1 Ray について

OSS 版の Ray[1] に以下のような改良を施してあります.

1. 木探索の着手評価に Factorization Machines + Bradley-Terry モデルを利用
2. 木探索部とシミュレーション部に特徴を追加
3. 木探索部, シミュレーション部ともに細かなパラメータ調整
4. Deep Learning 対応 (後述)

2 Deep Learning 対応

第 2 回 AI 竜星戦では, AlphaGo Fan 方式の実装で出場しましたが, 今回は 256 フィルタ 20 ブロックの ResNet を教師あり学習させたネットワークを作りました. いわゆる AlphaZero 系統のプログラムよりは人間に近い着手をするので, 見ていてわかりやすいプログラムになっています. 少しだけ強化学習して, 自己対戦で +100elo くらい強くなっています.

使用している Deep Learning のフレームワークは今は懐かしい Caffe[2] です. 今から囲碁プログラムを頑張って作りたい方には TensorFlow か PyTorch をお勧めします.

ネットワークが大きいので探索速度がすごく遅いのですが, わずかでも高速化できるように Caffe で学習したネットワークからパラメータを取り出して, CuDNN を直接使うようにしています. 小さなネットワークであれば, Caffe 独自の処理が重くなるため, 比較的高速化できているのですが, ネットワークの規模が非常に大きいので, 2 3 割程度しか高速化できていません.

3 ゼロからの強化学習 (間に合えば)

上述したのはベースラインのプログラムで, 現在ゼロからの強化学習にチャレンジしています. もしこちらがそれなりの仕上がりになれば, こちらのネットワークを採用するつもりです. KataGo[3] を参考に, 細かい工夫を取り込んだ実装をしています. ネットワークの出力は

- Player's Policy
- Opponent's Policy
- Player's Value
- Ownership

の 4 種類あります. Opponent's Policy は対局中に使われない出力ですが, 強化学習では KataGo で, 教師あり学習では DarkForest[4] で採用されており, それなりの効果があるので, そのまま採用しています.

4 その他

開発の助けになる機能をいくつか実装しています.

- 対局時の読み等の情報を埋め込んだ SGF ファイル出力機能

- Gogui Analyze Command をいくつか実装

対局の強さには全く影響ないですが, Deep Learning が台頭する以前から開発の手助けになっているので, 実装等が気になる方は遠慮なく聞いてください.

参考文献

[1] Ray, https://github.com/kobanium/Ray
[2] Caffe, https://github.com/BVLC/caffe
[3] KataGo, https://github.com/lightvector/KataGo
[4] DarkForest, https://github.com/facebookresearch/darkforestGo