第 28 回世界コンピュータ将棋選手権参加ソフト

「ねね将棋」アピール文書

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

使用ライブラリ

やねうら王 [1] (ソースコードおよび教師局面) : ユーザ定義エンジンの追加がしやすいため

~~python-shogi [2] (ソースコード) : 通信に用いる python 言語と親和性が良いため~~ （クラウドとの通信切断時に指し手生成を行わせることを予定していましたが、状態管理のミス等で逆に信頼性が下がりそうだったので実装しませんでした）

探索部

AlphaZero [3]等で採用されている MCTS (Monte-Carlo Tree Search)を実装します。USI 通信・合法手の列挙部分まではやねうら王ライブラリを利用しています。

2017 年 11 月の第 5 回将棋電王トーナメント(SDT5)では python 言語でゲーム木を実装していたのですが、速度上のボトルネックとなっていたため今回は C++言語で実装します。探索はシングルスレッドで行います。

定跡として、floodgate の 2017 年の棋譜にて 100 回以上登場した局面それぞれを 100 万回ずつ読ませた状態の置換表を初期値としてロードすることで代用します。探索の初期段階は GPU の並列性を活かしにくいのですが、この方式であれば定跡から外れた局面でも浅い読みの結果が残っているため、ちょっとだけ有利なのではないかと期待しています。

通常探索とは別のスレッドでルート局面からの詰将棋探索を行います。この結果が詰みの場合、通常探索の結果を上書きして詰ませに行きます。実装はやねうら王に搭載されているものを改造して用いています。通常探索の末端からも浅い詰み探索を行うことを試みましたが、NPS が下がってしまうためできませんでした。探索のマルチスレッド化が必要だと思われます。

評価関数

局面の勝率および各指し手の事前確率を出力する Deep Neural Network を深層学習により学習します。

やねうら王プロジェクトで提供されている教師データによる教師あり学習を行います。

速度上の問題で自己対戦による強化学習には至っていません。

評価関数の実行には GPU を用います。SDT5 では 5,000NPS 程度(NVIDIA GTX 1080Ti)で、CPU からのデータ供給側のボトルネックにより GPU の性能を使いきれませんでした。今回はクラウド上のマルチ GPU マシンを有効に活用できるよう、実装を刷新します。

モデル構造

14 層の Convolutional Neural Network です。Residual 構造があり、中間層は 192ch あります。

盤面入力は Ponanza チームの資料[4]を参考にした 86 チャンネル、指し手出力は AlphaZero の資料を参

考にした 139 チャンネルとなっています。
やねうら王の棋譜[5]を用いて教師あり学習しています。局面ごとの指し手および勝敗を回帰します。評価値は使っていません。最適化手法は Momentum SGD です。
学習した結果、探索なしに棋譜の指し手を正解できる確率が 40%程度となりました。この状態で floodgate レート 1700 程度でした。
モデル構造をこれ以上大きくしても正解率が上がりませんでした。ただ、不正解の 60%に悪手が含まれるため、現状が最善かはわかりません。

探索との組み合わせ

探索中に新たに到達した局面すべてについて、GPU 上で評価を行います。1 局面ずつ評価すると極端に性能が悪いので、512 局面のミニバッチにして非同期的に評価します。
Deep Learning フレームワークとして Python で書かれた Chainer を用いており、マルチスレッドが原則できません。マルチ GPU を活用するため、Chainer を用いた評価プロセスを複数立てて、C++で書かれたメインのエンジンプロセスとプロセス間キュー（自作）で接続しました。
GPU は AWS 上の NVIDIA Tesla V100 を用います。モデルを通常の float32（単精度浮動小数点数）から float16（半精度浮動小数点数）にし、Tensor Core を有効にすることで、ランダム入力に対する評価のみのベンチマーク上は 9,000NPS 程度得られました。
GPU を 8 台並列動作させ、探索部と組み合わせた場合標準的に 30,000NPS 程度となります。探索結果の重複が少なくなる、探索時間が長い場合で、最大 60,000NPS 得られるようです。
MCTS は前向き枝刈りと同様の性質があり、NPS が向上しても見えない変化があるようです。対局相手に意外な手を指されて置換表にない局面に飛び込み、一気に評価値が悪くなるという事象がみられます。現状、うまい対策ができていません。

おわりに

SDT5 からは開発言語の変更が主となりますが、大幅な探索速度向上が狙えます。
モデルを大きくしたこともあり、SDT5 から floodgate レートが 1000 以上上がっています。
クラウドに課金して一次予選突破が目標です。

[1] 磯崎元洋 https://github.com/yaneurao/YaneuraOu
[2] 末永匡 https://github.com/gunyarakun/python-shogi
[3] D. Silver et al., Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. http://arxiv.org/abs/1712.01815 (2017)
[4] HEROZ-JAPAN 「Ponanza における強化学習とディープラーニングの応用」 https://www.slideshare.net/HEROZ-JAPAN/ponanza-83900718 (p.19)
[5] やねうら王公式サイト「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」 http://yaneuraou.yaneu.com/2017/11/21/depth10%E3%81%A7%E4%BD%9C%E3%81%A3%E3%81%9F110%E5%84%84%E5%B1%80%E9%9D%A2%E3%81%AE%E6%95%99%E5%B8%AB%E3%83%87%E3%83%BC%E3%82%BF%E3%80%81%E6%9C%9F%E9%96%93%E9%99%90%E5%AE%9A%E3%81%A7%E5%85%AC%E9%96%8B/

2018 年 3 月 7 日（4 月 30 日修正） 日高雅俊 https://github.com/select766