追加学習もしてみました。(2023/04/18)

【baronの由来】

baronと書いて「ばろん」と読みます。

漢字で書くと、馬の狼で「馬狼（ばろん）」です。牙の狼ではないです。

将棋の馬って強いですよね。将棋は角（馬）の働き次第、と感じてしまうぐらい強いです、金銀3枚分ぐらいの強さでしょうか。

馬の狼だともっと強そうです。

baronの意味を調べると、男爵（だんしゃく、英:baron）と出てきます。将棋男爵ってかっこいいです。

『〇をすませば』にでてくるバロンもかっこいいです。

【第3回世界将棋AI電竜戦本戦アピール文書】

https://drive.google.com/file/d/14hTVA0b6HC-tBIy7cWVin-jUPRhj82bh/view

【独自の工夫】

独自の定跡処理を搭載しております。

多少のランダム性の確保もしております。

定跡を約4万局登録。手数では約600万手登録。

大会に向けて最新の棋譜データを取り入れたものにしたいと思います。

【ベース】

ベースは「将棋AIで学ぶディープラーニング」「強い将棋ソフトの創り方」の著書を参考にさせていただいております。

【系統】ディープラーニング系ソフト

【開発言語】Python

【学習環境】GoogleColabPro/Pro+

【使用・参考にしたライブラリ】pytorch、python-shogi、cshogi、dlshogi、dlshogi2

【開発者Twitter】

https://twitter.com/k_yuka_2020

質問のアピールという言葉の意味がわからない。
アピールは、野球で３塁ランナーが外野フライを取る前にスタートしたときに３塁にボールを送って審判に言ってアウトにする手法だと思うが将棋でなぜこの言葉がでてくるのかわからない。
著作権に関する質問だとすると、隠岐は自分でコーディングしたものであります。
思考部だけで約１０９８万ステップくらいあります。また別に詰将棋部が7453ステップ、画像部が４万ステップくらいあると思います。
だから、ステップ数だけだと他のどのソフトにも勝つと思います。
ただ、これを自分だけでコーディングしたかとなると、偽で、実はコンピュータにコーディングさせております。
つまり、将棋の思考部はワンパターンになりやすく、それを利用して将棋ソースを作るプログラムを開発してそのプログラムにやらせてます。
下記が隠岐の思考部のソースの一覧です。

```
ドライブ C のボリューム ラベルは Windows です
 ボリューム シリアル番号は 6A33-AF78 です

 C:¥VC2019¥OkiSikou のディレクトリ

2022/12/24  16:15    <DIR>          .
2022/12/24  16:15    <DIR>          ..
2022/11/02  12:19           434,483 aa0.h
2022/11/05  16:37           965,946 aa1.h
2022/11/02  12:19           934,441 aa2.h
2022/11/02  12:19           905,645 aa3.h
2022/11/05  16:37           661,680 aa4.h
2022/11/19  10:48           910,856 aa5.h
2022/11/02  12:19           915,388 aa6.h
2022/11/02  12:19           890,002 aa7.h
```

2022/11/02 12:19 121,598 aa8.h
2022/11/02 12:19 928,672 aa9.h
2022/11/02 12:19 930,887 aaa.h
2022/11/02 12:19 904,815 aab.h
2022/11/02 12:19 899,007 aac.h
2022/11/02 12:19 888,186 aad.h
2022/11/02 12:19 921,541 aae.h
2022/12/13 12:24 195,205 aaf.h
2022/12/04 17:46 901,242 ab1.h
2022/12/04 17:40 699,576 ab2.h
2022/11/11 10:21 907,728 ag1.h
2022/08/05 15:37 905,252 ag2.h
2022/11/11 10:21 863,759 ag3.h
2022/12/02 10:07 513,348 Ai.h
2007/02/26 16:50 332 Ai_g.cpp
2007/06/30 07:42 333 Ai_s.cpp
2022/12/20 17:50 887,899 aj1.h
2022/12/20 17:31 847,574 aj2.h
2022/12/22 09:30 855,557 aj3.h
2022/12/20 17:31 389,874 aj4.h
2022/12/20 17:31 900,100 aj5.h
2022/12/20 17:31 870,677 aj6.h
2022/12/20 17:31 261,747 aj7.h
2022/12/20 17:31 900,636 aj8.h
2022/12/20 17:31 898,224 aj9.h
2022/12/20 17:31 903,969 aja.h
2022/12/21 10:45 416,678 ajb.h
2022/12/11 06:17 878,892 ak1.h
2022/12/08 17:45 81,336 ak2.h

2022/12/10 09:16 916,091 ak3.h
2022/12/08 17:25 905,948 ak4.h
2022/12/12 06:14 905,985 ak5.h
2022/12/10 16:22 899,662 ak6.h
2022/12/08 17:25 904,184 ak7.h
2022/12/11 06:17 379,876 ak8.h
2009/09/13 16:05 444 amari.cpp
2022/08/20 14:59 8,665 ana.h
2007/02/26 16:50 765 Ana_g.cpp
2007/02/26 16:50 747 Ana_s.cpp
2022/09/11 09:57 930,673 ao1.h
2022/09/11 09:52 779,533 ao2.h
2022/12/24 16:15 859,862 as1.h
2022/12/24 16:04 871,528 as2.h
2022/12/24 16:04 192,886 as3.h
2022/12/24 16:04 901,232 as4.h
2022/12/24 16:04 892,935 as5.h
2022/12/24 16:04 689,947 as6.h
2022/12/11 12:12 971,405 at1.h
2022/12/11 12:31 975,592 at2.h
2022/10/17 06:17 907,175 az1.h
2022/12/18 08:39 884,638 az2.h
2022/12/18 09:04 599,057 az3.h
2022/10/22 06:02 358,709 Bougin.h
2007/02/26 16:50 427 Bougin_g.cpp
2007/02/26 16:50 428 Bougin_s.cpp
2009/09/05 15:20 684 check_2fu.cpp
2015/06/28 06:00 2,820 core0a.cpp
2015/06/28 05:53 3,828 core0y.cpp

2022/12/07 13:19 266,404 Eishun.h
2007/06/22 12:12 340 Eishun_g.cpp
2007/06/22 12:12 341 Eishun_s.cpp
2010/09/11 11:21 4,209 g11_map.cpp
2015/10/21 15:51 38,562 ga_total.cpp
2015/07/13 10:26 1,030 get_sasite.cpp
2011/01/04 09:08 13,565 gftotal.cpp
2015/02/26 09:53 20,985 gfttotal.cpp
2010/06/30 09:22 5,835 gictotal.cpp
2010/09/14 09:21 2,054 gobtotal.cpp
2005/11/08 06:12 4,768 Gote.h
2010/04/20 13:23 523 gou1total.cpp
2016/11/25 06:12 326 gou2total.cpp
2012/01/08 11:32 877 gou3total.cpp
2011/05/29 09:19 1,205 gou4total.cpp
2010/09/14 08:54 1,279 gou5total.cpp
2010/09/14 09:05 1,051 gou6total.cpp
2010/09/14 09:08 988 gou7total.cpp
2010/09/15 07:15 6,579 Goutotal.cpp
2011/01/15 10:28 29,165 Gqctotal.cpp
2010/10/04 14:44 18,649 gqctotal2.cpp
2010/09/11 10:12 1,754 gq_total.cpp
2010/10/04 14:49 1,684 gq_total2.cpp
2021/03/18 06:15 32,368 grctotal.cpp
2010/09/09 08:50 16,105 gtntotal.cpp
2010/12/29 13:53 606 gu1total.cpp
2010/09/14 09:32 1,786 guctotal.cpp
2022/11/22 10:30 863,968 gx1.h
2022/08/24 08:57 91,686 gx2.h

2022/10/05 16:45 902,487 gx3.h
2022/08/24 08:35 882,508 gx4.h
2022/11/22 10:24 226,670 gx5.h
2011/10/08 17:50 23,935 gxctotal.cpp
2010/04/20 06:06 22,298 gztotal.cpp
2009/12/10 08:19 3,952 g_kiki.cpp
2021/06/01 06:15 16,564 g_total.cpp
2021/04/30 11:09 386 hasshu.cpp
2004/02/26 22:26 15,987 hikyo_s.h
2000/04/08 05:06 2,115 hikyo_u.h
2019/02/24 17:11 54,145 Hineri.h
2014/03/27 06:31 392 Hineri_g.cpp
2014/03/27 06:31 393 Hineri_s.cpp
2010/12/20 15:11 82,574 hisshi.cpp
2011/01/03 10:26 7,203 hisshi5.cpp
2011/01/09 06:22 14,674 hi_s.h
2007/02/26 16:50 5,628 hi_sort.cpp
2011/01/09 06:27 4,550 hi_u.h
2016/04/23 10:08 6,273 h_open.cpp
2022/04/20 11:52 19,348 ibiana.h
2012/04/29 10:09 943 inaniwa.h
2011/05/21 13:58 211 Inaniwa_g.cpp
2011/05/21 13:58 212 Inaniwa_s.cpp
2016/10/13 06:30 1,048,576 index_s.dat
2022/12/15 12:41 1,024 INFO.dat
2016/05/04 08:57 2,038 Jissen.lib
2014/05/19 10:25 13,199 Joseki.cpp
2021/06/21 10:46 167,453 Joshiki.h

2019/10/17 10:44 37,575 Joshiki2.h
2007/02/26 16:50 1,532 Josiki_g.cpp
2007/02/26 16:50 1,538 Josiki_s.cpp
2004/03/11 05:45 5,449 kaku_s.h
2004/03/11 22:40 2,666 kaku_u.h
2010/07/13 15:25 483 kiki.cpp
2012/07/15 09:07 34,975 Kousoku.cpp
2009/08/19 09:20 23,950 ksort.cpp
2004/02/12 22:45 2,171 kyo_s.h
2004/02/14 07:23 3,710 kyo_u.h
2010/09/16 08:34 3,640 mawari.cpp
2011/09/07 06:16 4,156 mawari_ten2.cpp
2022/06/21 06:31 84,236 Migi4ken.h
2007/02/26 16:50 394 Migi_g.cpp
2007/02/26 16:50 395 Migi_s.cpp
2010/09/05 14:52 326 modori.cpp
2010/09/04 09:13 563 move_flag.cpp
2022/12/24 15:14 194,205 Mukai.h
2018/05/26 06:29 378 Mukai_g.cpp
2018/05/26 06:29 381 Mukai_s.cpp
2022/12/17 11:17 215,403 Naka.h
2012/11/28 13:58 380 Naka_g.cpp
2013/10/18 06:56 381 Naka_s.cpp
2010/09/15 13:50 1,909 narabikae.cpp
2010/09/15 13:50 1,671 narabikae_n.cpp
2010/09/15 16:42 1,475 nigeru.cpp
2010/09/15 16:42 975 nigeru2.cpp
2000/10/08 08:49 333 ochi10_s.h
2000/04/08 08:34 303 ochi10_u.h

2009/02/25 15:43 20,098 Ochi2_s.h
2005/01/05 22:22 10,295 Ochi2_u.h
2004/04/01 23:01 18,085 Ochi4_s.h
2004/04/02 22:39 4,097 Ochi4_u.h
2006/04/19 06:18 13,240 Ochi6_s.h
2000/04/08 08:09 1,221 Ochi6_u.h
2010/12/26 05:26 2,501 Ochi8_s.h
2018/03/04 10:22 910 Ochi8_u.h
2007/02/26 16:50 1,896 Ochi_s.cpp
2007/02/26 16:50 689 Ochi_u.cpp
2015/01/03 06:54 37,412 OkiSikou.aps
2021/10/28 12:16 16,912 OkiSikou.cpp
2006/02/15 21:41 146 OkiSikou.def
2006/02/15 21:57 527 OkiSikou.h
2022/01/06 16:53 3,145 OkiSikou.rc
2021/04/25 16:34 1,232 OkiSikou.sln
2022/12/17 09:35 30,889 OkiSikou.vcxproj
2022/12/17 09:35 53,466 OkiSikou.vcxproj.filters
2022/02/28 17:24 864 OkiSikou.vcxproj.user
2009/02/12 13:09 3,522 ouchifu.cpp
2022/02/05 09:21 37,658 oute.h
2007/02/26 16:50 335 Oute_g.cpp
2007/02/26 16:50 338 Oute_s.cpp
2009/01/06 11:29 2,333 outori.cpp
2016/09/30 10:35 2,560 out_moji2.cpp
2022/08/23 16:01 177,939 p1sort.cpp
2022/09/03 10:17 85,387 p2sort.cpp
2021/10/08 06:14 170,726 p3sort.cpp
2021/11/08 16:21 223,947 p4sort.cpp

```
2022/09/17  08:17         244,276 p5sort.cpp
2022/08/19  15:55         224,304 p6sort.cpp
2022/03/07  07:01         124,175 p7sort.cpp
2021/07/29  14:02           5,886 pmsort.cpp
2022/06/04  05:48         257,701 psort.cpp
2021/04/30  11:09          29,778 ransu.cpp
2006/02/15  21:41           2,402 ReadMe.txt
2021/11/01  18:43    <DIR>        res
2006/02/15  21:41             375 Resource.h
2021/11/10  05:35          10,352 rhasshu.cpp
2016/10/13  06:32          73,728 r_ems.dat
2010/09/11  11:20           4,201 s11_map.cpp
2021/04/30  11:09             843 sahasshu.cpp
2010/09/15  17:03             583 sakujo_y.cpp
2022/10/09  15:56         192,124 Sanken.h
2022/01/16  16:03             384 Sanken_g.cpp
2013/11/18  11:18             387 Sanken_s.cpp
2009/03/12  05:49             894 sasu.cpp
2015/10/21  15:32          38,586 sa_total.cpp
2010/09/07  06:16           1,757 seme.cpp
2022/07/18  13:24          42,613 semeru.h
2010/09/07  09:44           2,315 seme_h.cpp
2021/10/28  10:14           2,597 seme_r.cpp
2020/06/06  11:29          11,279 Senpo_g.cpp
2020/06/06  11:27          11,532 Senpo_s.cpp
2005/01/29  06:20           4,847 Sente.h
2011/01/04  09:08          13,598 Sftotal.cpp
2011/06/07  06:26          21,483 sfttotal.cpp
2009/04/04  14:38       1,666,568 sfu_jun.cpp
```

2009/04/05 07:38 21,702 sgin_jun.cpp
2012/03/14 16:58 21,863 shi_jun.cpp
2010/06/30 09:22 5,815 Sictotal.cpp
2022/10/15 15:04 359,396 Siken.h
2007/06/04 08:24 386 Siken_g.cpp
2007/06/04 08:24 385 Siken_s.cpp
2022/12/25 06:35 5,936 sikou.txt
2022/12/24 15:19 163,094 sikou11.cpp
2010/10/07 13:03 1,390 sikou11a.cpp
2022/12/13 12:24 922,374 sikou11aa.cpp
2022/12/04 17:46 650,694 sikou11ab.cpp
2022/11/11 10:21 700,431 sikou11ag.cpp
2022/12/22 06:29 723,064 sikou11aj.cpp
2022/12/11 05:50 690,345 sikou11ak.cpp
2022/09/11 09:52 495,768 sikou11ao.cpp
2022/12/24 16:09 863,734 sikou11as.cpp
2022/12/11 12:31 412,163 sikou11at.cpp
2022/10/16 17:15 210,460 sikou11az.cpp
2022/11/22 06:29 687,729 sikou11gin.cpp
2010/10/17 05:56 4,861 sikou11naru.cpp
2014/07/04 16:26 2,392 sikou11oute.cpp
2021/06/11 11:06 7,889 sikou11wz.cpp
2015/03/09 19:10 1,459 sikou11y.cpp
2022/12/13 09:34 711,482 sikou11yb.cpp
2022/12/09 06:09 748,549 sikou11yc.cpp
2022/12/18 17:48 576,739 sikou11yj.cpp
2022/12/13 06:14 880,948 sikou11yk.cpp
2022/12/23 15:24 761,916 sikou11yo.cpp

2022/12/23 14:19 934,973 sikou11ys.cpp
2022/12/12 15:00 482,772 sikou11yt.cpp
2022/12/20 06:29 975,137 sikou11yy.cpp
2022/12/17 10:31 209,591 sikou11yz.cpp
2022/11/14 16:08 820,291 sikou11zb.cpp
2022/11/04 17:29 825,709 sikou11zg.cpp
2022/12/18 15:54 245,653 sikou11zi.cpp
2022/12/19 10:39 906,704 sikou11zj.cpp
2022/12/11 17:24 960,073 sikou11zk.cpp
2022/11/22 14:19 844,176 sikou11zl.cpp
2022/12/20 11:04 784,330 sikou11zo.cpp
2022/12/22 16:58 921,671 sikou11zs.cpp
2022/12/09 11:00 656,648 sikou11zt.cpp
2022/12/13 11:32 964,108 sikou11zz.cpp
2022/12/25 10:21 0 sikou2.txt
2009/05/07 10:11 21,856 sjun.cpp
2012/03/12 12:49 21,865 skaku_jun.cpp
2009/04/20 07:28 17,324 skei_jun.cpp
2015/10/21 08:16 1,804,350 skin_jun.cpp
2012/04/01 17:00 19,920 skyo_jun.cpp
2010/09/14 09:17 2,054 sobtotal.cpp
2010/10/04 16:02 40,020 Soctotal.cpp
2022/10/25 05:56 112,335 Sode.h
2021/01/30 06:31 387 Sode_g.cpp
2021/01/30 06:31 395 Sode_s.cpp
2010/09/04 16:31 43,605 softotal.cpp
2010/04/20 13:23 524 sou1total.cpp
2016/11/25 06:11 328 sou2total.cpp
2012/01/08 11:28 874 sou3total.cpp

2011/05/29 09:19 1,203 sou4total.cpp
2011/03/15 17:09 1,265 sou5total.cpp
2010/09/14 09:01 1,054 sou6total.cpp
2010/09/14 09:13 988 sou7total.cpp
2010/09/15 07:15 6,803 Soutotal.cpp
2012/07/15 09:13 190,757 speed_sort.cpp
2011/01/15 10:28 24,711 Sqctotal.cpp
2010/10/04 17:24 18,676 sqctotal2.cpp
2011/01/15 10:28 26,364 sq_total.cpp
2010/10/04 17:19 1,688 sq_total2.cpp
2021/10/28 17:39 28,846 Srctotal.cpp
2010/04/05 08:51 696 ssort.cpp
2007/02/26 16:50 205 stdafx.cpp
2013/08/12 05:59 1,336 stdafx.h
2010/09/09 08:29 15,841 stntotal.cpp
2010/09/14 08:36 609 su1total.cpp
2010/09/14 09:29 1,787 suctotal.cpp
2011/10/08 17:50 23,284 sxctotal.cpp
2010/04/20 06:06 24,753 sztotal.cpp
2009/12/10 08:19 3,940 s_kiki.cpp
2021/06/01 06:15 16,598 S_total.cpp
2013/08/12 05:59 371 targetver.h
2022/06/04 09:30 1,744 te_valuen.cpp
2015/06/10 05:56 1,527 torui.cpp
2010/09/08 16:55 1,571 toruj.cpp
2016/07/15 13:11 2,664 toruj10.cpp
2016/10/27 11:35 2,590 toruj10f.cpp
2010/09/09 07:23 2,469 toruj10u.cpp
2010/09/09 07:33 1,593 toruj2.cpp

```
2010/09/20  07:24           1,855 toruj2n.cpp
2010/09/09  07:38           1,928 toruj3.cpp
2010/09/09  07:25           2,137 toruj4.cpp
2010/09/08  17:00           1,996 toruj4f.cpp
2010/09/08  17:08           1,948 toruki.cpp
2017/12/09  15:41           6,165 Total.cpp
2010/09/07  09:46           6,563 tumi_h.cpp
2010/09/07  09:50           2,616 tumi_h5.cpp
2011/02/20  08:04           1,125 tumi_k.cpp
2021/10/28  10:14           3,480 tumi_r.cpp
2009/06/02  07:02           4,764 uchifu.cpp
2010/09/09  09:07           2,524 uchifu_g.cpp
2010/09/09  13:25           2,523 uchifu_s.cpp
2011/12/15  07:41           5,031 uke.cpp
2010/09/07  09:38           4,718 uke_h.cpp
2020/07/21  05:52           4,998 uke_r.cpp
2015/05/09  08:34          13,719 ura.cpp
2010/09/15  17:09             628 wrong_y.cpp
2022/11/12  09:54    <DIR>        x64
2022/08/20  17:31          24,509 xana.h
2022/05/19  05:47          87,132 Xhineri.h
2014/03/27  06:31             406 Xhineri_g.cpp
2014/03/27  06:31             407 Xhineri_s.cpp
2020/10/12  08:45          12,728 xinaniwa.h
2013/11/06  05:13             190 xinaniwa_g.cpp
2013/11/06  05:13             189 xinaniwa_s.cpp
2022/08/20  17:13         181,331 Xmukai.h
2018/05/26  06:29             392 Xmukai_g.cpp
2018/05/26  06:29             393 Xmukai_s.cpp
```

2022/12/22 06:14 278,761 Xnaka.h
2012/11/28 14:00 344 Xnaka_g.cpp
2012/11/28 14:00 345 Xnaka_s.cpp
2022/10/16 06:01 216,700 Xsanken.h
2013/11/18 11:18 398 Xsanken_g.cpp
2013/11/18 11:18 401 Xsanken_s.cpp
2022/10/29 08:41 541,713 Xsiken.h
2007/08/16 06:13 405 Xsiken_g.cpp
2007/08/16 06:13 408 Xsiken_s.cpp
2022/12/02 09:23 532,221 Yagura.h
2022/10/05 10:31 427,181 yagura1.h
2007/02/26 16:50 365 yagura1_g.cpp
2007/02/26 16:50 368 yagura1_s.cpp
2007/02/26 16:50 526 Yagura_g.cpp
2007/02/26 16:50 525 Yagura_s.cpp
2022/12/13 09:34 877,672 yb1.h
2022/12/13 09:25 933,434 yb2.h
2022/12/13 09:25 770,048 yb3.h
2022/12/09 06:09 920,417 yc1.h
2022/06/06 15:30 880,852 yc2.h
2022/06/06 15:30 888,716 yc3.h
2022/06/06 15:30 890,149 yc4.h
2022/12/05 08:49 693,831 yc5.h
2022/10/29 16:15 913,470 yj1.h
2022/12/18 17:13 884,540 yj2.h
2022/10/29 16:15 857,614 yj3.h
2022/10/29 16:15 298,400 yj4.h
2022/10/29 16:15 932,125 yj5.h

2022/10/29 16:15 901,433 yj6.h
2022/10/29 16:15 857,237 yj7.h
2022/10/29 16:15 896,964 yj8.h
2022/10/29 16:15 898,980 yj9.h
2022/12/19 06:31 574,546 yja.h
2022/12/13 06:04 870,634 yk1.h
2022/12/13 06:04 130,706 yk2.h
2022/12/13 06:14 940,293 yk3.h
2022/12/13 06:04 921,693 yk4.h
2022/12/15 11:54 897,229 yk5.h
2022/12/13 06:04 893,555 yk6.h
2022/12/13 06:04 96,276 yk7.h
2022/12/13 06:04 897,428 yk8.h
2022/12/13 06:04 884,118 yk9.h
2022/12/23 06:28 657,291 yka.h
2022/12/23 15:24 930,903 yo1.h
2022/12/23 15:05 87,062 yo2.h
2022/12/23 15:05 953,849 yo3.h
2022/12/23 15:05 946,023 yo4.h
2022/12/23 15:05 927,030 yo5.h
2022/12/23 15:24 983,371 yo6.h
2022/12/23 10:33 117,235 ys0.h
2022/12/23 10:33 925,962 ys1.h
2022/12/23 10:33 74,302 ys2.h
2022/12/23 10:52 903,196 ys3.h
2022/12/23 10:33 888,195 ys4.h
2022/12/23 10:33 134,016 ys5.h
2022/12/23 10:33 891,415 ys6.h
2022/12/23 10:33 882,666 ys7.h

2022/12/23 10:33 887,835 ys8.h
2022/12/23 10:52 799,346 ys9.h
2022/12/16 06:20 327,304 Ysenpo.h
2010/09/15 17:26 621 ysenpo_g.cpp
2021/12/23 09:49 3,126 ysenpo_o.h
2010/09/16 08:40 405 ysenpo_og.cpp
2010/09/16 08:37 406 ysenpo_os.cpp
2010/09/15 17:10 622 ysenpo_s.cpp
2022/12/12 15:00 1,001,308 yt1.h
2022/12/12 14:46 928,487 yt2.h
2022/12/12 15:00 498,629 yt3.h
2021/10/27 20:17 57,169 YThreadProc.cpp
2021/06/03 16:33 3,120 yusen.cpp
2010/09/15 16:57 620 yusen_y.cpp
2022/11/29 11:32 994,372 yy0.h
2022/12/13 09:55 918,997 yy1.h
2022/11/29 11:32 946,059 yy2.h
2022/11/29 11:32 960,612 yy3.h
2022/12/13 16:21 760,868 yy4.h
2022/11/29 11:44 904,509 yy5.h
2022/11/29 11:32 903,470 yy6.h
2022/11/29 11:32 907,153 yy7.h
2022/11/29 11:32 803,027 yy8.h
2022/11/29 11:32 973,595 yy9.h
2022/11/29 11:32 941,281 yya.h
2022/12/13 16:12 940,881 yyb.h
2022/12/14 06:28 922,059 yyc.h
2022/11/29 11:32 27,528 yyd.h
2022/12/13 15:35 938,914 yye.h

2022/11/29 11:32 883,779 yyf.h
2022/11/29 11:32 903,159 yyg.h
2022/12/13 13:12 926,964 yyh.h
2022/11/29 11:32 921,038 yyi.h
2022/12/13 15:41 907,071 yyj.h
2022/11/29 11:32 96,723 yyk.h
2022/11/29 12:20 849,746 yyl.h
2022/11/29 11:32 883,497 yym.h
2022/11/29 11:32 877,313 yyn.h
2022/12/02 06:33 929,275 yyo.h
2022/12/14 06:28 596,539 yyp.h
2022/12/17 10:58 868,229 yz1.h
2022/12/17 10:31 77,826 yz2.h
2022/12/17 10:58 814,744 yz3.h
2009/03/13 07:28 274 y_zahyo.cpp
2022/11/14 16:08 906,354 zb1.h
2022/11/14 15:59 884,403 zb2.h
2022/11/14 15:59 886,978 zb3.h
2022/11/14 16:20 350,130 zb4.h
2022/11/04 17:29 907,067 zg1.h
2022/11/04 17:29 860,415 zg2.h
2022/11/04 17:29 324,126 zg3.h
2022/11/04 17:29 899,786 zg4.h
2022/11/04 17:29 294,937 zg5.h
2022/10/17 11:26 909,400 zi1.h
2022/10/17 11:26 810,010 zi2.h
2022/10/17 11:28 500,838 zi3.h
2022/10/31 10:09 929,832 zj1.h
2022/11/21 15:22 918,095 zj2.h

2022/12/03 09:31 886,756 zj3.h

2022/12/03 15:17 343,320 zj4.h

2022/10/30 09:39 917,651 zj5.h

2022/10/30 09:39 929,919 zj6.h

2022/10/30 09:39 325,754 zj7.h

2022/10/30 09:39 868,684 zj8.h

2022/10/31 09:34 873,695 zj9.h

2022/10/30 09:39 909,414 zja.h

2022/10/30 09:39 896,899 zjb.h

2022/12/19 11:46 892,135 zjc.h

2022/10/30 09:39 890,391 zjd.h

2022/12/19 12:33 903,088 zje.h

2022/11/26 06:17 898,239 zk1.h

2022/12/10 10:47 831,391 zk2.h

2022/11/26 06:17 877,663 zk3.h

2022/12/11 09:44 206,818 zk4.h

2022/11/26 06:17 895,817 zk5.h

2022/11/26 06:17 847,091 zk6.h

2022/11/26 06:17 860,515 zk7.h

2022/12/10 10:38 892,766 zk8.h

2022/11/26 06:17 897,991 zk9.h

2022/12/11 17:24 231,039 zka.h

2022/11/22 14:19 834,468 zl1.h

2022/11/22 14:19 724,594 zl2.h

2022/11/22 14:19 694,781 zl3.h

2022/12/20 11:04 952,506 zo1.h

2022/12/20 10:36 920,314 zo2.h

2022/12/20 10:36 926,506 zo3.h

2022/12/20 11:04 681,497 zo4.h
2022/12/22 16:58 266,236 zs0.h
2022/12/22 16:58 963,939 zs1.h
2022/12/22 16:58 890,058 zs2.h
2022/12/22 16:58 890,298 zs3.h
2022/12/22 16:58 248,500 zs4.h
2022/12/22 16:58 897,830 zs5.h
2022/12/22 16:58 903,730 zs6.h
2022/12/22 16:58 194,682 zs7.h
2022/12/22 16:58 892,363 zs8.h
2022/12/22 16:58 890,980 zs9.h
2022/12/22 16:58 905,846 zsa.h
2022/12/22 16:58 409,289 zsb.h
2010/07/06 10:43 2,495 zsort.cpp
2022/12/09 10:39 958,869 zt1.h
2022/12/09 10:39 923,058 zt2.h
2022/12/09 10:39 313,341 zt3.h
2022/12/06 16:03 953,905 zz00.h
2022/12/06 14:43 943,491 zz01.h
2022/12/06 14:43 70,487 zz02.h
2022/12/06 14:43 984,920 zz03.h
2022/12/06 14:43 975,622 zz04.h
2022/12/06 14:43 605,138 zz05.h
2022/12/06 14:43 919,898 zz06.h
2022/12/06 14:43 93,989 zz07.h
2022/12/06 14:43 950,809 zz08.h
2022/12/06 14:43 912,705 zz09.h
2022/12/06 14:43 932,218 zz10.h
2022/12/06 14:43 897,672 zz11.h

2022/12/06 14:43 80,339 zz12.h
2022/12/06 14:43 967,741 zz13.h
2022/12/06 14:43 924,535 zz14.h
2022/12/06 14:43 902,741 zz15.h
2022/12/06 14:43 126,099 zz16.h
2022/12/06 14:43 963,479 zz17.h
2022/12/13 11:32 941,325 zz18.h
2022/12/06 14:43 969,846 zz19.h
2022/12/06 14:43 899,366 zz20.h
2022/12/06 14:43 927,725 zz21.h
2022/12/06 14:43 943,924 zz22.h
2022/12/06 14:43 953,528 zz23.h
2022/12/06 14:43 902,695 zz24.h
2022/12/06 14:43 958,325 zz25.h
2022/12/06 14:43 934,127 zz26.h
2022/12/06 14:43 933,085 zz27.h
2022/12/06 14:43 939,824 zz28.h
2022/12/06 14:43 918,090 zz29.h
2022/12/13 11:32 896,692 zz30.h
2022/12/06 14:43 905,025 zz31.h
2022/12/06 14:43 873,686 zz32.h
2022/12/06 14:43 910,162 zz33.h
2022/12/06 14:43 915,056 zz34.h
2022/12/06 14:43 924,553 zz35.h
2022/12/13 11:32 459,121 zz36.h
520 個のファイル 211,215,264 バイト
4 個のディレクトリ 162,793,574,400 バイトの空き領域

一部が同じタイムスタンプになってますが、これがコンピュータにコーディングさせた

証拠で人間がやるとこうなりません。
通常、１ステップ４０バイトくらいでしょうから、ステップ数がどれだけ大きいかわかられると思います。
ただ、これだけステップ数が大きいとコンパイルからリンクまで２０分くらいかかるのでおすすめできないと言うか課題です。
また、ステップ数が大きいとデバッガーとエディターが異常な動きをします。
例えば、１ファイルのステップ数が65536ステップ数を超すとデバッガーがソースの位置を正しく表示しなくなります。

思考部がどうなっているかの質問ですと、序盤は定跡データを使った同一局面検索と類似局面検索で、同一局面検索は乱数で棋譜ファイルを検索して同一局面ではその手を指して類似局面ではそれが見つからなかったら、if文のら列でyagura.hとかがその部分になります。
従って、これらのソースがどうなっているかがわかってしまうと隠岐の指し手にどこが弱点かばれてしまいますので公開等はとてもじゃないですが、できないんです。
中盤以降は、２手読みなってます。
２手読みとは、森田将棋の故・森田和郎さんが教えてくれた手法で人間の感覚に近いと判断しましたので、この手法を採用してます。
２手読みでは読みの緩い部分が出るので、それを補正したらこれだけ大きいステップになったとも言えます。
隠岐の思考は、１手１秒を目標としてますが、終盤になって持ち駒が増えますと遅くなります。

最近、Bonanzaのソースを見て思ったのですが、彼のソースは極端に少ないです。
将棋というものをほとんど教えてなくて、単に統計データを使って、指し手を補正してると判断しました。
初期は、CSA将棋を使って作られたソースなのに、座標系は違うわ、bitboardは使うわ、知能指数の高い連中というか、毛の３本多い連中は、考え方が違うんだなーと思います。
まっ、彼の手法によって、将棋という古代人が残したパズルを解明する方程式が見つかり

つつあるように思えます。

従って、最近自信喪失というか、隠岐の修正を止めて、将棋の思考を若い人にまかせて

おいた方が良いように思えます。

ただ、それによって将棋という文化が失われつつあるのを危惧してる。

あうあう将棋・アピール文書2023
・7-10手程度の全幅探索を行っています。
・反復深化を使っています。
・評価関数はシンプルで駒の損得だけです。
・末端では王がとれるかどうかを1手延長して判断しています。
・強さとは関係ありませんが、OpenGL で 3D の盤面を作っています。
・盤面はくるくると回すことができるのですが、ＰＣが古いこともあり、
　こちらに処理が集中して、差し手を受信しそこなうのが恐ろしいので、
　おそらく当日は回さないと思います。。

爆裂駒拾太郎
アピール文書

作成日：2023年3月31日

みなさん

詰将棋
やってないですよね？

大事なことなので
もう一度

詰将棋
やってないですよね？

詰将棋はすべてを破壊します

詰将棋意味ないです

爆裂駒拾太郎が
証明しましょう

目次

1. 爆裂駒拾太郎について
2. 主な使用言語・ライブラリ
3. 定跡部
4. 探索部
5. 評価関数部

爆裂駒拾太郎について

- 開発方針
  - 詰将棋が意味ないことを証明する
  - 詰みではなく基本的に入玉宣言による勝ちを目指す

- GitHub
  - URL:https://github.com/burokoron/Yolts

目次

1. 爆裂駒拾太郎について
2. 主な使用言語・ライブラリ
3. 定跡部
4. 探索部
5. 評価関数部

主な使用ライブラリ(1/2)

- Rust 1.67.1
  - 対局エンジンの作成に使用
- yasai 0.5.0 [1] ベース
  - Rustで利用できる将棋ライブラリ
  - 指し手生成、局面管理に使用
  - 使いやすいように一部改変

[1] yasaiのURL : https://github.com/sugyan/yasai

主な使用ライブラリ(2/2)

- Python 3.9.16
  - 学習データおよび評価関数の作成に使用
- cshogi 0.4.0 [2]
  - Pythonで使用できる将棋ライブラリ
  - 学習データおよび評価関数の作成における局面管理に使用

[2] cshogiのURL : https://github.com/TadaoYamaoka/cshogi

目次

1. 爆裂駒拾太郎について
2. 主な使用言語・ライブラリ
3. 定跡部
4. 探索部
5. 評価関数部

定跡部

- 学習データの生成時の副産物として作成
- 学習データの棋譜から各局面の勝ち数、負け数、引き分け数を算出し、Thompson Samplingアルゴリズムに基づいて指し手を選択する
- ただし、訪問数が10未満の局面となるような指し手は選択されない

目次

1. 爆裂駒拾太郎について
2. 主な使用言語・ライブラリ
3. 定跡部
4. 探索部
5. 評価関数部

探索部

- 探索アルゴリズム
  - αβ探索
  - 反復深化探索
  - 静止探索(駒取り、王手回避のみ最大3手延長)

- ムーブオーダリング
  - Hash Move
  - Killer Heuristic
  - MVV-LVA
  - History Heuristic (piece-to)

- 枝刈り
  - Mate Distance Pruning

目次

1. 爆裂駒拾太郎について
2. 主な使用言語・ライブラリ
3. 定跡部
4. 探索部
5. 評価関数部

評価関数部(1/2)

- 方針
  - 詰みではなく入玉を目指すような評価関数を作成する
- 特徴量
  - 3駒関係(KKP)

評価関数部(2/2)

- パラメータの調整

1. 探索部により強制的に相手を詰ますと負けになるようにし、
   入玉でなければ勝てないようにした爆裂駒拾太郎を作成する
2. 通常の爆裂駒拾太郎と対局を行い棋譜を作成する。
   棋譜をばらけさせるために1度以上出現した局面については、
   勝ち数、負け数、引き分け数を算出ておき、
   Thompson Samplingアルゴリズムに基づいて指し手を選択する
3. 棋譜のうち1で作成した爆裂駒拾太郎の手番の局面と評価値のみで
   評価関数の学習を行う
4. 3で作成した評価関数を新たな爆裂駒拾太郎とし、1へ戻る。
   ただし、2では過去の評価関数との対局も行う。
   このときの対局割合は勝率の高い評価関数を小さく、
   勝率の低い評価関数を大きくする

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

2023 年　第 33 回大会提出アピール文

重力場計算法を引き続き開発中です。
目に見える成果が出るのは次回以降となりそうです。

****
2022 年　第 32 回大会提出アピール文

盤面評価を重力場計算法を用いて計算します。

重力場計算法 PV
将棋の数学的解明に必要となる哲学と理論物理学、そして開発者に課せられる心構えについて謳った動画。

ニコニコ動画
https://www.nicovideo.jp/watch/sm40237301

YouTube
https://youtu.be/m3nkswy9X3g

****

2021 年　第 31 回大会提出アピール文

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

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

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

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

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

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

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

■「このプログラムの思考における工夫や独自性について」

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

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

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

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

こまあそび
アピール文書

2023/03/26

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。

Ari Shogi WCSC33アピール文書(仮)

兵頭優空

1.概要

Ari Shogi(以降Ari)はディープラーニングを使用した将棋AIです。

今までと違い、今回はpython-dlshogi2をベースにする予定です。

2.使用ライブラリ

・python-dlshogi2: ベース

・cshogi: 合法手生成など

・KomoringHeights: root局面での詰み探索

・水匠5: 定跡生成関連で使う予定

3.教師データ

・書籍「強い将棋ソフトの創りかた」の付録の教師データ

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

・AobaZeroの公開されている教師データ(http://www.yss-aya.com/aobazero/)

・dlshogi with GCTのWCSC31バージョンの公開されている教師データ
(https://tadaoyamaoka.hatenablog.com/entry/2021/05/06/223701)

・Ari Shogi同士の自己対局のデータ(予定)

4.評価関数

現在はResNet以外の構造を実験中で、上手くいけばそれを採用する予定です。

その他

大会終了後により詳細について書いたアピール文書に更新する予定です。

リンクなど

GitHubアカウント: https://github.com/YuaHyodo

Kaggleアカウント: https://www.kaggle.com/yuahyodo

lichessアカウント: https://lichess.org/@/UA_2419006

2023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。

３駒間評価値をプロ棋譜約2600とコンピュータ将棋選手権上位の棋譜、

強い将棋プログラムと旧なり金との対戦棋譜を使い教師あり学習させています。

臥龍 アピール文書 (第33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 :
  - 84勝133敗4分
- 開発言語 :
  - Java, Python
- ソースコード行数 :
  - Java 思考部(詰探索) 7000行、UI部 10000行
  - Python 学習部 200行、推論部200行
- 採用している手法 :
  - 評価関数 : Neural Networkモデル(いわゆるValue Network)
  - 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、df-pn
- 探索速度 :
  - ? nodes/s
- 評価関数パラメータ :
  - 入力38チャンネル
  - 約200万
- 教師データ
  - floodgate2021年棋譜

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(WCSC32)からの改良

- なし

2023/3/23記

第３３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

２０２３年０３月３１日　高橋智史

SYUSSEKI
出席　しようぜ

DARE
誰　なの？

開発者
高橋　智史　（著作権上、問題ありません）

「わたしは　出席することだけが　すべてだぜ。
何も期待するなだぜ」

コンピュータ将棋エンジン
きふわらべ　（著作権上、問題ありません）

「アピールしたい内容が尽きてるの　わらう」

ひよ子　（著作権上、問題ありません）

「なんか書かないと　ＰＲ文書の要件を　満たさないんじゃないの？」

「いつもの感じで、ランダムムーブに何か付いた程度の
れさかい（れっさーうさぴょん改）に勝てないぐらいのものだぜ。

残りの２４ページは余ってるから、　適当なこと書くか。
来月の自分が　がんばって　実装してくれるだろ」

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

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

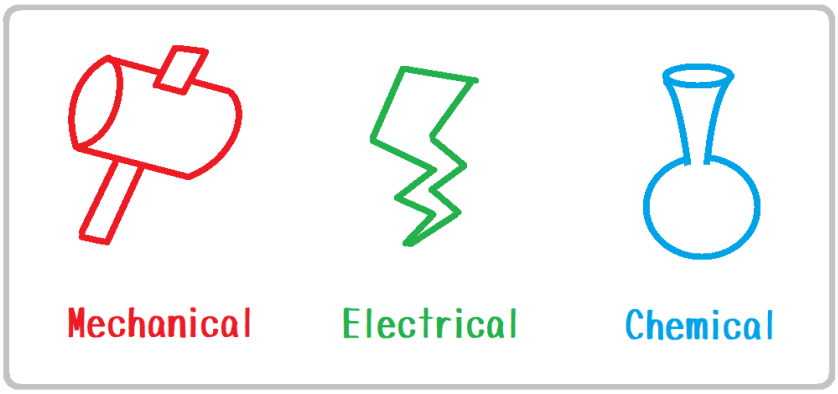
NEURAL NETWORK

ニューラル・ネットワーク　は使わないぜ

おさらい：　NOU 脳　の真似が流行ってまだ１０年ほど

参考
日本学術会議おもしろ情報館「学習と記憶　２．脳はこうして記憶する１」
https://www.scj.go.jp/omoshiro/kioku2/index.html

「早送りで　脳科学を
おさらいする。
その前に　話しを　単純化しよう！」



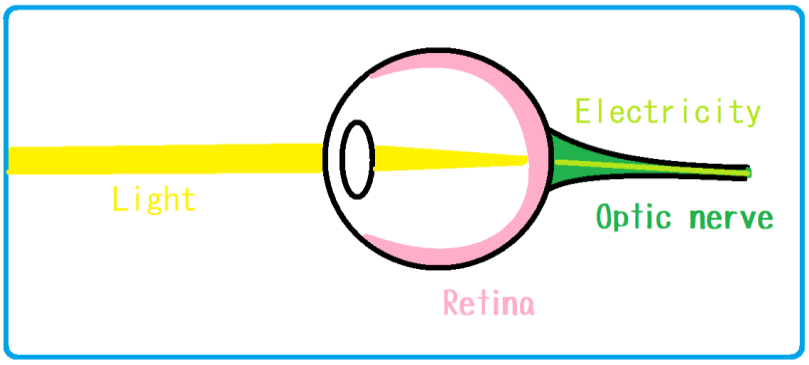
「☝　机を　ドンッ　と叩けば　揺れる、みたいなやつが
メカニカル（Mechanical；力学的）だぜ。

電気は　エレクトリカル（Electrical；電気的）、

物が材質レベルで変化したり、爆発したりするのは
ケミカル（Chemical；化学的）だぜ」

これらを　まとめて
フィジカル（Physical；物理的）とでも　呼ぶとしようぜ」

「電気の説明が　なおざり　だな……」



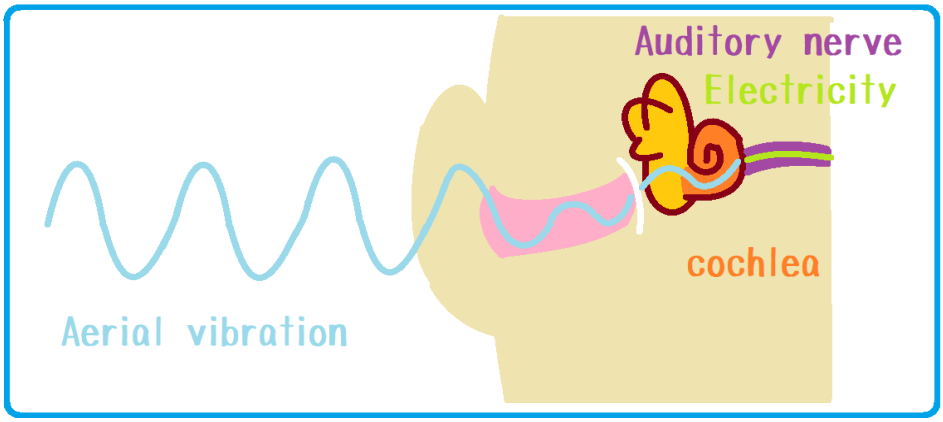
Eye

「 目では、ライト（Light；光）というエネルギーが
レチナ（Retina；網膜）に当たるまでが　メカニカル　だぜ。

レチナの奥で　脱分極　という現象が起こって
エレクトロシティー（Electricity；電気）というエネルギーに変わる。
オプティック・ナーブ（Optic nerve；視神経）では
エレクトリカル　だぜ」

「 人体も　電気で動いてるのねえ」

「 お父んに　電池を入れてみようぜ？」



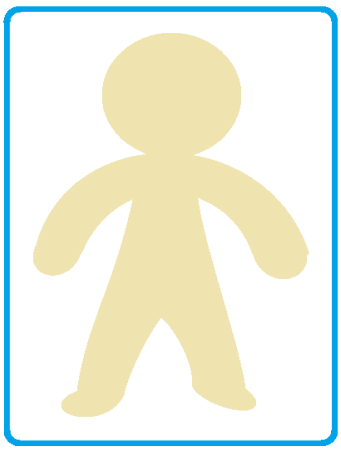
Ear

「 耳では、空気の振動というエネルギーが
コクレア（Cochlea；蝸牛）に生えてる毛に当たるまでが
メカニカル　だぜ。

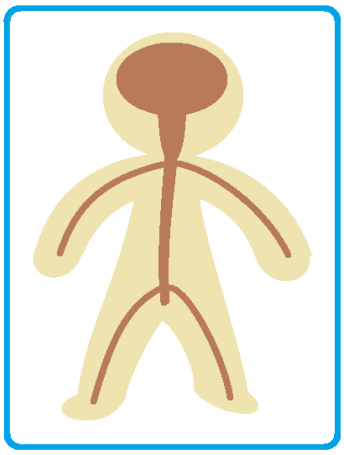
コクレアの毛で　脱分極　という現象が起こって
エレクトロシティー（Electricity；電気）というエネルギーに変わる。

その先の　オーディトリー・ナーブ（Auditory nerve；聴神経）では
エレクトリカル　だぜ」

「じゃあ　全身を
単純化しよう！
体があるとするぜ　☞」



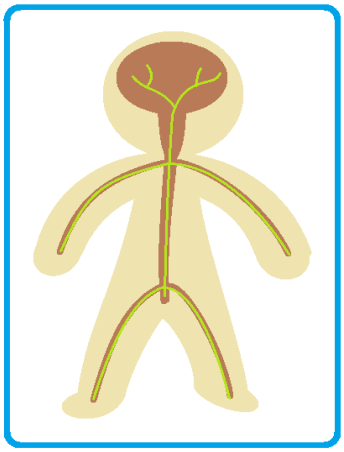
Body



Neuron

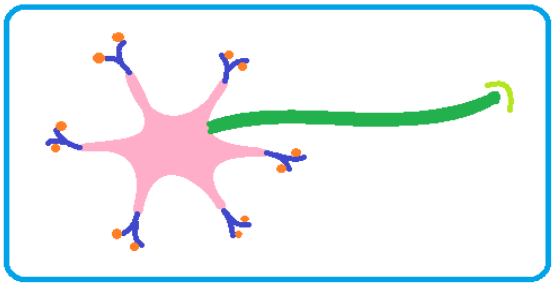
「☜　全身に栄養を送る川のようなものが通っていて、
ニューロン（Neuron；神経細胞）で　満たされている」

☞　ニューロンには　電気が
流れてるもんだと
思ってくれだぜ」



Electricity

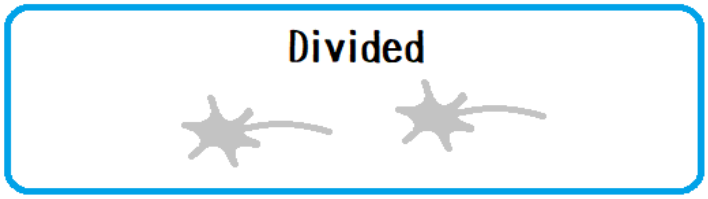
「そこまで　分かってるんだったら
アンドロイドも　誰かが作ってそうな
気がするなあ」



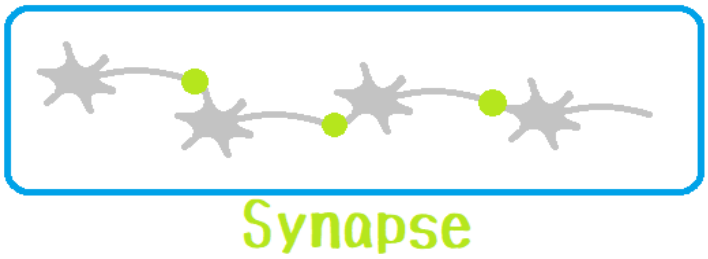
「ニューロンは　おおまかに
５つの部品で構成されている　☞」

Cell Body
- Dendrite
  - Receptor
- Axon
  - Synapse

セルボディ（Cell Body；細胞体）
デンドライト（Dendrite；樹状突起）
レセプター（Recepter；受容体）
アクソン（Axon）
シナプス（Synapse）

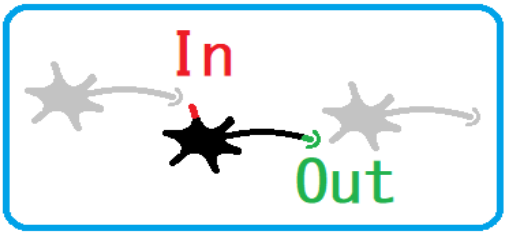


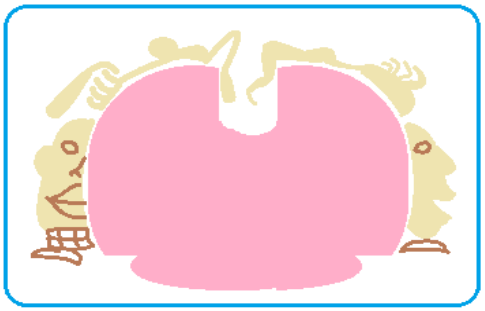
「 ☝ ニューロン と ニューロン は、 くっつかないぜ」



「 ☜ ニューロン と ニューロン の
橋渡しを シナプス が
ケミカル に やってくれるので
電気の信号が つながるぜ」

「 入り口 と 出口 も
はっきりしてるので、
ニューロンは
流れの向きを持つぜ ☞」



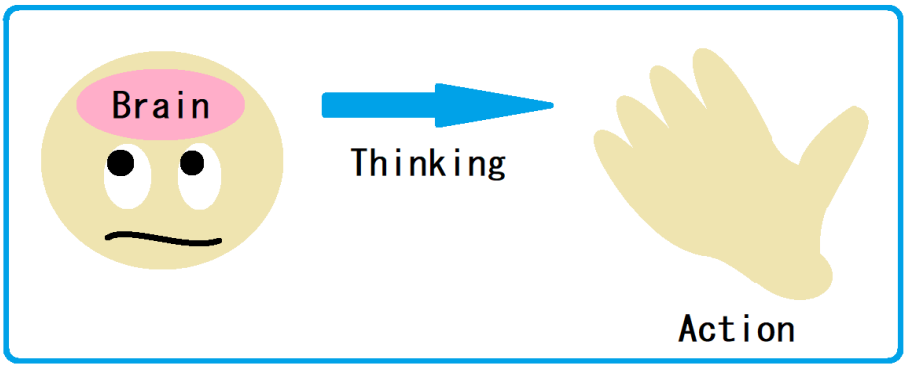


Homunculus by Penfield

「 ☝ で、ペンフィールドという人が
誰かの脳みその 表面 を つつきながら
どこが 体のどこと関係あるか 描いた図が
ペンフィールドのホムンクルスだぜ。
これが多分 １９５１年頃……」

「 将棋のホムンクルスを作ってくれだぜ」

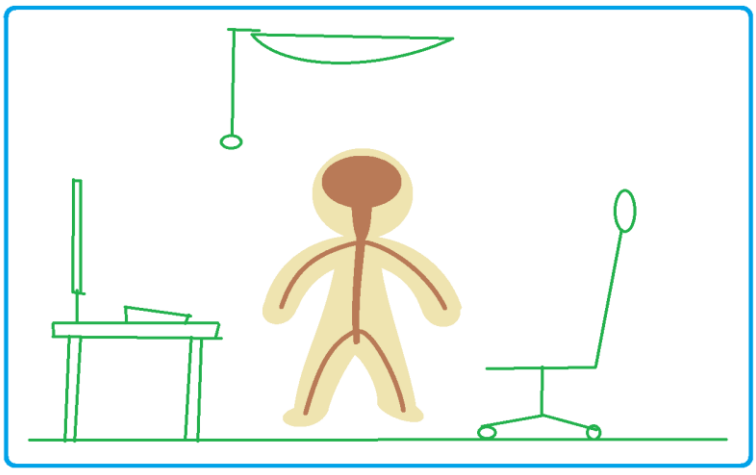
「 そこまで　分かってるんだったら　脳科学　わりといいとこまで
行ってる気がするのよねえ」



「 で、脳から全身に向かう　ニューロン　を使って、
動けと　栄養と　電気思考を　送っているそうだぜ　」

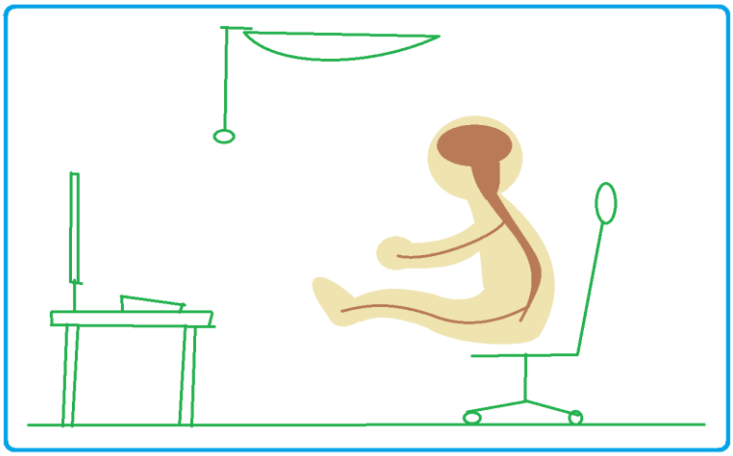
「 脳は　誰が　動かしているんだぜ？」

「 勝手に動いてんじゃないの？」



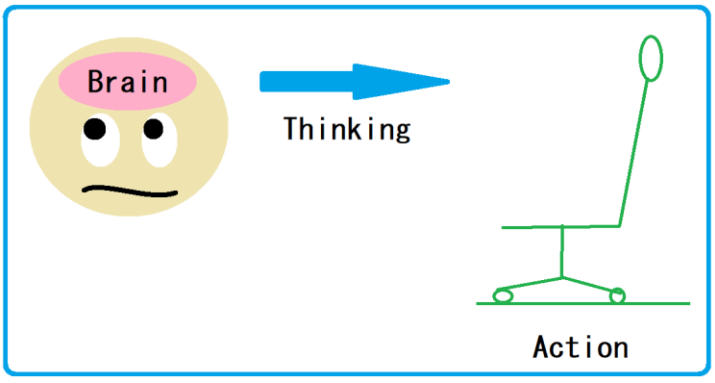
「 ☝ 周りのものを 何と呼ぶのか知らないが、
仮に エンバイロンメント（Environment；環境）とでも
呼ぶとしようぜ。
ここで、脳は 変わったことをするぜ」

「 ☞ 椅子に
座ってみようぜ」



「 いままで 直立歩行していた動物が
椅子に座って 手で 仕事を始める。
すごくないかだぜ？」

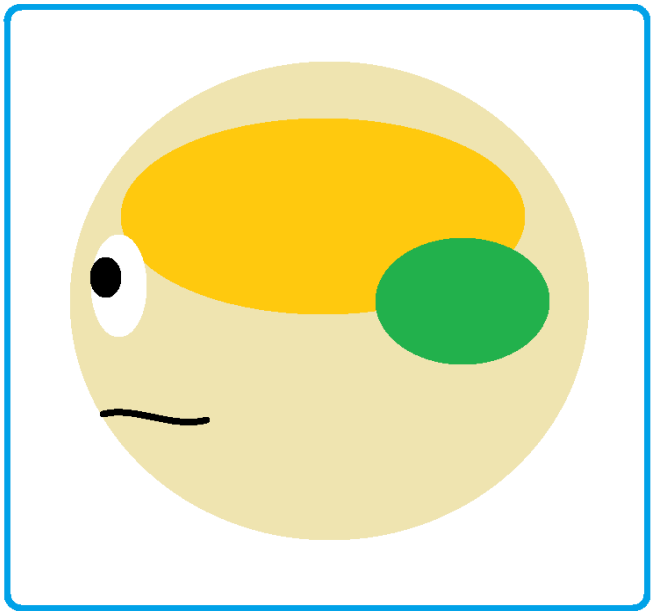
「 すごくは ないわねえ」



「 ☝ わたしたちの脳は、立ち仕事にも
椅子に座った仕事にも 対応できるんだぜ。 すごい！」

「 誰でも対応できるのに……」

「 ☞　全身の動きを、
主に　時間を正確に
タイミングを取って　動かす
働きをするのが、
セレブレム（Cerebellum；小脳）
だそうだぜ」

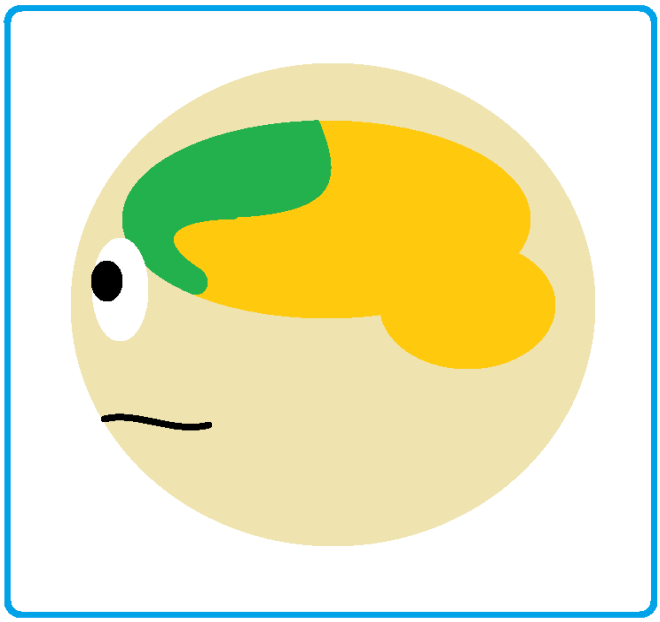


Cerebellum

「 脳って　大部分が　運動神経じゃない。
どこに　知能があんの？」

「 ☞　フロンタル・ローブ（Frontal Lobe；
前頭葉）にあるらしいぜ。

大脳半球の前側の方だぜ」



Frontal lobe

参考
脳の世界「料理は前頭連合野を使う知的な活動のひとつ」
http://web2.chubu-gu.ac.jp/web_labo/mikami/brain/42/index-42.html

「☝　例えば　ペンフィールドの姉さんは　右の前頭葉にできた脳腫瘍を
取り除く手術を受けたあと、
料理の手順を　完遂できなくなったという例があるそうだぜ。

前頭葉は、プログラミング能力に関係ありそうだな」

「でも　脳は　どこでも　ニューロン　でできてんでしょ。
なんで　部位　で働きが違うの？」

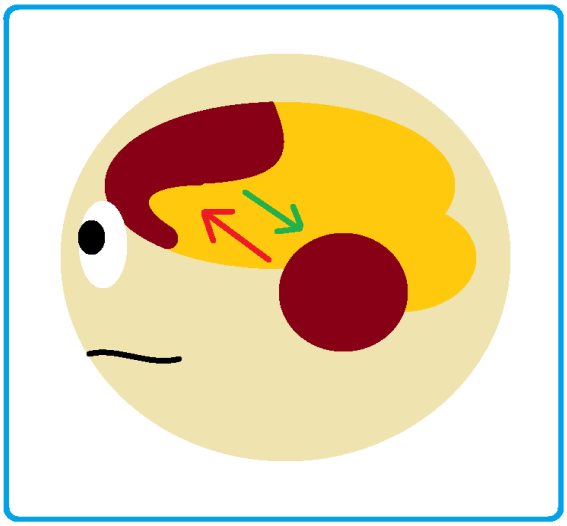
参考
東京都障害学習情報「Ⅰ　脳と心の発達メカニズム」
https://www.syougai.metro.tokyo.lg.jp/sesaku/nyuyoji-sonota/shidoshashiryo3-1.pdf

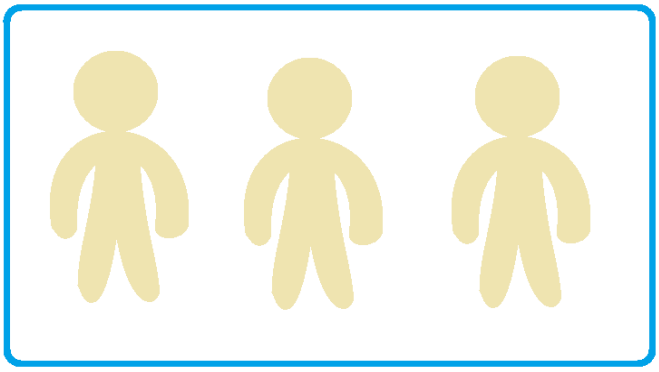
「☝👉　脳の根っこの方の
本能に近い方から
不安　が出てきて、

前頭葉の方は
思考して　大丈夫か
判定するような、

キャッチボールを
しているようだな」

「なんで　そんなことが
自然発生するんだろうな？」





Society

「 ☜ 前頭葉は
青年時代に発達
するみたいなんで

周りの人間や　教育が
関係あるんじゃないか？

知らんけど」

「 強化学習の仕組みが　あるのかしらねえ？」

「 あとは　インターネットで　全脳アーキテクチャ　を
調べろだぜ」

「 お父ん　ぜんぜん　ディープラーニングに　やる気が点火されないな。
プロンプト打つの面白くないとか　言っている
絵描きみたいになってるな」

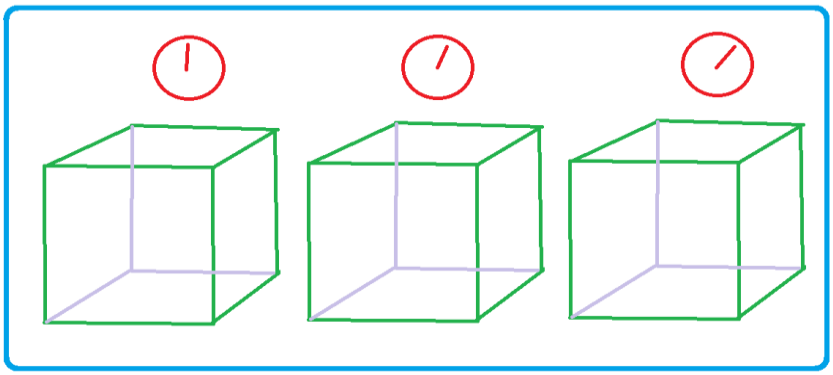
SOTOGAWA

知能の外側　の仕組み

おさらい：　なんでも　かんでも　量子（RYOSI）　がある　時空（JIKU）

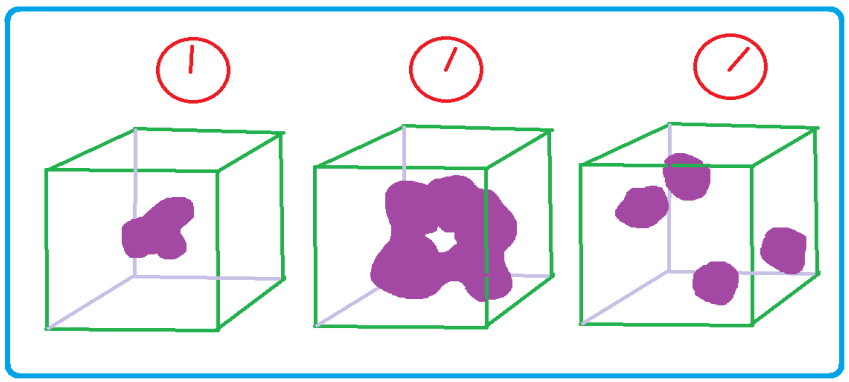
「 知能　以外のすべてのものも　調べてみようぜ？」

「 勉強熱心だな」



Space-time

「 ☝ まず、　3次元空間と　時間は　あることにしようぜ。
大雑把に　時空　とでも呼ぼう」



Quantum

「 ☝ 時空には　量子　が入っていることにしようぜ」

「 ☝ ここで ミクロな（Micro；小さな）世界と、
マクロな（Macro；大きな）世界は
人間から見ると だいぶ 違うものだとしようぜ」

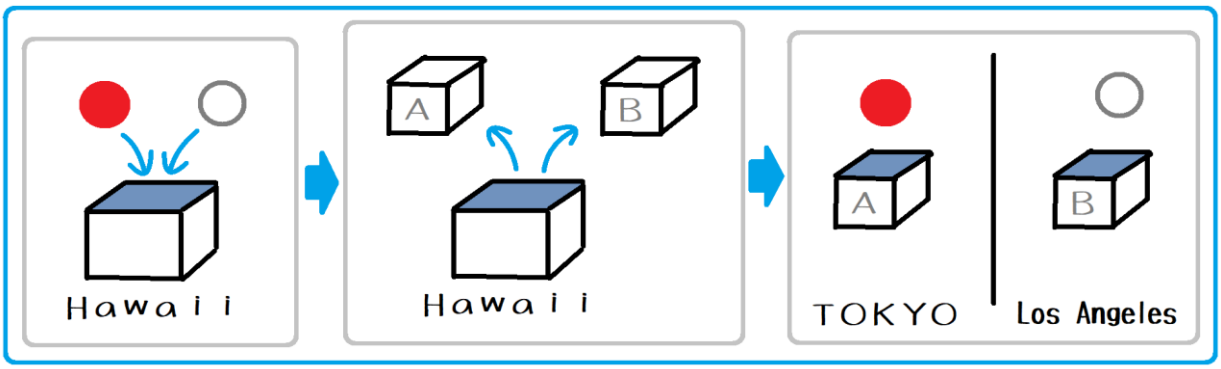
「 マクロってのは 脳 が感じている世界ねえ」

参考
総産研マガジン『２０２２ノーベル物理学賞「量子もつれ」とは』
https://www.aist.go.jp/aist_j/magazine/20230308.html

参考
東京新聞『ノーベル物理学賞「量子もつれ」とは アインシュタインが「不気味」
奇妙な性質がもたらす未来
https://www.tokyo-np.co.jp/article/209719

「 ☝ ミクロな世界にあって、マクロな世界では説明できないものがある。
それが
クォンタム・エンタングルメント（量子もつれ；Quantum entanglement）
だぜ」

「 解説してくれだぜ」



Quantum entanglement

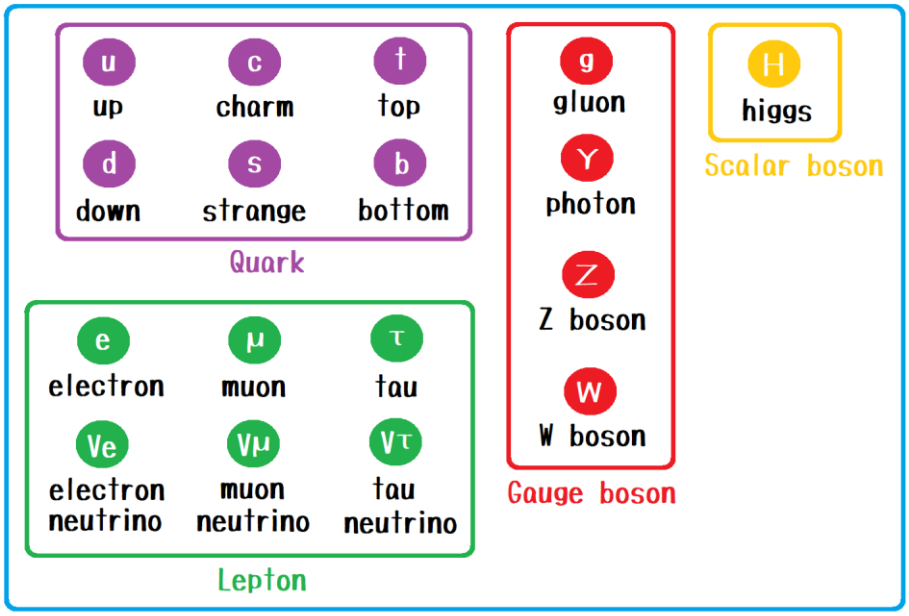
「 ☝ ① ハワイで 赤い玉と 白い玉 を箱に入れるぜ。
② Aの箱と、Bの箱、どっちに どっちの玉が入ってるか教えないぜ。
③ 東京で箱の中身を観たら 赤い玉 だったぜ。
じゃあ ロサンゼルスで 箱の中身を観たら 白い玉だな」

「 当たり前の話しに聞こえる……」

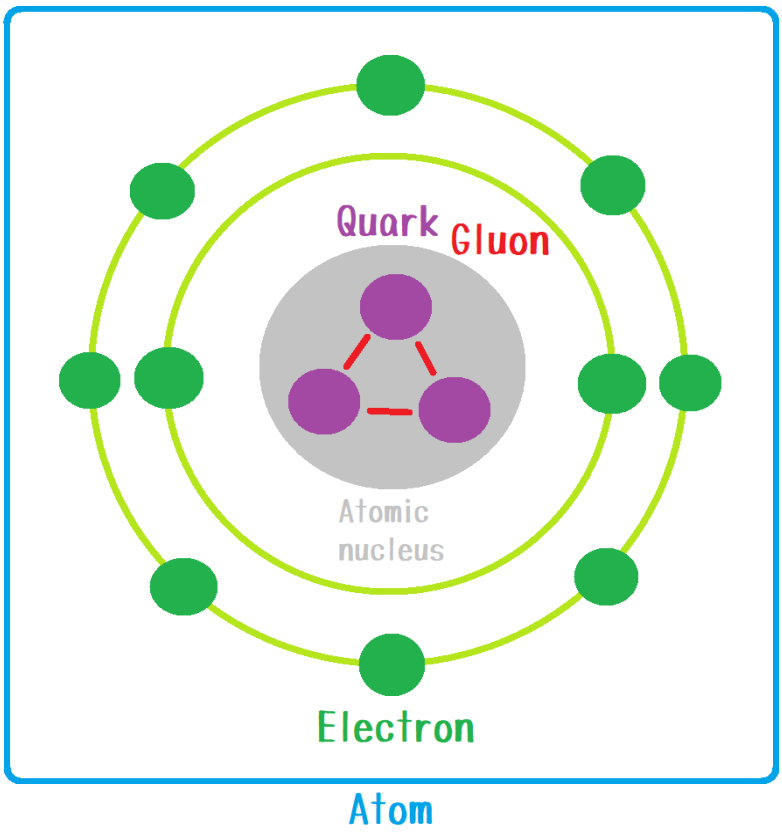
参考
量子将棋
https://shogitter.com/rule/108

「 ☝ 説明聞くより 量子将棋 を遊んだ方が早いな」

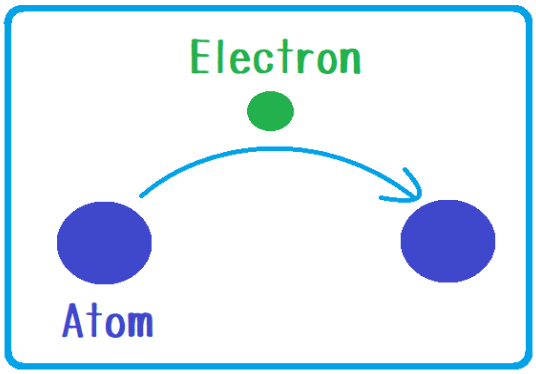
「 ☞ 量子には いろんな
種類があって、
その中の
エレクトロン（Electron；
電子）が、
電気だぜ」



Standard model of Elementary particles



「 ☜　これ、原子の大雑把な図だぜ」



「 ☞　居心地の悪いところにいる電子が、居心地のいいところへ
引っ越ししているのを
エレクトリック・カレント（Electric current；電流）とか、
電気と呼んでいるな」

「 ニューロンが　電気信号を　運ぶ仕組みねえ」

「 お父んも　原子で　できているのかだぜ？」

MICRO　　MACRO

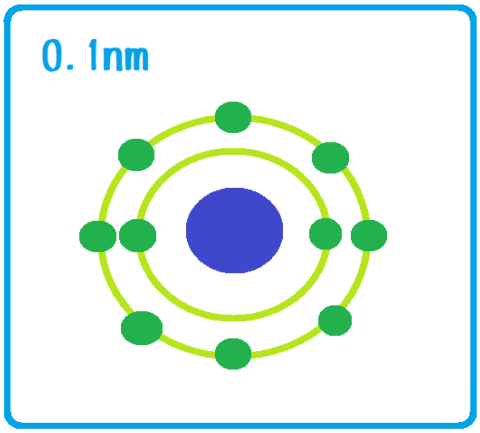
ミクロ　から　マクロ　へ

OOKISA

おさらい：　大きさ　だけで世界が違う

「☞　原子　めっちゃ　小さいぜ」

「電子顕微鏡って
よく　こんなの　見えるなぁ」

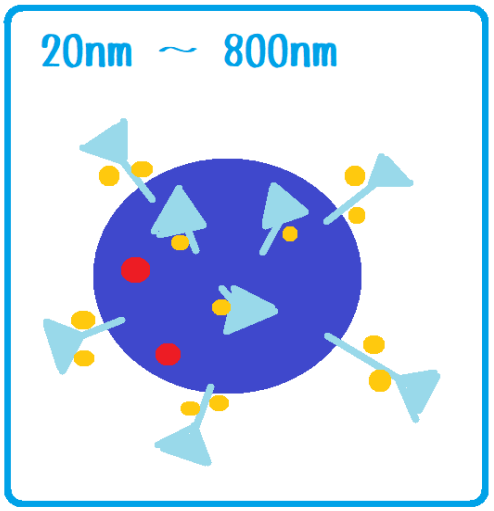


Atom

「☞　ウイルス（Virus）は
代謝を行わないから、
生物とは言われないそうだが、
まあ、定義では
そうなっているというだけだぜ」

「自己複製するそうだぜ。
なんでそんなことを
しようと思ったんだろな？」

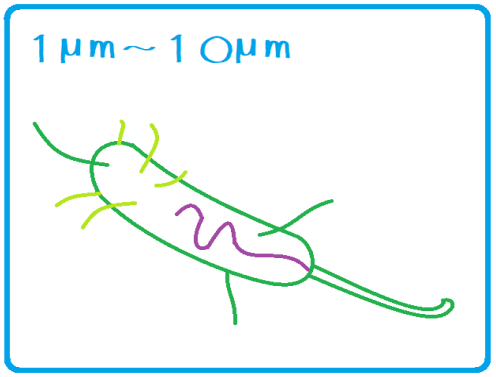
「シンギュラリティなのかなあ？」



Virus

「☞　バクテリア（Bacteria；細菌）は
生物の定義に入れてるやつで
小さいぜ」

「毛が生えてるよな」

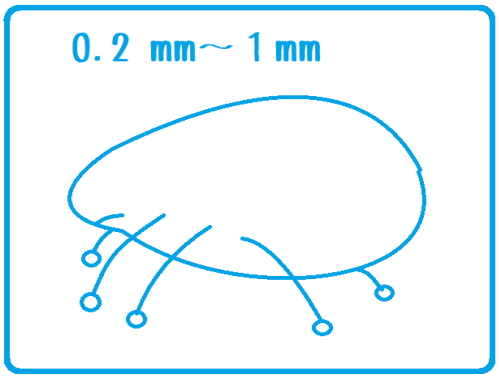


Bacteria

「☞　ティック（Tick；ダニ）だぜ。
顕微鏡で見えるので　観察を
趣味にしている人もいるぜ」

「足が生えてるよな」

「目は無いのねえ。
有っても　光を感知するぐらいなのね」

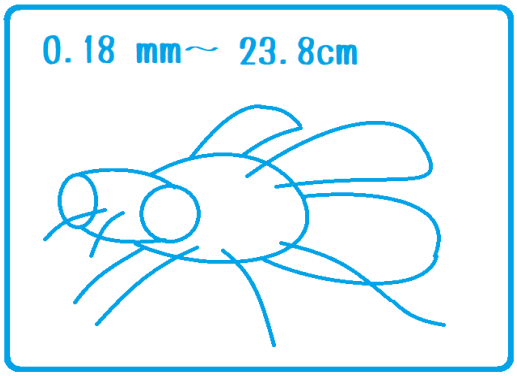


Tick

「☞　インセクト（Insect；昆虫）は
神様が好きな生物とも言われ
一番種類が多いぜ。
イラストは昆虫を代表して
ショウジョウバエだぜ」

「単眼、複眼を持ってるやつが
出てくるよな」

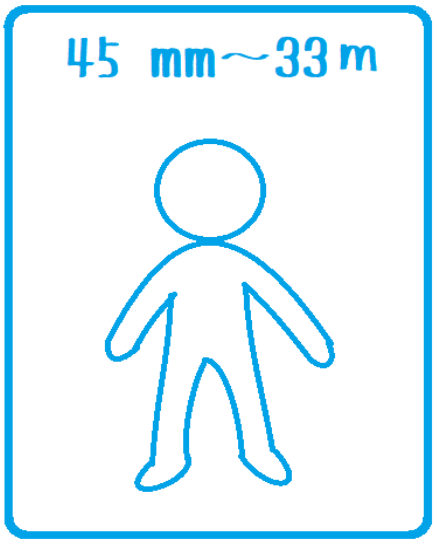
「ハチは　数を数えているという
話しもあるし、
知能なのか　習性なのかは
出てくるのかしらね？」



Insect

「☞　マーマリアン（Mammalian；哺乳類）
を代表して　ニンゲン　だぜ」

「脳を持ってるよな」

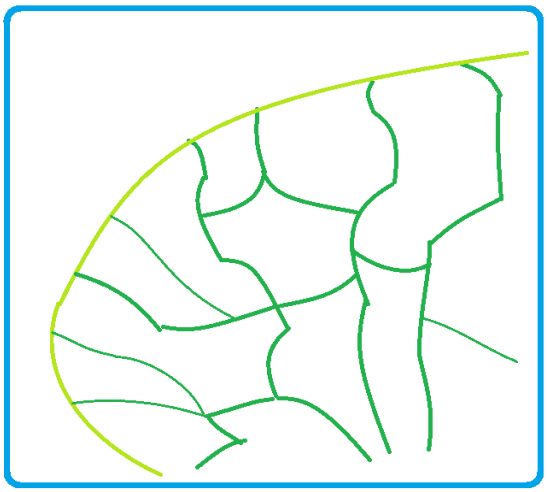


Mammalian

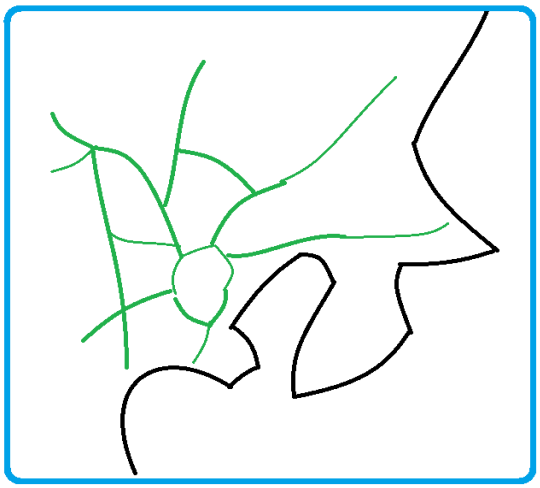
「じゃあ　もっとサイズが大きくなったら
脳　を超える器官が　出てくるんじゃない？」

「脳に　こだわらなくて　いいんじゃないか」

「☟　都市構造と　真正粘菌の物質輸送は　似てると思ってる人もいるよな」



Slime mold



Tokyo

参考
科学技術振興機構報　第708号「粘菌の輸送ネットワークから都市構造の設計理論を構築」
https://www.jst.go.jp/pr/info/info708/index.html

「知能を　持って無さそうな　菌　が
ニンゲンより効率のいい　ネットワーク　作ってるんだったら
知能って　何なのよ」

「問題設定が　たまたま　自然に前例があったということなのかな」

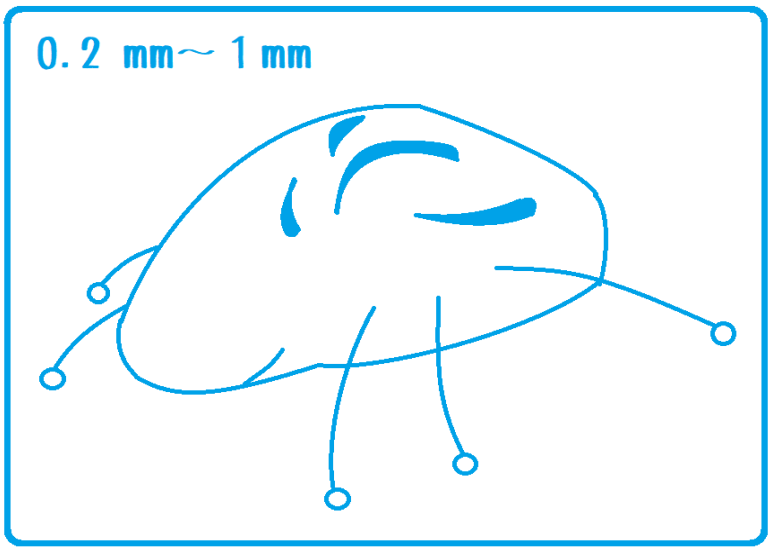
「 将棋に勝つ粘菌は　いないのかだぜ？」

「 粘菌に　こだわらなくても　いいのでは？」

「 ニンゲンの持ってる　前頭葉　が汎用的すぎない？」

「 前頭葉の　自画自賛　だぜ」

「 何にしろ　この　時空と量子の世界には
前頭葉が　自然発生　したんだぜ。
お父ん、
逆算して
将棋に勝つ粘菌を　発生　させる自然を具えた世界の原理を考えてくれだぜ」



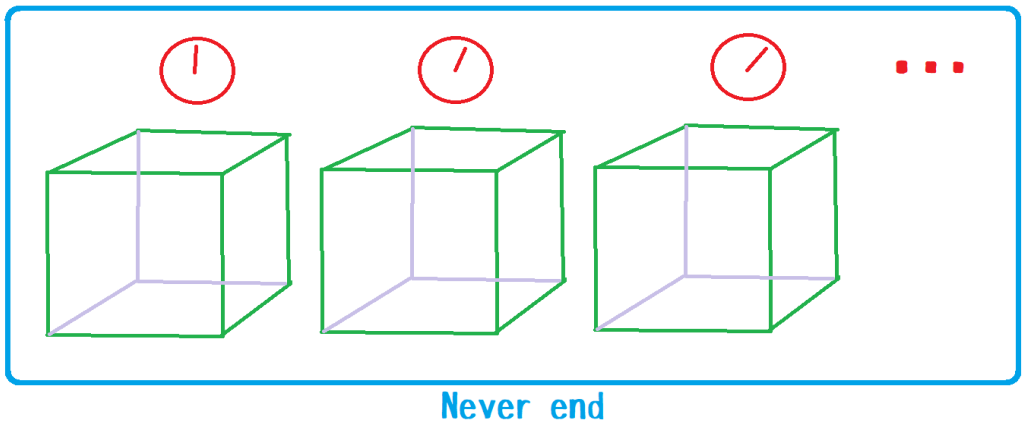
Shogi living things

SHOGI SYORI NENKN HASSEI SIZEN SEKAI
GENRI GYAKUSAN KOSO

将棋　勝利粘菌発生　自然世界　原理　逆算構想

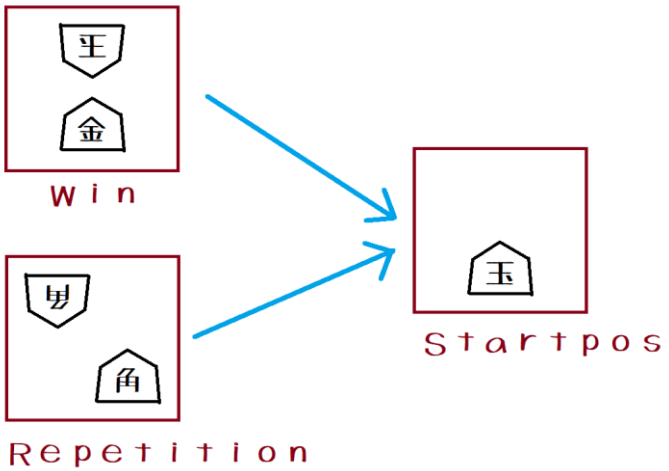
JIKUU RYOSI

おさらい：　時空と量子がある



「 ☝ じゃあ　持ち時間、および　手数に上限は無いものとしてくれだぜ」

「 ニンゲンの時間に　そんな余裕は無いのよね～」



「 ☜ じゃあ、相手の投了と、
千日手の、

次の局面を
平手初期局面につないで
くれだぜ」

「 連続対局してれば　自然と　そうなるんじゃないの？」

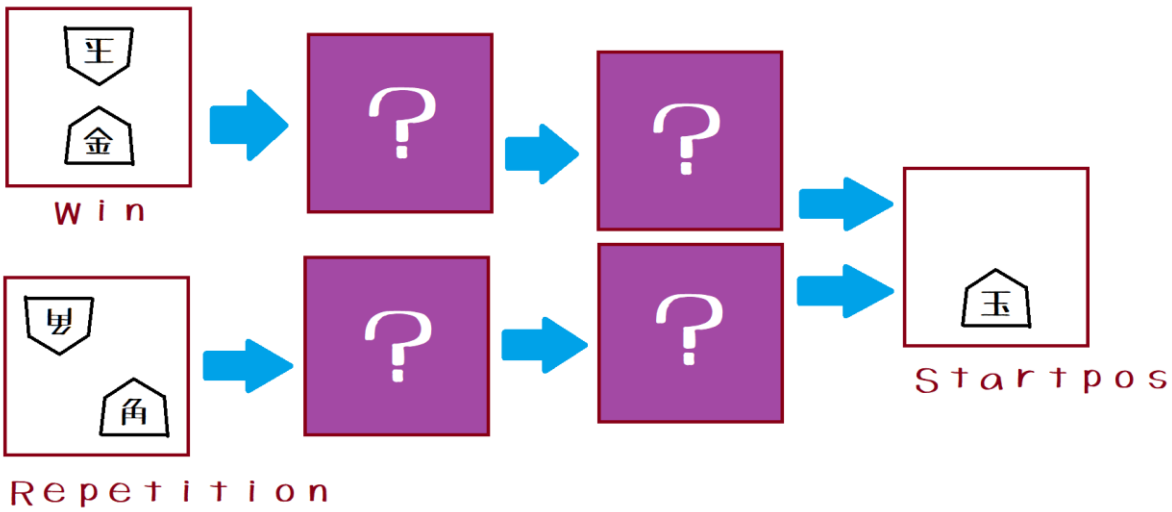
「 生物が　認知する世界として　『一局』　とか、
勝ちとか、負けとか、　そういう概念を廃止して
永遠に　次の一手を指している　としてくれだぜ」

「 将棋指しを　生物にするんだな」

「 将棋指しは　生物よ！」

「 これで　ひとまず　強いとは何かを　定義したな」

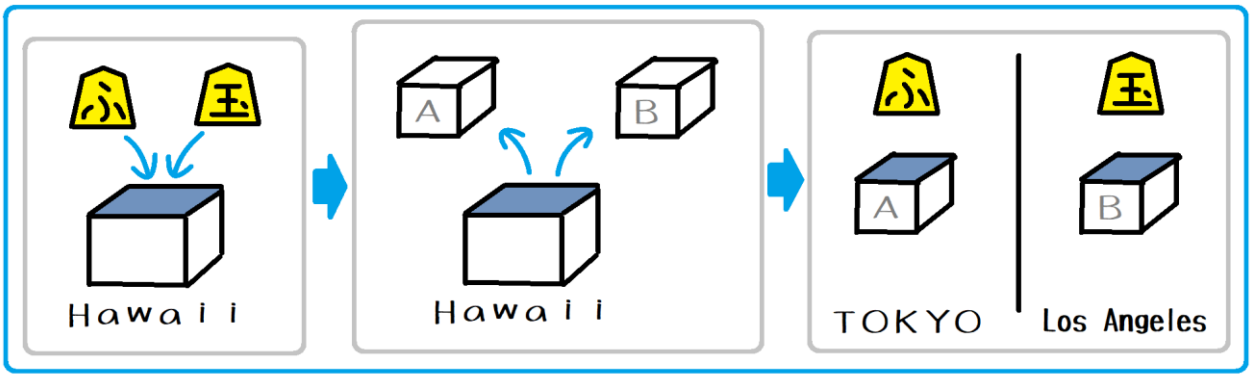
「 しかし　勝利局面や　持将棋の局面が　平手初期局面につながってるの、
突飛だよな」



「 ☝　そこが　連続してくれていたら　美しいよな」

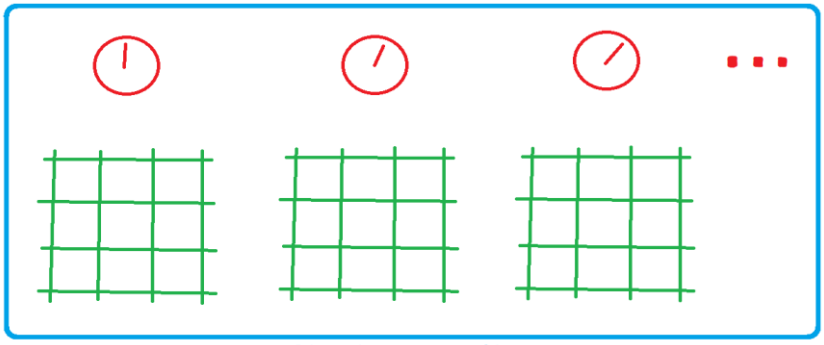
「 みんな　駒を　がしゃーってやって　１枚ずつ　置き直すわよ」

「 要らんところに　こだわるのが　お父んなんで」



Shogi quantum entanglement

「 ☝　空間か　時間か　量子の　どれかを変えないと
この世界と　同じになっちゃうんじゃない？」



Shogi space-time

「 ☝　空間が　セル（Cellular；細胞）なの　だいぶ違うだろ」

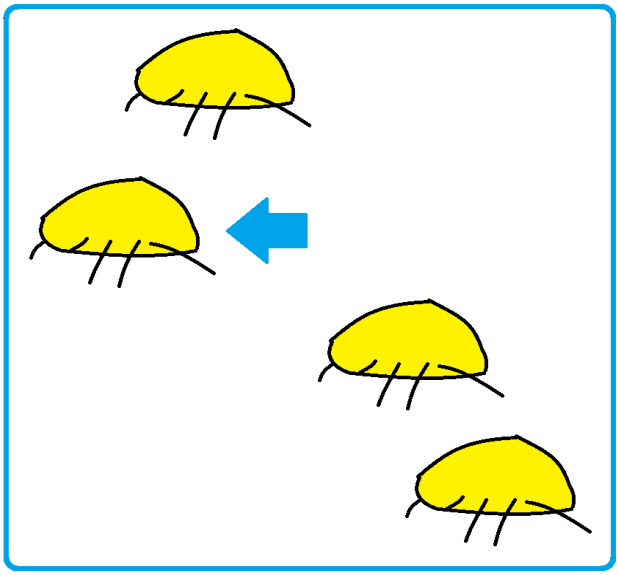
「 ☞　なんで
将棋リビングシングスは
同時に　２匹は
動かないんだぜ？」

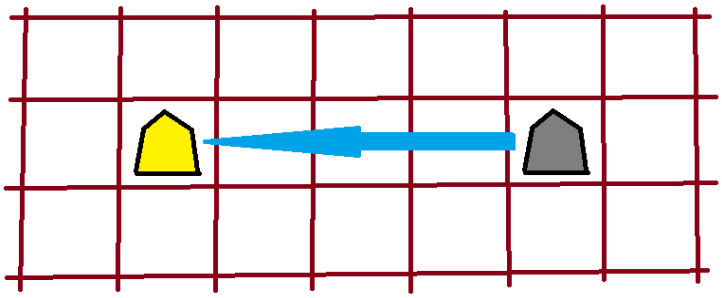
「 超・居心地のいい　安定した
世界なんじゃないか？」

「 どこが　不安定になるかが
分かれば、　最強の将棋指し
に近づくわよ？」

「 ☞　美濃囲いにしたいから
飛車を振ってくれだぜ」

「 前頭葉がないと
そんなこと　しなさそう。
自然な動きじゃ　ないよな」





TEIRE
自然を　手入れ　する装置が必要だぜ

TINOU
おさらい：　自然のままではないところに 知能 がある

「 前頭葉って　何があるから　そんなこと　できんの？」

「 前頭葉があるニンゲンや　イルカや　チンパンジーは　将来を考えることが
できるという説もあるしな。　将来を考える力　とか関係あるんじゃないか？」

「 将来を考えられる生き物と、
そうでない生き物の　違いって何だぜ？」

環境を　自分に合うように　手入れすることが
できるか　できないか　なんじゃないか？」

「 野球選手が言う　修正力　みたいなものかしらね」

「 ハードウェアの　眼球や　内耳に　修正力は無いよな。
ソフトウェアの　神経細胞と　シナプスが　修正力の実装方法かだぜ？」

「 レインフォースメント・ラーニング（Reinforcement learning；強化学習）
みたいな話しだな。
ニューラル・ネットワークを使わない　レインフォースメント・ラーニングなんか
聞いたこと無いぜ」

「 結局、 GPU の行列演算処理能力をフルに使う 並列処理 を
自力実装できる人でなければ、
ニューラル・ネットワークを使うことに行きつくのでは？」

「 そこに手を入れられないと　おもんないよな」

参考

Golly is an open source, cross-platform application for exploring Conway's Game of Life and many other types of cellular automata.

https://golly.sourceforge.net/

「 ☝ Golly というソフトが　ライフゲーム界隈で著名なんだが、
使い方分からないしな。
セル・オートマトンでなんか　できないのかな」

「 独自の手法をやるなら
バックプロパゲーション（Back Propagation；誤差逆伝播学習法）も
セットで　自分で考えださないといけないわよ」

SASITE SEISEI

そして　指し手生成　　へ

GAKUSYUBU

おさらい：　学習部　を作るの　まだ無理

「いじれるところといえば　指し手生成部　ぐらいだよな」

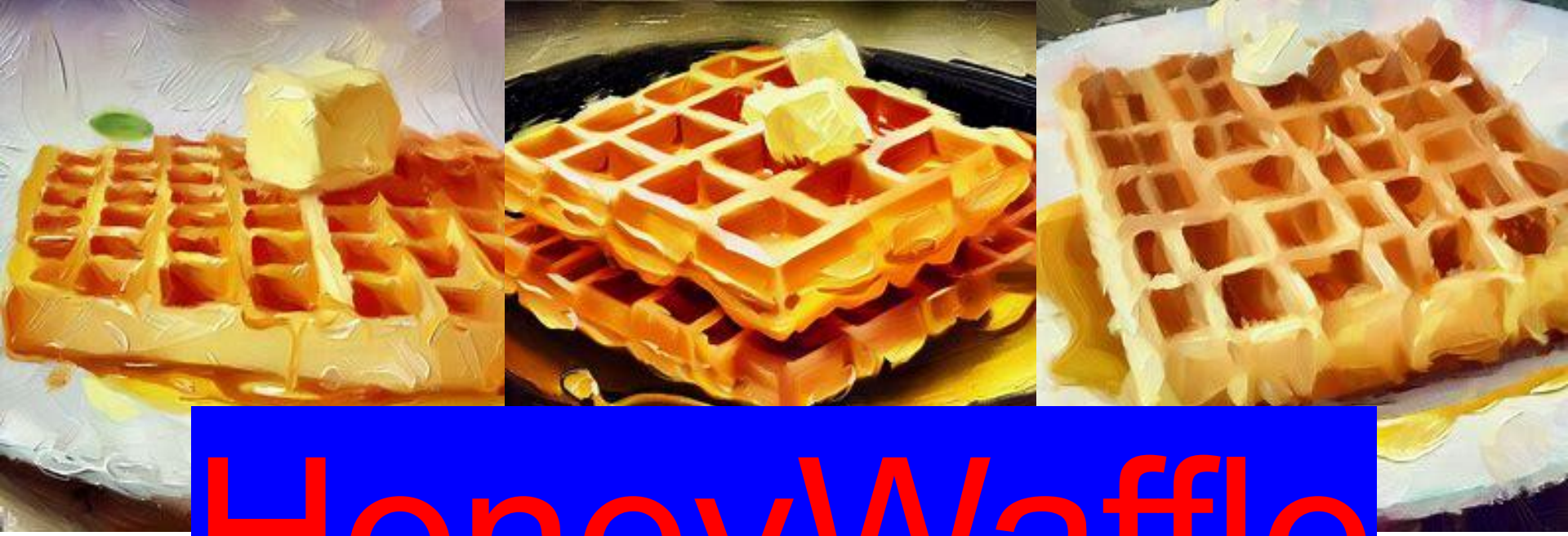
「お父んは　強化学習なんかやらず
古典的な　アルゴリズム組んだ方が
いいのかも知らん」

「大量のパラメーターを調整するような、
機械学習系から　離れた方がいいのよ。
やらないんだから」

「将棋の方を　数学的なモデルに　分解する方が
できないかな」

「れさかい（れっさーうさぴょん改）に勝ってくれだぜ」

>>> Dad, please make shogi players living things!!

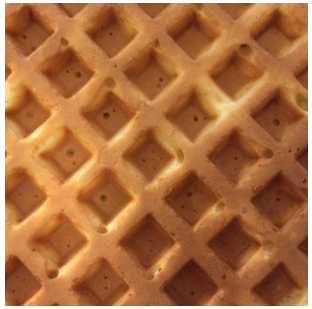


HoneyWaffle

第33回世界コンピュータ将棋選手権　アピール文書

開発者　渡辺 光彦

開発者



氏名: 渡辺 光彦

職業: プログラマー

棋力: 将棋ウォーズで2級、ぴよ将棋でR900-1000程度の振り飛車党

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

ニコ生の電王戦をきっかけにコンピュータ将棋を始める。

将棋連盟Liveやニコニコ生放送、AbemaTVの将棋中継が好き。

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

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

HoneyWaffle (ハニーワッフル) 名前の由来

・四角いワッフルは将棋盤と似ている

・ゆるふわスイーツ的なスナック感覚の軽さを表現

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

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

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

https://github.com/32hiko

コンセプト

青字は使用予定ライブラリ

「やっぱり振り飛車しかない」

（1）振り飛車定跡

定跡といえば水匠電竜の先手角換わり定跡が有名ですが、振り飛車でもそういうのができたらいいですよね？これまでに作成してきたものとは別に、新しく作成し直し中です。NoBookとの対戦では振り飛車の勝率は意外と高いです。

（2）DL系の振り飛車党評価関数（dlshogiとふかうら王）

第3回電竜戦のために、dlshogiの評価関数に追加で対抗形の棋譜を学習させたものを流用します。

（3）市販のゲーミングPCクラスのマシン

Core i5 11400（6C12T）、RTX 3080 Ti、メモリ32GB　…GPUだけ盛った変な仕様。Core i5に負けたらどうします？

最後に

　第3回電竜戦に出場した、qhapaqさんのJust Stop 26歩、定跡なしで振り飛車を指すのを実現していてすごかったです。優勝の水匠に後手振り飛車で勝つなど強くて、公開もされていますし、私自身が無理に実現を目指す必要はなくなったと思っています。

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

「技巧」アピール文書

2023 年 3 月 31 日

出村　洋介

1. Rust と Python で全面的な書き直し

以前の「技巧」は C++を開発言語としていましたが、将棋プログラムの複雑化に伴い、ポインタ絡みの不具合などの発見が難しくなっているといった困難を感じていました。

そこで、今回は思い切って C++版の開発を停止し、安全性が高いと評判の Rust で全面的に書き直しました。また、学習部は GPU を安定して動かせる Python の PyTorch[1]を利用しています。Rust や Python の特徴を活かして、以下のような点を工夫しています。

①危険性の高いコード（ポインタやグローバル変数など）の利用を最小化

②Rust のユニットテスト機能の活用（Rust ではユニットテストが標準装備）

③Rust と Python 間では、NumPy 形式で安全にデータを受け渡し（PyO3 を利用[2]）

2. 自動ベクトル化の採用（Apple M1/M2 への対応）

従来の「技巧」は、Intel や AMD の CPU（x86-64）向けの intrinsic 関数を使って SIMD 演算を行うことで、Bitboard などの高速化が図られていました。そのため、従来の「技巧」のままでは、最近購入した Apple 社の M1 Mac で動作しないという問題がありました。

そこで、今回の「技巧」では、intrinsic 関数の利用を原則廃止し、自動ベクトル化のアプローチを新たに採用しています。これにより、Intel や AMD の CPU に加えて、Apple の M1/M2 CPU でも動作するようになりました。

自動ベクトル化（auto-vectorization）は、コンパイラが自動的に for ループ等を SIMD 演算に置き換えて高速化を図る仕組みです。コンパイラの自動ベクトル化では、プログラマ側で CPU ごとに個別のコードを書かなくても、SIMD 演算による高速化ができる利点があります。自動ベクトル化はコンパイラの対応が進んでおり、Rust の LLVM コンパイラ[3]だけでなく、C++の GCC[4]などでも利用可能です。

3. AlphaZero と NNUE の組合せをテスト開発中

今回の「技巧」では、AlphaZero の手法[1]を参考に、新たに深層学習の導入をテスト開発中です。

ところが、AlphaZero の深層強化学習の手法は、学習に要する計算量が大きいことが知られています。実際、Silver ら[1]によると、将棋で AlphaZero の学習を 1 回行うには、TPU

---

[1] https://pytorch.org.

[2] https://github.com/PyO3/pyo3.

[3] https://llvm.org/docs/Vectorizers.html.

[4] https://gcc.gnu.org/projects/tree-ssa/vectorization.html.

（Tensor Processing Unit）5000 台を用いて 12 時間を要したとされています。単純計算では、TPU 1 台で 7 年ほどかかるという膨大な計算量です。

そこで、今回の「技巧」では、AlphaZero の学習手法を参考にしつつ、AlphaZero よりも学習に要する計算時間を短縮することを目指して開発しています。

アイデアとしては、AlphaZero の深層強化学習に、軽量・高速な NNUE (Efficiently Updatable Neural-Network-based Evaluation Functions)[2]をうまく組み合わせることで、AlphaZero よりも学習に要する時間を減らすことができるのではと考えて、試行錯誤しながら開発を進めています。

4. 使用ライブラリについて

コンピュータ将棋のライブラリは使用せずに、フルスクラッチで開発しています。

5. おわりに

「技巧」のアピール文書をご覧いただき、ありがとうございます。

2017 年に参加して以来久しぶりの参加となるため、初心に戻って開発していきたいと考えています。

参考文献

[1] D. Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, vol. 362, no. 6419, pp. 1140-1144, 2018.

[2] 那須悠. 高速に差分計算可能なニューラルネットワーク型将棋評価関数. 第 28 回世界コンピュータ将棋選手権アピール文書, 2018.

アピール文書
開発者名　有賀宏樹
書いた日　3/29
アプリ名　将スタ君

将棋 AI 制作者末吉竜介氏が、作成した「将スタ-将棋スタジオ-」を使って将棋ソフトを作りました。
定跡機能を追加する予定です。
超序盤「やねうら王（水匠）を使う」　python-dlshogi2（定跡抜けてから使う）　↑これらを制御するのは Ayane を使ったプログラム（過去の煌のプロトタイプ版）

JHBRのアピール文章
2023年

JHBRのアピール文章
JHBRの特徴は、指手選択部分にspiking neural network(SNN)を使用したことです。評価関数は、Floodgateの棋譜を使用してspiking timing dependent plastivity(STDP)ルールにより学習したものです。
それ以外の部分ではcshogiライブラリを使用しています。

第３３回世界コンピュータ将棋選手権「アストラ将棋」アピール文書

令和５年４月
恒岡 正年

1. 全体の構成
　dlshogi の探索部、学習部を利用。
2. 独自に実装した部分
　Network 構造、学習方法、model の生成の工夫。（詳細は後述）
3. 開発動機
　ディープラーニングの特に学習部に興味があり、「強い将棋ソフトの創り方」という書籍の内容をトレースする事で開発をはじめた。
　したがって、自前の model を作る事に注力している。
4. 主な開発内容
　独自構造の model の作成・学習方法
　棋譜生成：6000po 程度、及び 15 秒+F1.2 秒程度の物を中心に現在約 30 万の棋譜作成
　学習には、自己生成した棋譜以外に上記書籍の学習用データを利用
　AobaZero の棋譜も一時期利用したが、現在は未使用。
　探索パラメータの調整(手作業・optuna 利用)
5. 探索ロジックの検討(UctSearch.cpp の改造)
　標準の UCB 値の計算式は n が十分に大きい時に最良の計算式となるが、限られた試行回数では必ずしも最適な方法ではないと思う。
　オリジナルの計算式を独自の式と置き換える。
　探索パラメータの再調整後、R+36 程度の向上を確認。
6. 持ち時間の節約のための簡単な定跡を手入力で準備した。
　ベストラインでは深く、それを外れた場合は(有利になっているはずなので)浅い。
7. 今後の開発予定(大会後)
　指し手選択ロジックの追加(後述)
　思考時間制御(後述)

---

8. Network 構造

Network を次の 3 つに分ける。

(1)データ入力部 (2)ResNET 部 (3)データ出力部

1）データ入力部

3 つの情報の合流位置とデータのサイズを振って実験した。

現在はオリジナルに Conv2d()を 1 つ追加した構造にしてる。

データサイズは、9x9 以外にも 11x11、11x9 を試した。Play Out 数固定での評価では有意に強くなるが、探索速度の低下が大きく探索時間一定の条件では弱くなった。ネットワーク層数またはカーネル数を増やす方が良い。

将来的にネットワーク層数またはカーネル数を増やしても改善効果が小さくなった場合に、データサイズ 11x11 を試してみたい。

2）ResNET 部

オリジナルの ResNET に限らず何通りかの形状(Conv2D()3 層の ResNET や入れ子構造の ResNET 他)のネットワークを試したが、最終的にオリジナルと同じ Conv2d() 2 層の ResNET にした。

データサイズは 9x9。(11x11、11x9 も試したが不採用)

カーネルサイズは 3x3 のみ評価。

データ入力側と出力側でカーネル数が異なる構造を採用。

ResNET 部の層数は 25。

3）データ出力部(Policy/Value Network の分岐後)

オリジナルの構造に Conv2D()を 1 つ追加している。

ResNET を何層か追加する実験も行ったが優位性は認められなかった。

4）その他

活性化関数は 主に ReLU を、最も出力に近い Conv2D()にのみ SiLU を使っている。

計算量は入力部と出力部に Conv2D()を追加しているため ResNET27 層相当。

9. 学習方法

1）今回の使用モデルでは、学習率(lr)：0.2 から始めて 0.000000012 まで等比数列的に減少させ学習した。減少率は主に 0.85。最後の数エポックは

学習率を固定値とした。
100 エッポックの学習で 2 週間～3 週間かかる。
　2）バッチサイスは 512 から初めて GPU メモリの許す限り順次大きくしている。
　3）マクロバッチを実装し、最終的に 12 倍まで学習単位を大きくしている。8 倍～12 倍に最適値が有りそう。
(FP16 だとこれより大きくすると精度が足りない。(学習率) x (バッチサイズ) x (マクロバッチ倍率) が一定値を超えると収束付近で学習効果が無くなる様に思う。)
　4）小さめの学習セットで 1 エポックの学習を数回実行し、最も D 値 (=Loss-(PolicyAcc+ValueAcc)*0.5)が小さくなった物を採用し以降の学習を行っている。初期値の分布の素性の良さそうな物を選ぶ。
　5）25 エポックまでに 3 回、入力に近い部分の ResNET を入れ替えて学習する。これにより収束を早める効果を狙っている。
　6）26 エポック以降も D 値をモニターし、D 値の減少が期待値以下の場合に、意図的にイレギュラーなデータを与える。データの与え方は 2 種類の手法を用意した。
　特定の次元が局所解に陥っている可能性を考慮しそこから脱出するのを期待している。
　7） D 値をモニターしていると収束に近づいているかどうかがわかるので D 値の挙動によって終了するエポックを決める。今回使用するモデルの場合約 100 エポックまで学習した。何エポックまで学習するかはモデルに依存する。

10. model の生成の工夫
　学習済のモデルは、最も Loss 値が低い、または最も Accuracy が高い物が勝率が高いモデルとは限らない。これは、Loss 値や Accuracy 値は出現頻度の高い特徴を持つ局面での正解率に強く依存し、出現頻度の低い特徴を持つ局面での正解率の寄与割合が低いためと考えた。将棋の勝敗は出現頻度の低い特徴の局面で正しい手を指せるかにも依存する。
　すべてのモデルでベンチマークを取るのを回避しつつ、この問題の影響を低減する工夫を行った。

11. 定跡

　手入力で定跡を用意した。思考時間の節約を目的としており、最長でも30 手程度と思う。

今回のモデルを用いて思考時間一手１分～５分で思考させる。２手目４通りの開始局面からベストラインを長く登録している。

ベストラインからの枝は短く評価値が(先手に)振れたら打ち切っている。

一本道の局面では打ち切らず、同程度の評価値の候補手が複数現れたら打ち切る。

-以上-