1 Ray について

基本的には Ray(https://github.com/kobanium/Ray) に改造を施したものになりますが, 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 Latent Factor Ranking ではなく, Factorization Machines と Bradley-Terry モデルをうまいこと組み合わせたものを利用
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

Factorization Machines と Bradley-Terry モデルをうまいこと組み合わせた手法については私が考えたものではなく, チーム DeepZen の代表で, 第 2 回 UEC 杯準優勝プログラム" 不動碁" の開発者である加藤英樹さんから教えていただきました. これら改良のおかげで, OSS 版の Ray だと 19 路 1 手 1000 プレイアウトで GNU Go と互角のところ, 最新の Ray だと 1 手 500 プレイアウトで互角の強さになります. とは言え, 既に Deep Learning に対応しているので, おまけ程度な感じです.

2 Deep Learning 対応

Ray は Policy Network と Value Network を利用した APV-MCTS(Asynchronous Policy and Value Monte-Carlo Tree Search) を実装しています. 諸々の学習のさせ方については,

1. KGS の高段者の置き石のない対局 (互先か定先) を教師データにして Policy Network を学習
2. 1 で作った Policy Network 同士を対局させて, 対局結果から Policy Network を強化学習
3. 1 と 2 で出来上がった 2 つの Policy Network を使って自己対局
4. 3 で生成された棋譜と KGS の高段者の互先の棋譜を教師データにして Value Network を学習

と概ね, AlphaGo Lee 版の実装になっています. 強さとしては, 第 10 回 UEC 杯の時の Rayn より強く, 最新の Rn より弱いと言った感じです. Deep Learning のフレームワークは Caffe を利用しています. 強さとしては微妙なので, 大会までに他のプログラム (Leela Zero, Elf Open Go, minigo など) からネットワークを引っ張って来られないかと考えていますが, 時間がないので, 考えるだけで実行に移せない気がしています.

3 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.