チーム Barrel house のアピール文書@第 29 回世界コンピュータ将棋選手権

Barrel house とは岡山の駅前にあるビアバーです．昨年チームメイトを求めさすらって行きついたところです．マスターに許可頂いたので名前をお借りしました．その後，メンバーが変わって当初の方向性とは全く違って来ましたが，まぁ一度出した名前を変更するのもアレなのでそのままです．

プログラム名：Kristallweizen
昨年の Hefeweizen が濁った白ビールでしたが，Kristallweizen はフィルターでろ過した透き通った白ビールです．命名経緯は上記のビアバーに起因します．
命名時は昨年より洗練されたソフトにしたいとの気持ちでしたが，準備ペースは昨年同様で洗練とは程遠い感じです．
ちなみに，マスターによると Kristallweizen は樽で輸入するのが難しいので瓶になりますが大丈夫ですかとのことでした．

チームの特徴．
昨年は段取りが分からず全関係者を加えておりましたが開発者に絞りました．今年からのメンバーもお願いして参加頂きました．若いメンバーも大口スポンサーもありませんので，多方面に精通した開発者の独創的なアイデアだけが勝負どころと考えています．昨年は独創的な評価関数生成と先行先読み Multi Ponder が冴え渡り優勝致しましたが，今年は別のブルーオーシャンへ漕ぎ出す予定です．

CSA 使用可能ライブラリ使用表明．
メンバーの意思統一が図れておりませんので多めに入れておきます．また，試行錯誤ツールで色々使わせて頂いております．
Apery，やねうら王，tanuki-，Qhapaq，elmo，技巧，dlshogi，python-shogi，人造棋士 18 号

ライブラリ選定理由
python-shogi：python を用いた棋譜および局面の管理，sfen 文字列の展開など
dlshogi：ディープラーニング型の評価関数試行
Apery：評価関数作成のための教師データ作成に利用。
やねうら王：探索部が高速なため主に探索部の利用。定跡部作成時にも利用。
tanuki-：互換性があり強いため評価関数の作成及び仮想敵として勝率計算に利用。
Qhapaq：互換性があり強いため評価関数の作成及び仮想敵として勝率計算に利用。
elmo：互換性があり強いため評価関数の作成及び仮想敵として勝率計算に利用。

技巧：評価関数の作成及び仮想敵として勝率計算に利用。探索部のオーダリングにも利用。
人造棋士 18 号：独自のルーチンを用いた評価関数の作成に利用。

使用マシン.
普通のノートパソコンに加えて，クラウドの力をお借りする予定です．ベンチマーク等も未だなので具体的なことは全く決まっていません．正直あまり予算がないのですが，前年優勝者がマシンパワーで負けるのもアレなので一応ハイエンドインスタンスを借りようと考えています.

遺伝的アルゴリズムによる評価関数開発について
昨年はやねうら王が探索部の速さをもって有用と判断しましたが，NNUE 等の新型の評価関数が加わり新しい可能性が拓かれています．実際 KPP_KKPT や KPPT などの複数の形の評価関数が公式バイナリで配布されている上，ユーザ関数のテンプレートまで用意されています．そして共通の教師データ構造，共通のコマンドによる強化学習インフラとなっております.
昨年度の我々は複数の近親評価関数を選別する作業を行いましたが，事後これは遺伝的アルゴリズムに類するものであることに気づきました．そして，今回は積極的に同様の手法を用いる予定にしています．具体的には Hefeweizen による教師データで tanuki-評価関数に追加学習を施すような手順です．同手順を「たぬきにビールを呑ませる」と略します．同様にたぬきやカッパをビールに漬け込むことも可能ですし，少し改造すれば Apery も参加できますし，そしてそれら次世代も「交配」可能です.
また，突然変異体として新しいアイデアを盛り込みます．評価関数の形が違うものということになります．同世代で参加するために類似の評価関数からの転移学習から入ることが多くなります．意外かもしれませんがここが今年の目玉です.

クラスタリングについて
昨年 Multi Ponder のクラスタリングを行いましたが，さらに進化したものを用意しています．今年は Go 言語で再実装した上，少々追加機能を加える予定になっています．少なくとも遅延は減るということです.

ということですので，昨年チャンピオンは昨年より強くなってくる予定にしています.
昨年同様「予定」が多いのは可能性を探る段階で決定事項が少ないためです．申し訳ありません.

第29回世界コンピュータ将棋選手権

PAL
アピール文書

山口 祐

PALの概要

- pal/ﾊﾟﾙ/【名】<話> 仲間、友だち（例）a pen ―
- 前回大会時の工夫
  - **敵対的学習**
  - クラウドサーバを用いた並列学習
  - 96スレッド環境での探索部の最適化

開発者

- 山口 祐（@ymg_aq）
- 囲碁・将棋プログラム等を開発

PALの特徴

- Deep Learningを**使わない**（再）
- 評価/探索/定跡を同時に学習する **Trinity Learning**
- ~~KPPT型評価関数で開発中~~ **NNUE型に変更**

使用ライブラリ

- やねうら王:　データ構造・学習部
- Apery:　評価関数の学習先の一つ
- 技巧:　データ構造・思考部の一部を参考に
- tanuki-:　NNUE評価関数の学習

「名人コブラ」アピール文書

概要

名人コブラはやねうら王をベースに、評価関数と定跡に手を加えたコンピュータ将棋エンジンです。

特徴

作者がAlphaGoの影響でMCTS（モンテカルロツリーサーチ）に興味を持ってしまったので、定跡の作成と評価関数学習用棋譜の作成にMCTSを利用する予定です。

ただし、速度が直接棋力に影響する対局時の探索部では、やねうら王の探索部を利用いたします。

定跡の作成にMCTSを利用する理由は、より効率よく探索が可能ではないかという予想からです。有力な変化がより深く探索されるため、定跡の作成に適していると思われます。

学習用の棋譜作成にMCTSを利用する理由は、局面の分布がより実践的になると予想されるからです。Aperyや、やねうら王で採られているランダムに選んだ手を、ランダムなタイミングで挿入するという方法がとても不自然に感じられるためです。

使用ライブラリ

現時点ではやねうら王のみを利用する予定ですが、別途申請済みのライブラリを使用する可能性もあります。

狸王
アピール文書

ザイオソフト コンピュータ将棋サークル
野田久順 岡部淳
鈴木崇啓 河野明男

目次

- 狸王
- 主な工夫点
  - NNUE評価関数
  - 強化学習における教師局面生成
    - 思考ノード数の固定
    - 棋譜生成器の探索パラメータの調整
  - 探索パラメーターの自動調整
  - クラスタリング
- 使用ライブラリ

狸王

野田「『〇〇〇〇〇〇〇〇タヌポン』っていう
　　参加名にしようと思うんですけど…」
岡部「・・・」
那須「・・・第6回電王トーナメントに使う
　　予定だった名前のほうが良いと思います」
野田「…あぁ、…そうしましょうか」

狸王(続)

- もし第6回電王トーナメントが
  開催されていたとしたら、
  この名前で出場する予定でした
- ディフェンディングチャンピョン(＝王者)
  という意味が込められています

NNUE評価関数

- 狸王は評価関数にNNUE評価関数を使用しています
  - ネットワーク構造にhalfkp_256x2-32-32を使用しています

強化学習における教師局面生成

- 狸王では評価関数を強化学習により強化しています
  - 「教師データが不足した環境での機械学習結果改善手法」と「elmo式学習法」を用いています
- 教師局面の生成に工夫を加えています
  - 思考ノード数の固定
  - 棋譜生成器の探索パラメータの調整

思考ノード数の固定

- これまでの教師局面生成では、
探索深さを固定して自己対局を行い、
棋譜を生成してきました
  - 探索深さを深くした場合、対局の終盤において、
思考ノード数が極端に増え、
思考時間が長くなる場合がありました
- 狸王では思考ノード数を固定し、
思考時間が長くなりすぎるのを抑制しています
  - 棋譜生成時間が予測しやすくなりました

棋譜生成器の探索パラメータの調整

- 棋譜生成器の探索パラメータを、
  自己対局のパラメータに合わせて
  最適化しています
  - 後述の手法でパラメータの自動調整を
    行っています
- これにより、より質の高い教師局面が
  得られるようになったと考えています

探索パラメーターの自動調整

- WCSC26で導入した、
  探索パラメータの自動調整を行っています
  - hyperoptで探索パラメータと
    自己対局の勝率のサンプリングを行い、
    Gaussian Processを用いて
    なるべく良さそうなパラメータを選んでいます

探索パラメーターの自動調整（続）

※WCSC27PR文書より抜粋・改変

- 探索パラメーターを入力、ベースとなる
  思考エンジンに対する勝率を出力とする関数を考えます
- 関数の値を最大化する探索パラメーターを探します
  - 大域的最適化問題にモデル化できます
- 自己対戦により得られる勝率には大きなノイズが
  含まれています

探索パラメーターの自動調整（続）

- Tree-structured Parzen Estimator（TPE）
  - 機械学習のハイパーパラメーターの最適化に用いられるアルゴリズムの1つです
  - PythonライブラリHyperoptに実装されています
  - 1探索パラメーターセットあたり48対局、数百パラメーターセット分の自己対戦の勝率をサンプリングします
  - Hyperoptの出力をそのまま使ったらダメ！
    - Hyperoptは最高の値を出力した最初の値を出力します
    - サンプルの誤差が大きい場合、本来弱いはずの探索パラメーターが最終結果として出力されてしまう場合があります

探索パラメーターの自動調整（続）

- Gaussian Process（GP）
  - ノンパラメトリックな回帰分析などに用いられています
  - sklearn等に実装されています
  - Hyperoptで得られたサンプルをGaussian Processでノンパラメトリック回帰分析します
  - 探索パラメーターに対する勝率の関数の確率分布が得られます
  - サンプルの中で関数の確率分布の平均値が最も高いものを最終的な解として出力します

探索パラメーターの自動調整(続)

- 前回からの変更点1
  - サンプリング初期の、ランダムサーチによるサンプリング回数を増やしています
    - これにより、局所最適解に陥りにくくなると考えています

探索パラメーターの自動調整(続)

- 前回からの変更点2
  - 自己対戦を思考ノード数を固定して行っています
    - 時間を固定した場合、各思考エンジンに割り当てられるCPUリソースがばらつき、勝率が50%に寄る問題があります
    - 思考ノード数を固定することにより、この問題を解決しました
    - 合わせて、同時対局数を物理CPUコア数から論理CPUコア数に増やし、サンプリング精度を上げています

クラスタリング

- WCSC27でPonanzaが導入した「eXtream Lazy Smp」を改良したものを実験中です
- 他のクラスタリング手法と比較し、レーティングが高かったものを採用予定です

使用ライブラリ

- Apery
  - tanuki-wcsc28開発時点で評価関数が最も強かったため評価関数をランダムスタートで機械学習させるための教師局面の生成に使用しました
- やねうら王
  - 独自の工夫を加えるにあたり、改造しやすく、レーティングも高いため、使用しました
- tanuki-
  - 強化学習のもとになる評価関数として使用しました

よろしくお願いします

コンピュータ将棋ソフト
「大合神クジラちゃん」

第 29 回世界コンピュータ将棋選手権　アピール文書

2019/03/31　鈴木雅博

1.　大合神クジラちゃんの紹介

マスタ／スレーブの構成を取り、マスタがスレーブを管理しながらクラスタリング探索を行います。スレーブを動かすのは主にニコニコ生放送や YouTube 視聴者のパソコンを考えています。前回の選手権ではパソコン４００台以上のクラスタで動かしました。

クラスタの基本的な手法は GPS 将棋を参考にしています。マスタが各指し手をスレーブに展開することでより読みを深くしています。また、冗長性を持たせるため同じ手を 2 つのスレーブに探索させています。

2.　クラスタ部分についての工夫

独自の工夫として、探索を探索ツリーの末端ノードだけでなくすべてのノードで行うことで、スレーブの性能差を気にすることなく安定した棋力向上を可能にしました。　過去のシステムでは末端でないノードでは探索させず、末端ノードのみに指し手を展開し、展開できなかった手はそれ以外の手を探索する「その他ノード」に探索させるようにしていました。しかしながらその手法では、ある指し手とそれ以外の指し手の探索量が大幅に違った場合、なにが最善手であるか判断するのが極めて難しいという問題がありました。具体的には、評価値が 1000 点・探索深さ 14 という手と、評価値が 900 点・探索深さ 20 という手があった場合、どちらが良い手なのか判断できないという問題がありました。

現在のシステムでは、末端以外の親ノードでも探索を行うようにしこの不安定性を少し減らすことに成功しました。例えば上記の例であれば、末端ノードの探索結果（指し手 A：評価値が 1000 点・探索深さ 14、指し手 B：評価値が 900 点・探索深さ 20）に加え、親ノードの探索結果が加わります。例えばその結果が評価値 950・探索深さ 18 で指し手が A だったとすると、探索深さが深くなった場合でも指し手 A が有力ということがわかります。

またこのシステムはビンゴマスターさんのアドバイスをもとに作られたものなので、ビンゴマスターさんに深く感謝申し上げます。

3. 末端ソフトの工夫

「tanuki-wcsc28」の評価関数を、前回大会で使った「クジラ 9」により生成した 100 億局面から学習させました。また、それだけではあまり強くならなかったため WEB 上に公開されている「illqpha3」の教師局面も追加学習させています。

4. その他

- 探索部分には YaneuraOu をそのまま使用しています。
- 定跡ファイルの生成に Apery を使いました。
- 非公開協力者として、まふさんに定跡ファイルを作って頂く予定です。

5. ライブラリの選定理由について

- tanuki-
  - 評価関数の大元として使わせて頂いています。
- YaneuraOu
  - クラスタのスレーブ部分の基礎として使用。非常に強く探索部分も優秀なため使わせて頂いています。
- Apery
  - クラスタのマスター部分の将棋ライブラリとして使用。探索は行わず合法手の確認や指し手生成に使っています。また、定跡ファイルの生成にも使わせて頂いています。

6. 過去の戦績

- 第 23 回コンピュータ将棋選手権 27 位（独創賞を獲得）
- 第 24 回コンピュータ将棋選手権 16 位
- 第 25 回コンピュータ将棋選手権 19 位
- 第 27 回コンピュータ将棋選手権 4 位
- 第 28 回コンピュータ将棋選手権 6 位

7. 完全オリジナルのＧＵＩもあります。

- コンピュータ将棋を動かすための GUI も自作しています。
- クジラちゃんを擬人化し、GUI 内に取り込みました。
- URL: http://garnet-alice.net/programs/whalewatcher/
- ニコニコミュニティ： http://com.nicovideo.jp/community/co516151

Qhapaq di Molto (QDM)
のアピール文

Ryoto Sawada, Yuki Ito, Toshihiro Shirakawa

```
while (1) {
  alert("終わりのないのが「終わり」");
}
```

Who is Qhapaq

「かぱっく」と読みます。ケチュア語で「偉大なもの」を意味する単語であり、本作が様々な巨人の肩の上に立った作品であることを示しています。

主な実績：

- 第五回将棋電王トーナメント５位入賞
- 第28回世界コンピュータ将棋選手権７位入賞
- 将棋神やねうら王に搭載
- 非公式レーティング１位（だった）

開発者：

Ryoto Sawada ..... 物理屋。謎の数式を並べて評価関数を作ってる
Yuki Ito ..... 最適化職人。大会当日にデスマする
Toshihiro Shirakawa ..... ゴルファー。変な入れ知恵をしてくれる

今季の意気込み＝勝ちたい、でも金は掛けたくない

計算資源への無課金プレイ

ご家庭用ゲームノートPCで強いソフトを作りたい

← 5年くらい前に買った4コア8スレのノートPC

やねうら王が公開している教師データ(深さ12、110億曲面)を作ろうとすると

致命的な計算力の差

資源不足がもたらす問題：学習が出来ない

将棋ソフトの強化学習にかかる計算資源

- 強化学習でよく使われる教師曲面の深さは10〜14程度
- ４コア８スレだと深さ12でも1日で500万曲面ぐらいしか作れない
- 10億局面作るとなると200日かかる → 平成が終わる

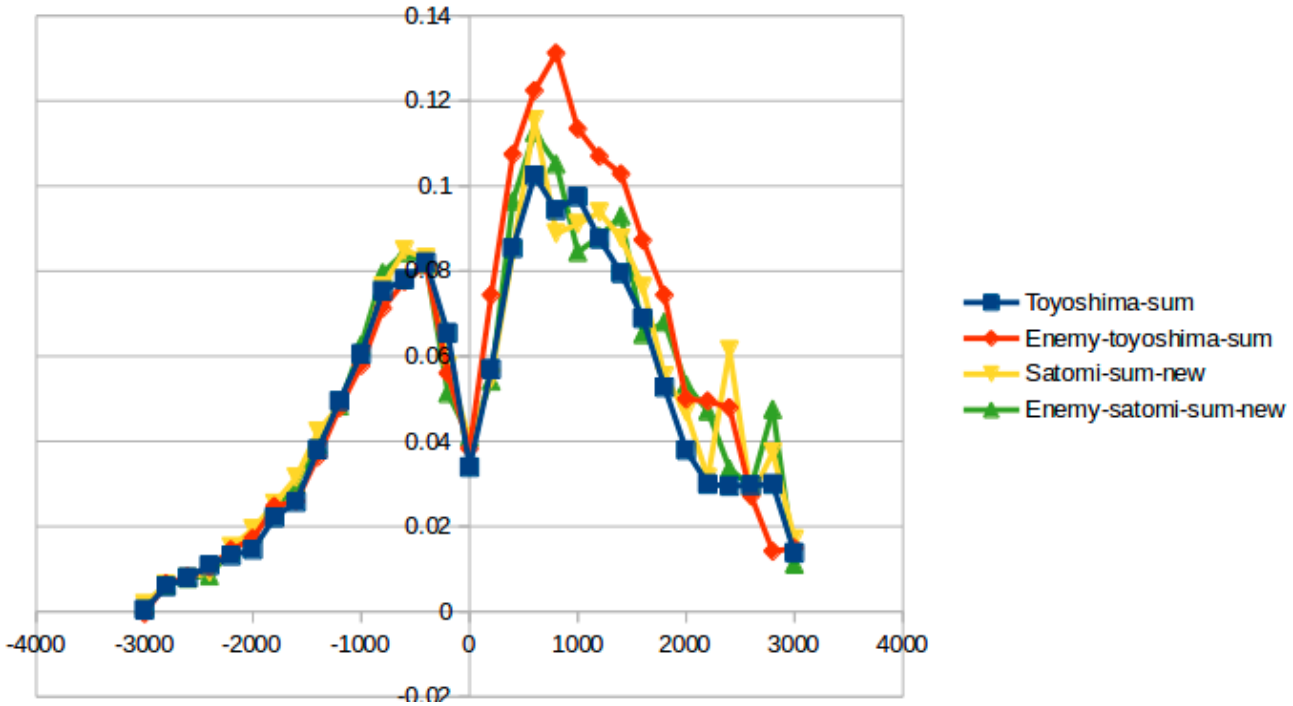
解決策 ： webサービスを開発して巨人の肩を増やす

コンピュータ将棋の纏めサイトを運営し、集合知の力に頼る

https://www.qhapaq.org/shogi/

解決策２　：　学習や対局データのより詳細な解析

- 人間やソフトの棋譜を解析して、ソフトの強さの出処をより精密に可視化する
- 通常のloss functionに加え、幾つかの指標（少なくとも大会中は秘密にする予定です）を用いることで学習や評価関数の質を見積もる



例：女流棋士のレート上昇を評価値解析から見積もってみた話

http://qhapaq.hatenablog.com/entry/2018/09/22/144405

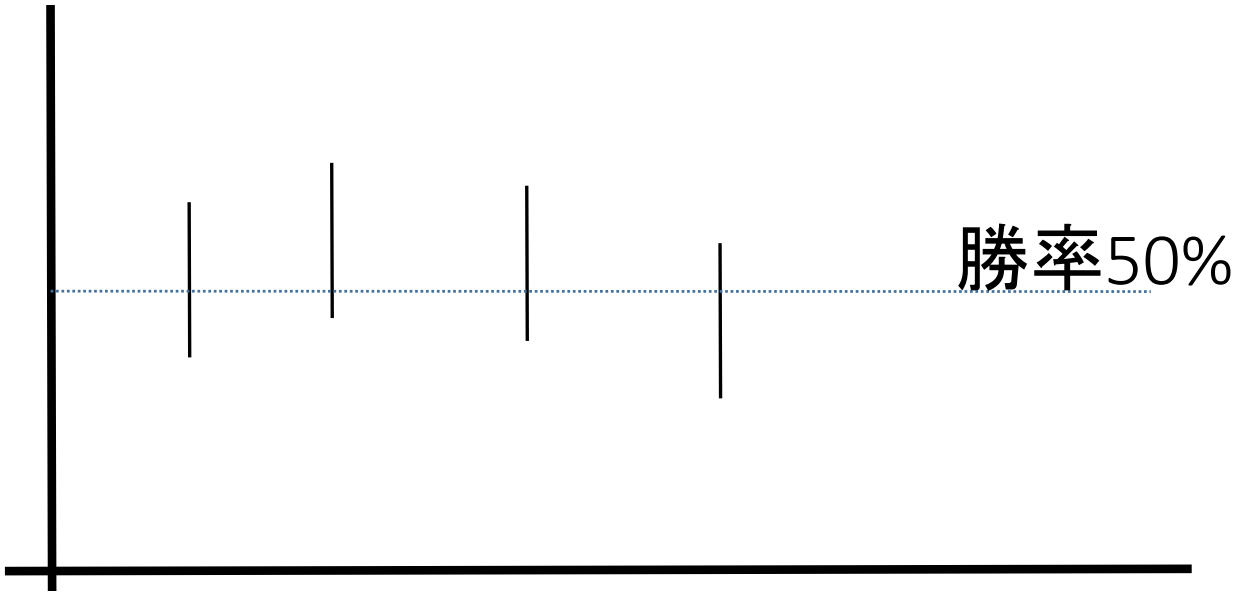
資源不足がもたらす問題：レーティング測定すらできない

多用なパラメタ、ゆっくりした成長

一度の改造でのレート上昇は精々20-50程度（であることが多い）
勝率に換算して55%弱。500局程度の対局が必要
学習パラメタ、教師の組み合わせは数十単位
ご家庭用PCではレート測定さえ終わらない
測定のかなりの部分を有志(Rotaさん)に頼っている

解決策：人力 Gaussian Process

評価関数のモデリングから事前確率モデルを立てることで
より少ないデータでの最適化を目指す



例：
学習回数に対する勝率は単調減少/単調増加/単一の峰を持つ凸関数のどれかであると仮定することで、ノイズの大きい勝率データから学習の良し悪しを推測

大会当日は計算力がないと勝ちにくい

年々進化し続けるAWSの最高火力
くじらちゃんを始めとするクラスタ化技術の発展に伴う計算資源
チキンレースの激化

解決策：同人誌やイベントを駆使した資金調達

サインインしてチェックリストに追加

サークル名	河童2.0（カッパツーポイントオー）
配置	え30
ペンネーム	SAWADA Ryoto
Webサイト	https://twitter.com/Qhapaq_49
ジャンル	ソフトウェア全般

- 大会中のクラスタ構築は出来そうな額は調達できました
- 皆様有り難うございました
- 2019年4月の技術書典には出ません。ネタを書く余裕がなかったのです。申し訳ありません

その他

角換わりに飽きた

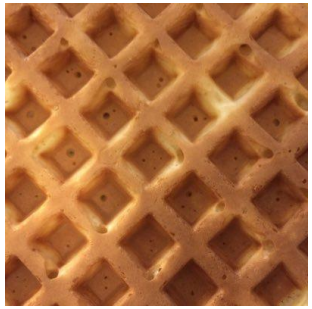
- 角換わりを避ける定跡を作りたい
- しかし、ソフトは定跡なしだと角換わりを指しまくる
- 人間の棋譜も角換わりだらけである
- 定跡は評価関数との相性もあり、生成、テストに時間がかかる
- ご家庭用PCで戦える余地は当然ない

棋譜のビッグデータから特定の戦型を抽出、検品する

- 棋譜の戦型を細分化し使いやすい戦型(角換わり以外)を探す
- 短い持ち時間での解析結果を補外することで検品コストを削減

HoneyWaffle
第29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やニコニコ放送、AbemaTVの将棋中継が好き。

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。

※表紙やTwitterアイコンのワッフルはうちで焼いたものを使用しています。

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

戦績ふりかえり(1)

2016年、Go言語でオリジナル開発版

・第26回世界コンピュータ将棋選手権　出場　一次予選2勝5敗

・第4回将棋電王トーナメント 出場　予選リーグ3勝5敗

盤面データを複素数平面で表現しているのが特徴。Go言語なのにゴルーチンを使いこなせず、シングルスレッド動作。

戦績ふりかえり(2)

2017年、やねうら王ライブラリ使用

・第27回世界コンピュータ将棋選手権　出場　決勝リーグ 7位

・第5回将棋電王トーナメント 出場　決勝トーナメント初戦敗退

ここから振り飛車をテーマに。当初は、振り飛車にすれば対抗形になってじわじわとした渋い展開になる　→手数が伸びる→最初から時間を温存しつつ時間攻め、という戦略がメイン。電王トーナメントでは、振り飛車をひいきした教師局面生成を行い、それを学習した評価関数を通常の評価関数とブレンドすることで振り飛車評価関数を作成。

戦績ふりかえり(3)

2018年、Apery、やねうら王ライブラリ使用

・第28回世界コンピュータ将棋選手権　出場　決勝リーグ 8位

定跡、評価関数を振り飛車仕様にした上で、玉の美濃囲い落とし穴（ 28玉型にボーナス加点）を採用。

シードで二次予選からの参加。5勝3敗1分けでギリギリ決勝リーグに進出。決勝リーグでは 1勝6敗も、唯一の勝ち局は振り飛車党の方にはぜひ御覧いただきたい。

定跡の出来がいまいちで、評価値つきで見ると見どころのなかった将棋が結構あったのが反省点。

ハードの面でも、AWSで選べる一番いいやつにするべきだった。

ポエム(1) WCSC28以降の状況（個人の感想）

NNUE型の評価関数のレート向上がすごいことに。

（参照: https://www.qhapaq.org/shogi/ ）

また、NNUEkaiシリーズのたややん氏の、振り飛車評価関数もすばらしい。これこそが期待されていたものなのでは。

→評価関数界隈はとてつもないレッドオーシャン。海の生き物の名前も多いし。

→何か自信のある理論の裏付けがあるわけでもなく、機械学習のリソースも豊富ではないので、正直まともにぶつかり合うことすら難しい。仮にうまくいったとしても、評価関数だけで決定的な差をつけられるのか？

→というわけで評価関数は前回の KPPTのままで、それ以外のところで何とかできないか考える。

ポエム(2) AlphaZero完全に理解した（個人の感想）

将棋の知識なしでも膨大なリソースをうまく使うことで elmoを超えた、どやぁ（物量で殴ってるだけのようにも見える）。

千田七段（当時六段）の叡王戦パラダイス 12月号での棋譜解説は必見。電王盤の Ponanzaが拗ねたように起動しなくて、電王ぽんぽこと交代したのが、うける。

（ http://live.nicovideo.jp/watch/lv316751884 ）

モンテカルロ？だからか、詰みロジックがない？からか、勝勢になっても詰ましにいかない。

先手なら初手26歩固定なのが興味深い。

なお論文は全く読んでいない。

→学習の結果、NNの中に、現在の将棋ソフトにはない「あるもの」が実現されているように見える。

ポエム(3) 振り飛車について（個人の感想）

最近のソフトは平手初期局面の開始直後の数手で振り飛車は損だと見ていると、認めざるを得ない。対抗形にすると、勝敗問わず相居飛車よりも手数がかかりがちというのが影響しているのでは。仮に相居飛車が100手、対抗形が130手なら、例えば飛車先の歩を突く→勝ちまで1/100の価値の手、飛車を振る→勝ちまで1/130の価値の手。その数字に基づいて手を選んだ結果、自然と相居飛車の将棋になる、みたいな。手数が長いと不確定要素が増えて、短い将棋よりは勝ちにくくなるのでは？というのも。

ただ、飛車を振っただけで一気に-200とか、そういった数字ははっきりとおかしいと思う。ソフトとしては、いくつかある手のうちでわずかでもいい手を選んだ結果が居飛車だと思うので、振り飛車の手は居飛車の手よりも評価値が1とか10とか低いだけであるはず。そもそも-200は振った飛車を戻せ、振り飛車指すよりも2手損した方がまだマシだということだから、単にそのソフトが振り飛車を指せない、振り飛車の局面をまともに評価できてない、というだけのことで、振り飛車という戦法が悪いわけではない。と強く主張したい。

コンピュータ将棋での振り飛車のメリットは、こちらはほぼ必ず対抗形になると分かっているところ。相居飛車の定跡にはまることはなく、逆にこちらの土俵に引きずり込むことができる。まあ居飛車党のソフトからしたら、わざわざ自分から評価値を下げる相手だと思っているのだろうけど。少数派の振り飛車対策をしている暇は正直ないだろうし、学習の時にわざわざ振り飛車の局面も作るの大変だろうし、そういうところを突くのが楽しみ。

正直なところ、振り飛車をコンピュータ将棋でやるのは、個性派揃いの振り飛車党棋士の方々の領地を重機で荒らすような、無粋な行為だと思って後ろめたい面があった。また、振り飛車をコンピュータで調べていって、振り飛車のすばらしさを追求するつもりが、逆にこういう理由で振り飛車はだめなんだよという結果になったらイヤだなあというのもあった。ただよく考えれば、自分ごときがちょっとやったところで何の影響もないし、そもそも将棋はそんなに浅くないだろと。ということでやれることは何でもやっていきたい。

コンセプト

・「自分の戦い方で、最強の振り飛車党ソフトを目指す」

今回、評価関数のバトルには参加しない。

AlphaZeroそのものの再現や実装はしない。

AlphaZeroが実現したであろう「あるもの」は振り飛車にとっても重要なので、そこだけピンポイントで実装を試みる。

「あるもの」は決勝リーグ進出した場合に出す文書で種明かしできたらいいな、ということで。

ハードはAWS EC2インスタンス1台（極力高性能なものを選択）の予定です。今回こそ一番いいのを。

構成（予定）

■開発部（探索）

WCSC28で使用したやねうら王[1]ベースに、前述の「あるもの」を追加する[2]予定

■評価関数

WCSC28で使用した、Apery[3]ベースの振り飛車評価関数、進捗状況によってはtanuki-[4]のNNUE型に手を出すかも？

■定跡

WCSC28で使用したファイルをベースに、WCSC28後から5か月くらいかけて手動で作り込んだものを使用

■ライブラリ採用理由

[1]やねうら王：最高の探索、フレームワーク。2017年から使用している。[2]python-shogi：python使うので。

[3]Apery：KPPT最高クラスの評価関数。振り飛車との相性も悪くない。[4]tanuki-：最新最強の評価関数。

最後まで読んでいただき、ありがとうございます。

第29回 世界コンピュータ将棋選手権

大将軍（たいしょうぐん） アピール文書

開発者 横内 健一 横内 靖尚

○大将軍の概要

評価関数に主眼をおいた将棋ソフトです。

過去には4駒の位置関係（KPPP:N4(2013) / KKPP:N4S(2013-2015)）を評価関数に使用していました。

現在は、学習作業の効率を考慮し現在は3駒関係の評価関数を用いています。

○大将軍の特徴

評価関数の作成に関しては、以下の点を工夫しています。

（評価関数は、以下の2ステップの手順にて学習しています）

1.プロの棋譜からの学習

いわゆるBonanzaメソッドをベースに、手番の評価やミニバッチの手法を取り入れています。

ミニバッチを用いることで、学習が安定し、短時間で学習の成果を確認することができます。

手番に関する評価は、3駒に手番加えたKKPT型を採用しています。KPPT型よりも計算コストが小さいため、

10%程度探索速度が向上します。

2.自己の探索結果からの強化学習

プロの棋譜からの学習において勝率が飽和したところで、浅い探索結果と深い探索結果の評価値を用いて、

評価値の不整合を修正していきます。

学習させる局面により勝率に影響するようですが、とのように設定するとよいかは今後の課題です。

○ライブラリの使用と実装方針

やねうら王,elmoライブラリを使用します。選定理由としては、評価関数の開発に注力するために、

探索部分はライブラリを使用して他のソフトと同等レベルをキープしたいと考えました。

今日では、プロの棋譜を使わなくても評価関数を作成することが可能ですが、本ソフトでは、

従来手法の学習結果（ウナギ屋のタレ）を活用していきます。

第 29 回世界コンピュータ将棋選手権アピール文書

概要

「たこっと」は，電王戦を見てコンピュータ将棋に興味を持った筆者らが， フルスクラッチで実装した (している) 将棋プログラムです。 Web 上の解説記事や論文， Stockfish, Apery, やねうら王, tanuki- (wcsc28 版), Bonanza 6.0 のソースコードを参考にしています。

特徴

- 各種処理が AVX2 命令で実装されていて高速
- AVX2 命令を適用しやすいデータ構造 (非ビットボード)
- Stockfish 風の探索アルゴリズム
- ニューラルネットワークを使用した評価関数
- オンライン学習ルーチン

以前のたこっとから引き継いでいる特徴については過去のアピール文書を参照してください。

- 第 26 回世界コンピュータ将棋選手権アピール文書
- 第 27 回世界コンピュータ将棋選手権アピール文書
- 第 28 回世界コンピュータ将棋選手権アピール文書
- 第 4 回将棋電王トーナメント PR 文書
- 第 5 回将棋電王トーナメント PR 文書

ライブラリの使用について

以下のライブラリの最新バージョンを使用申請しています。

- **Apery**
  ニューラルネットワークの初回学習時に教師データを生成するために評価関数バイナリのみを使用する予定
  コンピュータ将棋ソフトのリファレンスと考えていることが選定理由

- やねうら王 コンピューター将棋フレームワーク
  現時点では使用予定はありません
  必要なモジュールがそろっており，ソースコードを読み込んでいて慣れ親しんでいるのが選定理由

- **tanuki- (wcsc28版)**
  現時点では使用予定はありません
  ニューラルネットワークの実装でソースコードを参考にしたことが選定理由

ライブラリを提供していただき，Apery の平岡様，やねうら王の磯崎様， tanuki- (wcsc28 版) の tanuki- 製作委員会様 (野田様、岡部様、鈴木様、日高様、那須様、河野様) には感謝いたします。

評価関数

WCSC28 で tanuki- が CPU で計算するニューラルネットワーク型の評価関数で成果を上げました。

当時 KPPT 型の三駒関係のモデルでは強化学習しても棋力の向上が微々たるものになっており，新たなモデルを考えるのが急務となっていました。 ニューラルネットワーク化するという発想にいたっても，線形の三駒関係のモデルより計算速度が遅くなり，弱くなりそうという意見が多かったように思います。 そんな中，非線形のニューラルネットワークのモデルでも十分戦えることを示した tanuki- 製作委員会の皆さまの成果は素晴らしいの一言につきます。

というわけで，たこっとの評価関数もニューラルネットワーク化しました。 単純に tanuki- の真似をしても計算資源の乏しいたこっとチームでは強化学習競争に負けてしまいます。新たな新天地を求めて特

徴を KP 2 駒から別のものに変えました。

特徴名は KKPAR とします。

> 詳細は大会終了後に公表します。

今のところ強くなってないため，誰の興味も引かないかもしれません．．．

学習ルーチン

以下に挙げるような近代的な機械学習の機能を一通り実装してあります。

- 確率的勾配降下法
- オンライン学習
- ミニバッチ
- 複数の損失関数
- 複数の最適化手法
- 正則化

最近 (2019 年 2 月末頃)，新しい最適化手法が発表されました。 AdaBound と AMSBound と呼ばれるものです。
以下に URL を上げておきます。

Adaptive Gradient Methods with Dynamic Bound of Learning Rate

これを実装予定ですが，PC のメモリが不足しており (ニューラルネットワークのモデルがメモリを大量に消費する)，動作はさせられないかもしれません。

また，学習の効率アップとは質の良い教師データ作成と考え，いくつかの手法を実装しました。

> 詳細は大会終了後に公表します。

nozomi アピール文書

2019/03/21 Yuhei Ohmori

インテリジェントな評価関数

- 特別に開発されたNNEU(**N**ew li**N**ear evaluation function divid**E**d by t**U**rn)評価関数は、より正確に局面を評価します。
- いままでの評価関数と比較して2倍の大きさを持つパラメーターにより、多様な局面に対して局面評価が正確になり、驚くような強さになる理由です。

卓越した教師局面

- より最高の教師局面のために、教師局面作成専用の評価関数を作成しました。
- 教師局面生成のための評価関数は、それ専用の学習により、常に更新され、よりパワフルにアップデートされます。
- 専用の評価関数は、より多様でより正確な学習のための局面を生み出します。

進化した指し手予測

- nozomiの指し手予測が進化しました。
- 進化した指し手予測はより正確になり、最高のオーダリングにより、最高の手を少ない探索で見つけることができます。
- さらに、より強烈な枝刈りを行うことが可能になり、より高速により深い探索が可能になりました。

最後に大事なこと

- Stockfishをベースに作成しています
  - https://github.com/official-stockfish/Stockfish
- 一部Aperyを参考にしています
  - https://github.com/HiraokaTakuya/apery
- 評価関数の学習のためにfloodgate等一般に流布している棋譜の途中局面を使用しています
  - 評価値は使用していません
- 今回は何もしていないので、実は去年より弱くなっているかもしれません

WCSC29 elmoアピール文書(暫定)

瀧澤 誠 : elmo開発者
Twitter: @mktakizawa

Mar.31.2019

NNUE難しい

単純に良い教師データを持ってきて学習すれば強くなるかというと、そうでも無い。

(弱くなるか、ほとんど変わらないかが多い)

何が起きているか、どうすれば強くなるのか、実際色々試して、強くなってから事後的に理解する。というのが普通なのかも、と感じました。

野良関数強いっす。

ネタは色々考えているのですが、
今から大きく変えるのは難しそうです
(すみません)

とりあえずやってきたことで
採用されそうなところを書いておきます

ゼロベクトルから評価関数作成

- Heの初期化(重みの初期化)をAffine変換層に導入
  - tanuki-のWCSC28版は2-3層間のActivationがほぼ0の(ニートな)ニューロンがあり、orqha1018まで同様の傾向がある。
  → 新しく作り直すことでこれを解消。
  ※ どこに収束するかは運次第であり、ニートなニューロンがあることで弱くなるかどうかは不明

最初の層にHeの初期化を入れると、Activationが0になり学習されない、という状況が多発し、ここは使えなかったです。
ReLUならHeの初期化すれば良い、というわけでも無いんですね。
(Affine層も含めて、tanuki-チームが敢えてHeの初期化を避けてるかも)

教師データ生成方式の変更

- 時間制限の対局棋譜を導入
  - 従来のDepth固定方式の他に、時間指定の対局形式による教師データ生成を利用しています。
    (観点はtanuki-チームと同様で、終盤のノード数増加による思考時間の偏りを平準化するもの)
    Node数固定も考えましたが、乱数要素の1つとなることを期待して時間指定としました。効率も良いしNode数固定にしようかな。

Depth固定とどっちが良いかは今のところよくわからないです。

Qhapaq方式の学習導入

- 兄弟局面の比較
  - 教師の指し手による局面の評価値が高くなるように更新

  割と本質的な学習をしている(実際指し手一致率は上がる)と思われるものの、一時的に良くなっても段々悪化していく印象あり。
  (なんか使いづらい…)
  ※ 私の実装が悪い気がします

大会までに考えていること

- 評価関数の高精度化
- 定跡の作成
- AWSに局面をちゃんと送信する(200手で落ちない)

大会までにやらないこと

- クラスタ化

※ 興味無くは無いものの、優先度が低い…
　単に余裕が無いだけ

WCSC30までにやりたいこと

1．NNUEへのCNN技術取り込み
2．アンサンブルな？評価関数導入

構想は作っていて、割とやる気満々だったのですが、
まず普通に既存手法で「NNUEを強くする」ことに
とても苦しんでおり、多分次の大会までお蔵入りです

ライブラリ選定理由

1. やねうら王
   - コードはやねうら王ベースに追加したり書き換えたりで作成しています。強いソフトであり対局/学習何れも利用しています。
2. tanuki-
   - NNUE関数を利用しています。8bitでの高速演算は今風で、とても完成度の高い評価関数だと思います。教師データ生成に利用しています。
3. Apery
   - 教師データ生成に利用しています。

割とelmo方式に言及されている方が多くて
とても嬉しい(恐縮)です。
＃作ってよかった

Novice
アピール文書

Novice とは

2015年から開発

- WCSC28 二次予選14位
- SDT5 予選18位
- WCSC27 一次予選13位
- SDT4 予選21位
- WCSC26 一次予選13位
- SDT3 予選25位

フルスクラッチで開発

- Stockflsh ベース
- 横型 Bitboard
- 王手延長
- オーダリングの調整
- 探索パラメータの調整

WCSC29 ver.

- チーム開発に移行
- ライブラリからの脱却
- 学習部の実装
- 3駒関係
- 探索部の向上
- クラスタの実装

メンバー

- 熊谷啓孝 【チーム代表】
  主任開発者
  【専門分野】舞台演出・演劇・身体表現 等

- 矢内洋祐
  ハードウェア担当
  【専門分野】 FPGA・プロセッサ・アイドルマスター 等

- 幅野莞佑
  学習部担当
  【専門分野】機械学習・セキュリティ 等

- 笹井雄貴
  棋力担当
  【専門分野】振り飛車・アマ4段 等

探索部

- StockFishをベースとする
- 各種枝刈りを手動で調整
- Mate1Plyなどを追加
- 置換表のKeyを32bit化

評価部

- 教師生成部と学習部を実装
  ライブラリからの脱却
- KPPT型評価関数
- elmo絞り
- 0から数百億局面を深さ8~10で生成し elmo式で学習
- 学習率・バッチサイズの調整

並列部

- LazyClusterを実装
  置換表をノード間で共有

- 大規模クラスタを構成
  Xeon Scalable プロセッサーを複数台接続 (台数未定)

- Omni-Pathを使った高速通信

今年のNoviceは
人数+物量勝負で
頑張ります

2019/3/30

Argo（あるご）アピール文

Yes！魔法少女いちむら・ユタカ

3月22日になりました。そろそろアピール文を書いて、ソフトを作ろうかと思いますね。現状としては「こころ」の主人公が卒業論文を書かないといけない、みたいな状態ですね。青空文庫から引用してみましょう。

夏目漱石「こころ」の二十五、

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

その年の六月に卒業するはずの私は、ぜひともこの論文を成規通り四月いっぱいに書き上げてしまわなければならなかった。二、三、四と指を折って余る時日を勘定して見た時、私は少し自分の度胸を疑った。他のものはよほど前から材料を蒐めたり、ノートを溜めたりして、余所目にも忙しそうに見えるのに、私だけはまだ何にも手を着けずにいた。私にはただ年が改まったら大いにやろうという決心だけがあった。私はその決心でやり出した。そうして忽ち動けなくなった。今まで大きな問題を空に描いて、骨組みだけはほぼでき上っているくらいに考えていた私は、頭を抑えて悩み始めた。私はそれから論文の問題を小さくした。そうして練り上げた思想を系統的に纏める手数を省くために、ただ書物の中にある材料を並べて、それに相当な結論をちょっと付け加える事にした。

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

↑私も「今まで大きな問題を空に描いて、骨組みだけはほぼでき上っているくらいに考えていた」ですよ。「こころ」の中で一番印象的な部分です。「私だけはまだ何にも手を着けずにいた。私にはただ年が改まったら大いにやろうという決心だけがあった。」とか、あまりにも痛々しくて読んでいる私が死にたくなるじゃないですか。というか、むしろここ以外に何を言っている話でしたっけ、これ。まあいいや。まあ頑張ってアピール文を書こう。ソフトは4月に入ってからなんとかします。というか本当にこの10ヶ月何してた＞私。

・去年から変化ないです

技術的な部分については去年から追加できたところはないのですみませんが技術的な独自性については去年の文を参考ください。

時代はどう考えてもNNUEですが、当方はいまさら三駒のキメラ化を試していました。現状は以下のとおりです。

キメラ化した評価関数を50回ずつ自己対戦させてみた結果。

AP-QH 4.1 17-7-26 APERY4.2

AP-QH 4.1 20-3-27 QHAPAQ4.2

AP-EL 4.1 21-4-25 APERY4.2

AP-EL 4.1 23-5-22 QHAPAQ4.2

AP-TN 4.1 20-9-21 APERY4.2

AP-TN 4.1 18-7-25 QHAPAQ4.2

AP-EL40 4.1 22-8-20 APERY4.2

AP-EL40 4.1 16-6-28 QHAPAQ4.2

AP-EL60 4.1 19-10-21 APERY4.2

AP-EL60 4.1 20-5-25 QHAPAQ4.2

AP-EL30 4.1 27-12-11 APERY4.2

AP-EL30 4.1 6-2-12 QHAPAQ4.2 これは50回やってない

AP-QH70 4.1 18-10-22 QHAPAQ4.2

AP-QH60 4.1 23-3-24 QHAPAQ4.2

AP-QH30 4.1 20-6-24 QHAPAQ4.2

AP-QH40 4.1 22-6-22 QHAPAQ4.2

AP-（EL-QH）67 23-5-22 QHAPAQ4.2

AP-（EL-QH）50 20-7-23 QHAPAQ4.2

AP-（EL-QH）75 21-3-26 QHAPAQ4.2

AP-（EL-QH）85 16-6-28 QHAPAQ4.2

AP-（EL-QH）60 17-9-24 QHAPAQ4.2

AP-（EL-QH）70 25-3-22 QHAPAQ4.2

256手で引き分け ソフトは両方共YO4.82、定跡はyaneuraBOOK3を使って確率で手を採択させた。
持ち時間は15分、秒読み10秒。
書いている通り左側がSSE4.1で右側が4.2でコンパイルして動かしています。
比較対象として対局しているAPERYとQHAPAQは両方共WCSC28のものです。略称は、
EL elmo WCSC28
TN tanuki- SDT5
AP Apery WCSC28
QH Qhapaq WCSC28
何も書いていないものは50%ずつキメラ化したもの、数字があとに書いているものはその割合でキメラ化したものです。
例として、AP-（EL-QH）70 は、
エルモとカパックを50%ずつキメラ化した評価関数とエイプリーを70%：30%の比率でキメラ化した、ということを表しています。
25-3-22とあったら左側から見て25勝3分22敗、という意味です。
使っているのはデスクトップとラップトップでCPUがそれぞれi5の何だったか調べるのだるいので調べませんが、4コア有る内の1コアで計算をさせています。

・AWS使いたいですね
やり方が分からなくて結局去年使えなかったので今年は頑張ってAWSを使うべく頑張りたいとは思っています。
去年の2次でタヌキさんチームとやったですよ。フッと見るとタヌキさんがm5.24を8つクラスタリングしていたですよ。「8つかー」って思ったですよ。私的にはアメリカ軍のB52爆撃機がジェットエンジンを8発搭載しているのをそのときに思い出してですね。それを相手にして1つも使わないで挑むっていうのはちょっと無理があるんじゃないですかってことを思ったわけですよ。ちなみにビールさんも5つクラスタリングしていましたね。まあ、そういうのとやらないといけないわけなんですよ。なんとか頑張って今回はAWSを使いたいと思うのですが、頑張りたいですね。

ちなみに、ハニーワッフルさんはm5.24ではなくてその一個下のm4.16を使っていたですよ。「なんでですか？」と伺ったら、サッカーでFC東京が好きだから東京リージョン以外のを使いたくないのですと言っておられました（東京リージョンにはm5.24がない）。私もF1でホンダが好きだから多分自動車を買うとしたらホンダ以外に買いたくないので気持ちがよくわかりました。ただ言えることは、もしハニーワッフルさんがm5.24を8つクラスタリングしたハードで闘っていたら決勝でもっと上位に行ったんじゃないかということを思わざるを得ないので、なかなか難しいなと思いました。というか、これなんの勝負なんですかね。

・ライブラリの選定理由

なんで「やねうら王」を使うのかということについて書きたいと思います。

私が「やねうら王」を使っているのはそもそもやねうら王nano/miniが当時公開されて、そのコードを読んで改造するところからコンピュータ将棋に入ってそのまま「やねうら王」を使い続けているということです。この話は去年の文にも書いたと思います。

「やねうら王」のすごいと思うところは改造をしてくださいという工夫というか情報提供が活発になされているところだと思います。

やねさんが110億局面の教師データを公開してみたり、シボレーの情報を公開してみたり、私も使いましたがキメラ化のコマンドについて情報公開がなされていて、やねうら王形式の評価関数もいくつか公開されています（使用申請をしたelmo,qhapaq,tanuki-,Aperyはそのキメラ化を試してみたいという理由によります）。そもそも当時に教育用のnano/miniが公開されたということも先進的です。

そして現時点ではWCSC28でタヌキさんチームが発表したNNUEがホットトピックであると思うのですが、現時点ではそれは基本的には「やねうら王」で使うもので、NNUEをやろうとなったらなにか特別な用事がなければ「やねうら王」を使わないですかね。このNNUEについてもカパックさんがブログで情報提供をしていたり、あるいは同人誌を発表されたりと、そういうふうに改造をしやすい環境が整っているから「やねうら王」のユーザーが多くなり、その結果としてやねうら王が強くなっているのではないかと思います。

・テンションが微妙な事もあって最近は作文ができません。そんなアンニュイな気分になることが珍し

いのでその状態を楽しんでいる最近です。それにしても、最近は欲望がなくなってきました。私はこれまで欲望が強い生き方をしていたので（特に名誉欲が強かった）こんなに欲望がない生き方をすることが珍しく、いつまで続くのかと思いつつも珍しいのでしばらくは欲望が出てくるまでは欲望のない生き方をしてみようと思っているところです。おかげで最近はぼんやりしていることが多く、そのぼんやりとあれこれ考えている状態が心地よいということを思っています。

というか、私は基本的に名誉欲が強い人だったのですが、昨年の選手権で「一次予選通過候補はArgo」みたいに私のことを言及してくれるツイートがあったのですが、それを見たらすごく恥ずかしくて、自分で「あれだけ名誉欲が強くて有名になりたいって思っていたはずなのに他人が自分について言及しているのがとにかく恥ずかしいってなんだろう」と思ったわけですよ。多分、有名になって他人から言及されることが多くなっても私はそんなに幸福には感じないのだろう、ということをそのときに思いました。それだけでこの選手権に参加してよかったと思います。ありがとうございます。

ポナンザの山本さんがツイッターで「勝ちたい」って定期的にツイートするじゃないですか。最近経済学の本を読んでいて、そもそも社会というのは「競争」ではなくて「協力」を基本としている、ということをしみじみと思うようになってきているので、そもそも勝つとか負けるとかいうことがどうでも良いことだと思うようになってきました。科学の研究でも「この分野は競争が激しい」とかって新聞記事とかに書いてあったりするじゃないですか。私が思うに、そもそも他の人がやっているというのならば他の人に任せたらいいんじゃないですかってこと。他の人がやってくれると嬉しいけど誰もやっていないから仕方ないから自分でやるかって言うことをそれぞれがやっていたらいいと思いますよ。将棋のプロ棋士とかも、勝つとか負けるとかってことをずっとやっているように思うのですが、それってそんなに重要なんですかね。学校を卒業して、試験で点を取ることがなんであそこまで重要だと思っていたのかがもはやわからないように、将棋の棋士をしていないと将棋の対局で勝つということがどうしてそんなに重要なのかというのがよくわからないんですよ。将棋の棋士とかの評価基準はすごく変わっていると思う。対局で勝利するほうが偉い・良い、というめちゃめちゃ変わっている基準。学校の評価基準、「勉強ができる・試験で良い点が取れる」やつが「偉い・良い」、という基準がかなり変わっているということは思います。新聞に「試験の点だけで評価しないでほしい」という学生の投書が時々載ったりする。学校のこの評価基準はとにかく公平に評価しようということをすごく重視して制度設計されていると今にして思う。というのも、学校を卒業して民間企業で雇われてみると、民間企業の評価基準が年

功序列とかコネだろうというふうにしか思えないので。なんの理由もなくコネで評価されるとか入社してから何十年もいる人がそれだけで偉いということを基準とする民間企業の評価基準を前にすれば、学校の「試験の点だけで評価する」ということがめまいがするほどに公平な評価方法であると思う。性格の良し悪しを考慮しないのですから。そして将棋の棋士の「対局で勝利すること」を評価基準とするというのもまたびっくりするレベルで変わった評価基準だと思う。

・将棋と経済学

そもそも思うことは、将棋の勝利条件についてです。藤井七段が朝日杯を2連覇したそうです。そういう感じのニュースを聞くたびに前から薄々思っていたのは「棋戦とかで優勝した一人だけが勝利者であって、残りの全員は敗者なのだろうか？」という疑問です。

大体において「対局で勝利すること」を勝利条件と置くと50%が勝利条件を満たせないじゃないですか。いくらなんでも敗者の比率が多すぎませんかと。基本的に適当にやっていたら全員が勝利条件を満たすことができる、というゲームじゃないとそのゲームは成立しないんじゃないですかということを思うわけですよ。将棋の勝利条件はなにか？ということを考えてみると、「将棋に関わるすべての人を幸福にする」ということがそもそも将棋というゲームの目的であって、「将棋をすることで幸福になった」ということが勝利条件なのではないかということを思うわけです。「勝ち負けに関係なく将棋をプレイして楽しかった」ということならば全員が達成可能であるはずだということを思うのです。それが勝利条件であり、それゆえに全員が勝利者になることが可能であると思う。

遊園地を考えてみましょう。ディズニーランドとかUSJとか。遊園地の目的は「来る人を全員幸福にする」ということだと思うので将棋と立ち位置は近いと思うのですよ。そして三人で遊園地にでかけて三人とも楽しかったというのならばそれは全員が勝利条件を満たしているじゃないですか。「アトラクションを回った数が一番多い人が勝利者」とかが遊園地の勝利条件ではないはずだ。勝者は一人だけで残りの二人は敗者ということでもないはずだ。将棋の勝利条件もそういうものであるはずなんですよ。

個人的に経済学の本を読んで経済学の勉強をしていました。その話を書きます。個人的には自然現象を数学的に取り扱うのが物理学で、社会現象を数学的に取り扱うのが経済学という印象を持っています。それなので物理学を勉強していると経済学は物理学のアナロジーで理解するとわかりやすいと思いまし

た（ついでにいうとツイッターで流れていた「社会学は非計量経済学」というのが社会学の定義？としてしっくりきています）。

「有限の財の効率的な分配方法を考える学問」というのが経済学のオーソドックスな定義だと聞いています。大雑把にいってお金で買えるモノ・サービスを「財」と呼びます、多分。

例として国民が10人でその年のお米が1000ｋｇ収穫できた、という国を考えます。単純計算すると一人100ｋｇずつお米を分配するということになりますが、年が若い男性はたくさん必要でしょうし、女性・子供・シニアの人は100ｋｇもいらないかもしれません。そういうときに、管理人が「あなたは130ｋｇ，あなたは70ｋｇ」というように配ってまわるという方法が一つあります。そしてもう一つの方法が、お金というシステムを使って分配を行う、という方法です。私達日本人が通常行っているようにお金（貨幣）を使って個人個人が勝手に買い物をする、という方法でお米を好きなだけ買うということをみんながすると、最適な分配が自動的に実現できる、というふうに考えます。それで現在の日本なんかでは「国民それぞれが好きなように買い物をする」という方法で財をやりとりすると、最高効率の分配が最小のコストで実現できる、というふうに考えて、可能な限り、お金というシステムを用いてというか、「個人個人が好きなものを好きなだけ取る」という方法を基本として分配を行っているのだと思います。そしてその方法が使えない種類のモノ（税金とか、生活保護とか、日本ではないけど徴兵制とか）については「仕方がないから」やむをえずに、管理人が強制力を用いて分配を行う、ということをしているのだと思います。

そして、そもそもの話として、分配の基本は、「1.満足度（効用）を最大化するように2.財を分配したい。それを実現するための道具が3.お金（貨幣）である」というその順番です。一番重要なのは「満足度（効用）」です。二番目に重要なのが「財」です。「お金（貨幣）」は三番目に重要なものです。多分その順番です。

お金についての本を読んでいると大抵の場合「お金が一番重要だ」というような書き方をしているのでなんでかなと思うのですよ。ポナンザの山本さんが面白いとツイッターで書いていたので「お金2.0」という本を読みました。この本は面白かったですがやっぱりお金が一番重要だというスタンスで書いてあるように読めてそれがなんでかなと思うんですよ。どう考えてもいちばん重要なのは「満足度（効用）」じゃないですか。どうしてその視点があまり出てこないのかと。

そして物理学においてエネルギーが生成・消滅をせずに姿が移り変わっているアナロジーとして、お金というものも生成・消滅はしないで財との交換を行うことで移動するという印象を持っています。「宝

くじで10億円当たらないかなー」というようになんにもないところからお金がポンと湧いて出てほしいという種類のお金の欲しがり方をする人が時々いますがそれは物理学で言うエネルギーが湧いて出る永久機関みたいな感じで多分実現できない、多分お金は財との交換によって手に入れるものであると思う。生成・消滅を行うのは「財」であるというイメージを私は持っています。だからお金を手に入れようと思ったら財を提供しないといけない。もうちょっというと「他人を幸福にする」ということをするとお金が対価として得ることができるのだと思います（実はこの問題は根深いと思う。お金を財と交換する、というのではなく、順番が逆で「お金と交換されたものを財とみなす」ということで財を定義しているのではないかと思う）。

だから「お金をたくさん得たい」というのならば、「たくさん他人を幸福にする」ということで実現が可能であるはずだと思う。それがこのシステムのすごいところだと思います。「皆さん、利他的に振る舞ってください、他人を幸福にしてください」と呼びかけてもそれができる人は少ないと思う。他人はどうか知らないけど少なくとも私はそんなに利他的には行動できない。だけど、「他人はどうでも良いからとにかく自分がお金持ちになる」ということならばモチベーションとしては維持できる気がする。そして「自分がお金持ちになる」には「他人を幸福にする」ことが必要なので、それぞれが利己的に行動すると結果的に「他人をたくさん幸福にする」という利他的なことを結果的にしてしまう、ということになってしまうのだと思う。常識的に考えると「他人を幸福にする」という利他的なことと、「他人なんかどうでもいいから自分だけ得をすれば良い」という利己的なことは両立させることは難しいと思う。だけどこのお金というシステムにおいては「他人を幸福にするとお金が手に入る」というゲームのルールになっていることによって、「他人なんかどうでもいいから自分だけ得をすれば良い」ということがそれすなわち「他人を幸福にする」ということと結びついているように私には見える。それが本当にすごいというかよくできたゲームのルールであると思う。だからお金について書いてある本で、「お金が一番重要だ」という視点で書いてあっても実用上あまり不都合はない。お金儲けを頑張ってするということはイコールで他人を頑張って幸福にするということにつながっているのだから。

将棋の棋士はどうなのでしょうか。上に書いたように「将棋の勝利条件は勝ち負けではなく、将棋をして楽しかったということ」だとか言っても、「何言ってるんだ、勝つか負けるかだろう」というふうに言う人は一定数いると思うんですよ。それは経済学において「満足度が一番重要だ」ということに対して「満足度とか何言っているんだ、お金が一番重要だろう」というふうに言う人が一定数いるだろうということと同じです。

このお金というシステムにおいて、利己的なことが利他的なこととダイレクトにリンクしているように見えることと同じように、将棋の棋士においても「とにかく自分だけ勝てばそれで良い」という利己的な振る舞いを全員が取れば、結果的に全体としてうまくいく、ということが実現できているのでしょうか？それができているのがこのお金というシステムなので、ここまで将棋が続いているのならばそういう事ができているのだろうということを思うのですが、そのへんのロジックが私の中でまだしっくりきていないでいます。

注：上記の内容はコミックマーケットC95で発行された、サークル「アニ☆ブロ」様の同人誌「アニブックログ2018 冬の増刊号」に寄稿した作文「『りゅうおうのおしごと！』に見る将棋と経済学」に記載した内容と一部重複しています。

将棋の棋士という商売のビジネスモデルが何なのだろうかということを思います。誰に何を売っているのか？ということが考えてみてもよく分からないんですよね（言い方を変えると、誰に対してなんの価値を提供しているのかがよく分からない、少なくとも新聞社に棋譜を売っているとかではないと思う、そう考えるとBtoBだけど将棋の棋士というビジネスモデルはどちらかというとBtoCに近いんじゃないかと思うので）。これから持続可能なビジネスモデルなのかということもよくわからないです。当事者がそのへんを考えて運営をしているのかどうかもよく分からないです。

　そういう事を考えていると思うのが、世界はそういうビジネスのほうが主流になりつつあるのだろうということです。30年くらい前はまだ食料が重要であるという考え方は今よりも強かったと思うのですよ。個人的には宇宙に行ったTBSの秋山さんがTBSをサクッと辞めて自給自足の農家の暮らしを始めたあたりが印象的でしょうがなくて。私は古い人間なので「農業は大事である」「突き詰めて考えれば秋山さんがやった自給自足の農家の生活こそ正しい生活である」ということを思うわけですよね。そういう感じだったので私はとりあえず製造業の会社に入りました。ただ、最近知ったのですが2018年位には、日本での一次産業従事者って4-5%で、二次産業従事者が25%くらいで、三次産業従事者が70%くらいなんですよね（働いている6000万人ちょっとの内の5%・300万人くらいだけで日本の1億3000万人分の食料って供給できちゃうんですね）。あとは2018年の世界の企業の時価総額ランキングで、1-6位がGAFAとマイクロソフトとバークシャー・ハサウェイで、7位がアリババで8位がテンセントだったのを見たと

きの「あれ、今って農業とか工業の時代じゃないんだな」っていうあの感じとかですね（今更かって感じですが）。多分いま現在とこれからは将棋の棋士みたいに「なんだかよくわからない」職業が一番、みたいな感じなのかなと思います。あと思うのは任天堂ですね。あの任天堂って会社は「自分達が売っているものは人間が生きていく上で全く必要がないものである」ということをよくわかった上で、「人を楽しませるにはどうすればよいか？」ということを徹底的に考えている会社だと私は思っているのですが、そのへんは将棋の棋士にも通じるものがありますね。

・アニメ「エガオノダイカ」感想　3/22までなので最終回まで見てません、ついでに今週の分もまだ見ていません

　なんでコイツらは戦争をしているの？という話ですよ。
　どうやらエネルギー源となる鉱石「クラルス」を巡って争っているらしい、ということもあれば、そのクラルスを動力源とした食糧生産プラントの描写があったから「エネルギー＝食料」であるという世界でも有るらしい。そして食糧生産プラントのエネルギー源にもなるし、軍事用途でも使うと強力、であることから、クラルスが現代社会における「原子力」のメタファーなのかということも思う。最近の話だと「クラルスはこの惑星をテラフォーミングしたときに散布した惑星改造用のナノマシンからエネルギーを取っている」ので、クラルスをエネルギー源として使用すると惑星の環境が悪化してしまう、ということになっていてますます面倒くさい感じになっているっぽい。ニコニコ動画でみていると「電気使うなと今更言われても困るよなー」というコメントが流れていた。

　戦争が起きる原因的なものの経済学的なスタンダードな解釈は、財の絶対的な不足による、というものだと思います。

「少女終末旅行」のアニメ版1話「戦争」にあった、
「戦争って殺しあうんでしょ。なんでそんなことするんだろうね」
「さあ…相手と自分の利害が一致しなかったり、例えば3人いるのに食料が2人分しかないみたいなときに、武器をとって戦うしかなくなるんだよ」

↑これですね。

それなので、経済学的な考え方をするのならば、財が足りている状態を維持することができれば戦争は発生しない、ということなんだと思います。実際のところは財が足りていたとしてもなんか争うんじゃないかということは思わなくはないですが、大雑把に言えばそういうことになるとは思います。「マクドナルドがある国同士は戦争をしない」という「マクドナルド理論」とかもそういう話をしているんじゃないかと思います。早く平壌にマクドナルドオープンしないですかね。グラミン銀行のムハマド・ユヌスが「貧困は博物館行き」と言っていますが、貧困が博物館行きになった時点で戦争も博物館行きになるのではないかと思います。

本作を見ていても、グランディーガ帝国の皇帝がソレイユ王国の首都を陥落させて「勝負は決した」と言いながら軍備の拡張をまだやろうとしているシーンが有りました。このシーンを見ていて、グランディーガ帝国がソレイユ王国に攻め入ったのは財が不足したとかではなくて、ただでさえ少ない財を軍備の拡張に使ってしまったからではないか、ということを思ったわけですよね。財の絶対的な不足ではなくて失政によるものではないかと。

さり気なく描かれているシーンから見ている私が勝手に妄想すると、「この戦争はやらなくてすんだ戦争だったんじゃないか？」ということを思わざる得ないんですよね。

あとは引っかかるのはソレイユ王国の王様であるユウキの姿勢ですね。ユウキは戦争をするにあたって、自国のみならず敵側の死者もなるべく少ないほうが良い、という姿勢で指揮を取ります。私の知り合いはユウキのこの姿勢を「甘い」と言っていましたし、ソレイユ王国のユウキ以外の人もついていけていない考え方なわけですけど、私はそれはそれで一つの思想である、とは思いましたし、戦争をしていて「敵側の死者も少なく」という立場を取ることができるのはむしろすごいんじゃないかとびっくりしています。ユウキはその姿勢で首尾一貫しています。「こういうふうでよいのだろうか」と悩むシーンとかは一切ない。私は自分がソレイユ王国の王様だったとしてこういうふうな振る舞いができるだろうかと考えると自信がなくて、だからユウキは少なくとも私よりもよほど強い人間だと思う。私以外の見ている人がどう思っているかはわかりませんが私がすごく印象的なのは「ユウキの人間的な強さ」についてですね。それが印象的でしょうがない。

見ていて思うことは、ソレイユ王国は日本国のメタファーである、ということです。だから日本国がこういうふうであってほしい、というメッセージ的なものを私は勝手に感じています。

・（広告・宣伝）コンピュータ囲碁の大会「AI竜星戦」に皆さんぜひ参加しに来てください

　2018年12月の中旬にコンピュータ囲碁の大会AI竜星戦2018が開催されました。そのときに2019年もやりますという話があったのでおそらくは2019年12月の土日にAI竜星戦2019が開催されると思うのですが、ぜひ皆さんも参加していただけるとありがたく思います。2018のときはエントリーが15チームで寂しかったのでこの将棋の大会みたいに60チームくらい来てくれると賑やかで良いと思います。よろしくご検討いただきたくお願いします。

WCSC29 W@ndre アピール文書

Updated 2019/03/17
Dated 2019/01/17

由来

W@ndre は wander で wonder な将棋を指してもらう意図で命名したプログラムです。

実績

- SDT5:19 位
- WCSC28:16 位
- 振り飛車評価関数 KPPT の作成:
  - ツイートリンク
  - 配布リンク (GoogleDrive)

本大会での特徴

- 人間の棋譜を用いない振り飛車評価関数(３駒型)の利用
- ネットワーク構造を変更した振り飛車 NNUE 型評価関数の使用
- 定跡に向飛車を採用

人間の棋譜を用いない振り飛車評価関数(３駒型)の利用

ここ数年で将棋ソフトの棋力は著しく向上しましたが、その一方で学習過程で居飛車を過大に評価するようになった可能性も示唆されています[1-4]。
これまで振り飛車を評価する様々な評価関数作成の試みが行われており ・定跡などで振り飛車局面を多く含んだ教師を生成する[5-7]・振り飛車局面を過大に評価させた教師を生成する[8]・振り飛車が負けた局面を取り除いた教師を用意する[9]・学習時に振り飛車局面で出現しうる形を評価する[10, 11] などの手法で既存の評価関数に対する追加学習、ゼロベクトルからの学習において成果が確認されています。
今回のW@ndreでは、通常の初期局面が飛車先を突きやすく居飛車を過大評価しやすい状況だと仮定し[*]、十数種類の特定初期局面 (Fig. 1) から教師局面を生成し、強化学習を行うことで居飛車を過学習しない評価関数の作成を行います。
本学習はやねうら王[12]・習甦 (WCSC27,28)[13] より着想を得ました。

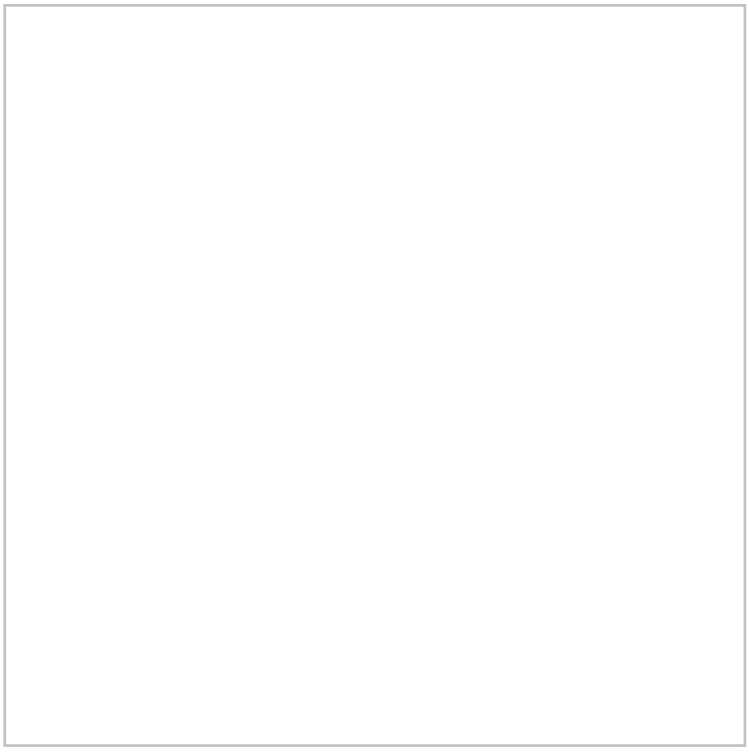


Fig1. 初期局面例
画像生成に局面ジェネレーター を利用しました。

*なお、初期局面において飛車先を突きやすいと仮定しましたが、入門者に将棋を教えている際に閃いたものであり、実際に平手初期局面で強化学習を行った際に初期段階で居飛車の局面がどの程度出現するかは調べたほうがいいのかもしれません。
例えば、評価関数を読み込まない時のエンジンの指し手が飛車先の歩を突くかどうかは探索部に依存し、やねうら王(v4.83)や技巧2では16歩など端歩を突く、shogi686(SDT4)、Apery(WCSC28)では飛車先を突く、nozomi(WCSC28)は角道を開けるなどの指し手の差を確認しています。

ネットワーク構造を変更した振り飛車NNUE型評価関数の使用

ネットワーク構造を変更した振り飛車NNUE型評価関数を使用します。少ない教師で実験したところよさげなネットワークを見つけた(本当か？)ので、そろそろ教師を増やして学習を行う予定です。
学習に際しては・上記で作成した評価関数より教師を用意する・定跡で振り飛車局面を増やした教師を用意する・やねうら王ライブラリで配布された教師[14]を利用する などを予定しています。

定跡に向飛車を採用

去年の採用定跡は角頭歩、阪田流向飛車、都成流、ダイレクト向飛車だったのですが、振り返ってみると向飛車ばかりでした。
それならば角不換型の向飛車も指せた方がいいだろう、ということで今年は向飛車のレパートリーを増やします。

参考

(1) http://yaneuraou.yaneu.com/2019/01/31/AlhpaZeroが居飛車を好む件
(2) https://twitter.com/Qhapaq_49/status/1048752286128717827
(3) https://twitter.com/tayayan_ts/status/1039442167041732610
(4) https://twitter.com/_illqha/status/1039472978931503104
(5) tibigame. 2017. HandicappedRook, https://twitter.com/tibigame/status/880757081632002049
(6) W@ndre. 2018. 第 28 回世界コンピュータ将棋選手権アピール文書
(7) FF2. 2018. 振り飛車を学習させた結果,

https://twitter.com/mafu_op_theory/status/1049979306170966016
(8) HoneyWaffle. 2017. HoneyWaffleSDT5
(9) NNUEkaiXF. 2018. https://twitter.com/tayayan_ts/status/1095668210412613636, https://twitter.com/tayayan_ts/status/1094237851942772736
(10) tttak. 2017. やねうら王の教師局面で学習した技巧2の評価関数（params.bin）_20171214
(11) tttak, 2018. 振り飛車評価関数
(12) やねうら王リゼロ. 2017. http://yaneuraou.yaneu.com/2017/06/12/人間の棋譜を用いずに評価関数の学習に成功/
(13) 習甦. 2017-2018. 習甦WCSC27アピール文書, 習甦WCSC28アピール文書
(14) http://yaneuraou.yaneu.com/2017/11/21/depth10で作った110億局面の教師データ、期間限定で公開します

- 以前のW@ndreのアピール文書
  - 第 28 回世界コンピュータ将棋選手権アピール文書
  - 第 5 回将棋電王トーナメント PR 文書
- QRLリンク
- 定跡のレパートリー増やしに以下の棋書を参考にしました。
  - 加藤治郎　「新・将棋の公式」
  - 島朗　「島ノート」
  - 勝俣清和　「消えた戦法の謎」
- 以前から参考にしている棋書は以下のとおりです。
  - 米長邦雄　「角頭歩戦法」
  - 西川和宏　「角頭歩戦法」
  - 飯塚祐紀　「奇襲振り飛車戦法」
  - 豊川孝弘　「パワーアップ戦法塾」
  - 木屋太二　「超阪田流角命戦法」
  - 湯川博士　「奇襲大全」
  - 週刊将棋　「痛快！ワンダー戦法」

使用ライブラリ

以下のライブラリを使う予定です。

- やねうら王
  最初に開発を始めた頃、定跡DBが編集しやすいということで選択しました。それ以降もソースを読み慣れて改良がしやすいので使い続けています。

- **tanuki-**
  NNUE型の評価関数を利用するため申請しました。

- ~~**dlshogi**~~
  ~~前回お世話になりました、今回もお世話になる予定です。~~今回は振り飛車評価関数に注力する関係で使いません。

- **python-shogi**
  現状では具体的な利用方法は煮詰まってないです。エンジン呼び出し等に使うかもしれません。

ライブラリを提供していただける、やねうら王のやねうらお様、tanuki-のtanuki-製作委員会様、dlshogiの山岡様、python-shogiのグニャラくん様に感謝いたします。

Contact: Twitter Account

「習甦」

探索

・ルート局面でdf-pn[1]により1Threadで詰みを探索する。

・α-β探索部はStockfish[2]を参考に実装した。

評価関数

・玉の位置に対する各駒の位置と各升の利き数を特徴量とするニューラルネットワーク型の評価関数とした。

・利きのデータ構造については技巧[3]を、ニューラルネットワークのアーキテクチャについてはNNUE[4]を参考に実装した。

評価パラメータの機械学習方法

・5手目までMultiPV 16として16^5局の自己対戦棋譜を数セット作成し、評価値の推移をフィードバックして推定された勝率を割引報酬として強化学習を行った。

・ニューラルネットワークの初期化状態から短時間でパラメータを最適化するためAdam[5]をベースとし、annealingやweight decay[6]等を用いた。

[1] https://ci.nii.ac.jp/naid/110002726401

[2] https://github.com/official-stockfish/Stockfish

[3] https://github.com/gikou-official/Gikou

[4] https://github.com/ynasu87/nnue

[5] https://arxiv.org/abs/1412.6980

[6] https://arxiv.org/abs/1711.05101

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。

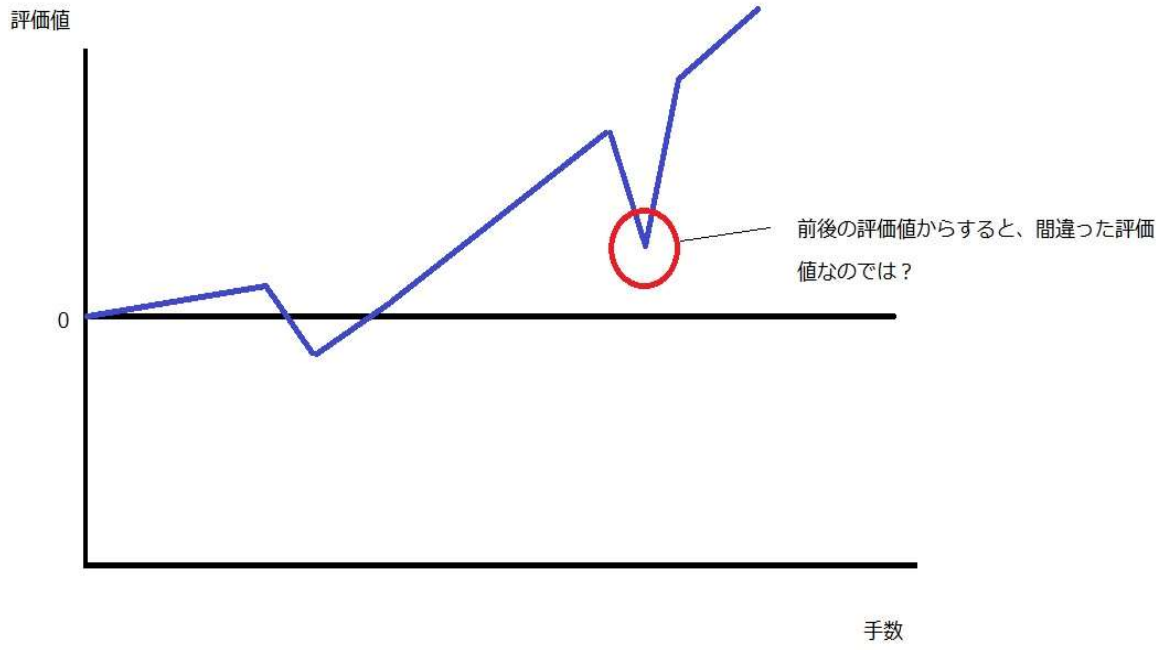
の高い評価関数を作成するという手法です。

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

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

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。

●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。

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。

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