GA将？ アピール文書

2019年3月16日 森岡 祐一

概要

- GA将は私（森岡）が趣味で作成しているコンピュータ将棋プログラムです。
- “GA将”の読み方は“がしょう”です。
- 初期バージョンの学習ルーチンにGA（遺伝的アルゴリズム）を使用していたので、この名前にしました。が、現在はGAは全く使っていません。
- 昨年はVer.9だったので“GA将!!!!!!!!”でしたが、今年はVer.10なので“GA将？”になりました。‘？’１個で‘！’１０個分という表記方法です。
- なお、本アピール文書は読み易さを重視して記載していますので、一部の表現には厳密さが欠けている事が有ります。ご了承下さい。

特徴

- 自己対局の結果から、強化学習[1]を用いてゼロから自力で学習可能です。
- 探索ルーチンの並列化はせず、シングルスレッド探索のクライアント8つで多数決合議[2]を行います。
- 合議の票の割れ方に応じて、思考時間の延長をしています。
- 定跡は一切無し。

探索ルーチン

- αβ探索、全幅ベースで各種枝刈りを実装。
- 静止探索はKFEnd流の2段階のもの[3]。ただし、生成する手は「駒を取る手と駒が成る手」に変更しています。
- 正直、ここは特筆すべき事は有りません。

評価関数

- 以下の評価項目からなる、線形の評価関数です。
  - 駒割
  - PPT（二駒関係＋手番）
  - 王将の移動可能範囲[4]
- 局面評価とパラメータ修正で同じ制御構文を書かずにすむ様に、C++のtemplateを使用して、実装を工夫してあります[5]。

合議

- 多数決合議を採用しています。
- 評価関数パラメータは、後述する学習ルーチンを用いて生成した1つのパラメータベクトルをベースにします。
- このパラメータベクトルを、Deep Learningで用いられるDropout[6]の様に「一部のパラメータをランダムに0にする」事で、複数のバリエーションのパラメータベクトルを生成します。
- 合議クライアント数は、通常探索8クライアント＋現局面での詰将棋ルーチン1クライアントです。

思考時間制御

- 前述した様に、GA将？では多数決合議を使用しています。
- そこで、「合議クライアントの票の割れ具合」をベースに思考時間の延長制御を行っています。
- 例えば、「手Aに投票したクライアントが7個で、手Bに投票したクライアントが1個」の場合は、「おそらく手AでOKだろう」と判断して、そこで思考を打ち切ります。
- 逆に、クライアントごとに投票した手が割れた場合は、現局面は難しい局面だと判断して、思考時間を延長します。

学習ルーチン（１：基本）

- GA将？の一番の売りです。
- “PGLeaf Drei”と命名した、強化学習とαβ探索を組み合わせた手法です。
- 基本となるアルゴリズムはPGQ[7]で、αβ探索のPV Leafの評価値を元に勾配計算を行う様に拡張しました。
- PGQとは方策勾配法とQ学習を組み合わせたアルゴリズムで、各アルゴリズム単体よりも良い性能を叩き出す事が、実験的に示されています。

学習ルーチン（2：拡張）

- Noisy Networks[8]風に各パラメータに「平均と標準偏差」を持たせ、その両方を学習します。
- これにより、自己対局時の棋譜のバリエーションが増え、学習効率が向上しました。

学習ルーチン（3：学習の流れ）

1. 評価関数パラメータを、極小さな乱数で初期化する。
2. 100局自己対局を行い、その結果からPGLeaf Dreiで評価関数パラメータを修正する処理を、10回繰り返す。
3. エース評価関数（過去最強のパラメータの評価関数）相手に先後入れ替えて計400局対局を行い、勝率が55%以上であれば、現時点でのパラメータをエース評価関数のパラメータとする。
4. 2に戻る。

学習ルーチン（4：アルゴリズムの概要-1）

- まず、自己対局時はSoftmax方策を用いて、ランダムに指し手を決定します。ただし、完全にランダムでは無く「良さそうな手ほど高確率で選択する」様になっています。
- 方策勾配法では、勝った時は「本譜の手は良い手だった」、負けた時は「本譜の手は悪い手だった」と判断し、評価関数パラメータを修正します。
- 例えば、勝った場合は本譜の手の評価値が上がり、それ以外の手の評価値が下がる方向に、評価関数パラメータを修正します。
- 負けた場合は、評価関数パラメータの修正方向が逆になります。

学習ルーチン（５：アルゴリズムの概要-2）

- Q学習では、本譜の局面の評価値を、その2手先（次の自分の手番）の評価値に近付ける様にパラメータを修正します。
- 方策勾配法は「自己対局の結果から、強い（勝率の高くなる）評価関数を直接作る」事を目的としたものであり、Q学習は「自己対局の結果から、精度の高い（勝率をより良く近似する）評価関数を作り、間接的に棋力向上を実現する」事を目的とするものです。
- これら2つの学習ルーチンを組み合わせる事により、従来のPGLeaf単体よりも棋力が向上しました。

学習ルーチン（６：対局数等）

- 探索条件は全幅1手＋静止探索です。
- 24時間あたり10万局程度の対局速度です。
- おそらく、2週間～1ヶ月程度は学習を行わないと、強くならないと思われます。

参考文献（1）

[1] Richard S.Suttonほか. 強化学習. 森北出版, 2000.

[2] 伊藤 毅志 ほか. 将棋における合議アルゴリズム: 多数決による手の選択. 2011, 情報処理学会論文誌 52-11, 3030 - 3037p.

[3] 有岡 雅章. 松原 仁編. コンピュータ将棋の進歩4: 将棋プログラムKFEndにおける探索. 共立出版, 2003, 18-40p.

[4] https://gasyou.hatenablog.jp/entry/20100922/1285152080

参考文献（2）

[5] https://gasyou.hatenablog.jp/entry/20120508/1336489213

[6] Nitish Srivastavaほか. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. 2014, Journal of Machine Learning Research 15, 1929-1958p.

[7] https://arxiv.org/abs/1611.01626

[8] https://arxiv.org/abs/1706.10295

最後に

- SPECIAL THANKS
  - 強化学習関連のご指導：mooopanさん（@mooopan）
  - ディスカッション等：桜丸@mEssiah_β1さん（@sakuramaru7777）
- 宣伝など。興味を持って頂けたら、御覧下さい。
  - Blog：https://gasyou.hatenablog.jp/
  - Twitter：@Gasyou_Official、@MoriokaYuichi
  - Web Site：http://gasyou.is-mine.net/

762alpha アピール文書

2018年3月30日 V0.1

天野史斎

■ まえがき

762alphaは学習の自動化を目的とするコンピュータ将棋ソフトウェアです。

今回も新機軸（と私が勝手に思い込んでいるﾌﾞﾂ）が盛りだくさん！

…orz

■ 新機軸（と私が勝手に思いこんでいるﾌﾞﾂ）

＿人人人人人＿

(その1) ＞ 今さら ＜ 新しい局面表現 ＦｌｏｗＢｏａｒｄ

￣Ｙ＾Ｙ＾Ｙ＾Ｙ￣

局面を(駒の移動先dst, 移動方位bn)の集まりとして表現します。

これは置駒の手番を無視して移動可能性のみ抽象化したもので、Flowと勝手に名づけました。

盤面を構成する一連のFlowは、[dst]にある駒の価値（手番を考慮しない絶対値）の降順に

ソートされた状態を保ちます。

＜メリット＞

1. あえて手番を含まない形をとることで、盤上の利きの書き換えを最小化する規則が簡単化、

2. 置駒を[dst]に移動させたときの駒得が常時丸見えのため、理想的なオーダリングを行える。

3. 分割指し手生成しやすい

2に関しては、次の順で指し手生成します。

- 駒を捕獲する手（捕獲する駒の価値の降順）

- 駒を捕獲せず、かつ移動先の当方利きが相手利き以上となる手（ただし成れる場合の不成りは後回し）
- 移動先の当方利きが相手利き未満となる置駒移動（ただし成れる場合の不成りは後回し）
- 上で後回しにした、成れる場合の不成り
- 当方利きが相手利き以上であるマスへの駒打ち
- 当方利きが相手利き未満であるマスへの駒打ち

個々の指し手は置駒についてはFlow単位、持駒については(駒の種類×筋)単位で分割生成です。

駒の捕獲の有無や指す位置の利きの大小を考慮した指し手生成順序とすることで、
何年か前にやろうとした、駒の移動を7種類に分類してオーダリングするという蒸気プランに
なんと実装が与えられたことににになると思います。

指し手生成速度は、超シンプルなαβ法の下でCore i7-860 @ 2.80GHzで24万NPS。

（その2）壁打ち式学習法

良い評価関数とは、次の3条件を満たすﾌﾞｯのことだ、と思います。

(1) 水平線をもたらさないこと（無矛盾性（？））
(2) 有利に指し進めるという目的に照らして妥当であること（健全性（？））
(3) 同じ尺度で局面の良し悪しの比較が成立すること（全順序性（？））

(1)は対象の評価関数で探索と着手を繰り返したときの評価値の推移でテストできます。
指し進める中で当方手番の評価値の低下があったら水平線を生じた証拠。

(2)のテストのためには、当方の着手xによって当方が見込んだ最悪の利得が
本当に正しいのか、対戦相手に当方の評価関数が想定していないかもしれない
様々な手を指させて試す必要があります。

(3)は今回非考慮（後述）。

で、(2)のテストの相手役をベイジアンフィルタに務めてもらうというのが壁打ち式学習法です。
¦{KPP}¦個の単語を認識し、2クラスを識別するベイジアンフィルタが
評価関数サイド（当方＝挑戦者）の評価値を下げる手を学習しつつ繰り出します。
（単語やクラスの生起回数だけとはいえ）過去の履歴を「全部」憶えているような学習器を
長期的に騙し通すことはまず不可能。評価関数サイドピンチ…！

（その3）KPPの高速調整
一方評価関数サイドはKPPに対する荷重和という普通の評価関数です。（KPP以外に移行しない理由は後述）
KPPというとTO、NY、NK、NGをKIと同一視したとしても¦{KPP}¦＝1696*(1696+1)＝29万種類あり、
ある探索結果の評価値を変更しようにも、どれを上げ下げすれば良いのか迷ってしまいます。
ここではその選択にもベイジアンフィルタを使います。
すなわち、¦{KPP}¦個の単語を認識し、¦{KPP}×{UP/DOWN}¦クラスを識別するベイジアンフィルタを設けて……
…

嘘です。そんなことをしたら今日日のPCのメモリに乗りません。
そこはさすがに縮小したタイプのベイジアンフィルタを使います。

（その4）ダイナミックアスピレーションサーチ
反復深化する中で、PVの先頭の手（当座の着手候補手）が変化したときの
評価値の変化量のヒストグラムをとり続け、上側と/下側のx％境界点を
初回探索時の窓の上端と下端とします。

これは探索結果の評価値の振れ幅が大きいと成立しませんが、
局面評価基盤に上記工夫を積み重ねたのだから成立してほしい…

■ 前回と同じ点

Futility purningしきい値の動的調整は今回もやります。

■ 前回と違う点（新機軸以外）

場合分け方針は一時転進。KPPの荷重和による局面評価のアラを

非力な軽量評価関数でカバーするのまうまくいかなかった。

Null move purningは探索ルーチンの書き直しに伴い削除。

復活は後日検討。

■ 局面評価をKPPのままとする理由

KPPによる局面Sの評価というのは、BonaPiece全体の集合から

Sに含まれるBonaPieceのサブセットを得るという離散的な選択過程を含むため、

能力の解析は~~できな~~もとい非常に困難ですが、どうも関数近似能力は無いらしい。

しかしながら、現行の探索処理という

1度の探索で限られた範囲の局面同士しか比較されない条件において、

局面の順序を学習するには十分なのではないかという印象をもっています。

（あるKPP xの重みの変更は、xを共有するその他の局面の評価値にも変化をもたらすが

その変化を打ち消すことが他のKPPの重みの調整で成し得ないケースを想定し難い。

および、順序の学習といってもβ値、最善のPV、それ以外、の3種類の評価値が

現実に起こる比較の範囲で降順で並びさえすれば良いので比較的緩いのではないか。）

以上

猫将
アピール文章

2019.03.31

Yuko Yasuga

猫将の特徴

ネコのように自由な棋風！

…というテイで、合法手をランダム生成する
シンプルなプログラムになる予定です。

猫将の概要

開発者　安賀裕子

開発の動機　普段はブックデザイナーをしているのですが、仕事上 JavaScript が出来ると生産性が上がるため、その勉強として

開発の進捗　なるほどわからん（`Φωφ´）

応援要員　**クロちゃん**

19 才（夏に 20 才）のおじいちゃん猫。野良出身で、赤ちゃんの頃に外耳が切れてしまったらしく、片耳です。作業中に飯や添い寝を要求して休憩を取らせたりする役割です。よく寝てよく食べる良い子。真っ黒なので目をつぶると顔がよくわかりません。

棋風 WCSC29アピール文書 2019/03/31

- AlphaZero ベースの強化学習
- 相手に応じて棋風を変えて対局
- WCSC29参加に向けて実装中

使用ライブラリの選定理由

Apery, やねうら王, tanuki-, elmo, dlshogi, python-shogi
の利用を検討しております。
- 合法手生成: Apery
- 教師局面生成: Apery, elmo
- 評価関数: Apery, tanuki-
- Pythonを利用: python-shogi, dlshogi
- 探索: やねうら王
- 規定プロトコルで対局: やねうら王