真やねうら王

やねうら王のオープンソース化による影響

Apery(2015)に倣い、やねうら王もGitHubで公開することにした。

https://github.com/yaneurao/YaneuraOu

これにより、いつくかバグの指摘をもらえたりして助かっている。また、やねうら王をベースとして開発する開発者が増えた結果、開発ノウハウの共有などが図れて開発が捗っている。

自己対戦の自動化

『技巧』(2015)は、1手0.1秒での連続対局による勝率を見ることで改良前・改良後の比較を行なうという方法を確立した。短い持ち時間での自己対戦は長い持ち時間での自己対戦と性質が異なるところも多々あるが、特定の条件下においては、長い時間での勝率とさほど変わらない。そこで、真やねうら王もこの仕組みを取り入れることにした。真やねうら王では、1手0.1秒1万局対局の結果を基準としている。

実験用PCにXeon PCを購入

電気代のことを考えると、どうもmany coreで低クロックのCPUのほうが結局は安上がりのようである。そこで、Xeon 2698(20コア)×Dual×5台を購入することにした。これにより、物理200コア、論理400コアの実験環境が整った。

Jenkinsの導入

「プログラムの改良→自己対戦→結果の観測」というのは、自己対戦に一定の時間を要するため、このTAT(ターンアラウンドタイム)自体を短くするのは難しい。また開発者が寝ている間にPCが休んでいるのももったいない。ジョブを管理し、効率的にジョブを24時間、寿司詰めにして実行するためにはCI(continuous integration : 継続的インテグレーション)ツールが必要不可欠である。このため、真やねうら王の開発にCIツールの代表格であるJenkinsを導入し、24時間効率的にジョブを実行できるようにした。

hyperoptを用いた探索パラメーターの自動チューニング

また、自己対戦によりチューニングできるパラメーターは完全に自動化してチューニングするようにした。これにより、探索パラメーターのチューニングは人手を介さずに出来るようになった。しかし、探索パラメーターは長い持ち時間のときにしか意味がないものもあり、1手10秒程度かけて対局させないと結果を信頼できない。このため1万対局させるには、膨大な計算資源を必要とする。自動化されているとは言え、あまり何度も探索パラメーターの調整を行なうことは出来ない。そこで、真やねうら王では『tanuki-』(2015)の手法に倣い、hyperoptを用いる。これは、Tree-structured Parzen Estimator Approach(TPE)などを用いて効率的にパラメーターの最適化を行なうフレームワークである。

評価関数テーブルに共有メモリを使用する

新しく導入したPCは物理40コア論理80コアあるので、自己対戦として並列80対局させるのだが、KPP+手番型の評価関数は、評価関数テーブル自体が1GB程度あるのでそれだけで80対局×2ソフト = 160GBものメモリが必要になる。サーバー機用のメモリは高額であり、メモリが惜しいので、Windowsのmemory mapped fileを用いて、共有メモリ上に評価関数テーブルを展開するようにした。何気ない改良のようで、効果は絶大であり、これにより、並列80対局が実現できるようになった。今後、many core化が進むと、この技術はコンピューター将棋開発者の間で標準的な技術として取り入れられるだろう。

80億局面からの学習

『ponanza』(2015)は、自ら棋譜を生成し80億局面から学習しているそうで、その手法に倣うことにした。Aperyも今回、基本的にはこの手法をとっているらしい。ただ、機械学習の手法は様々あるので(目的関数の違いや、最適化手法の違い、学習率の違いetc…)話はそう簡単ではなく、それぞれのソフトにおいて同じような結果が出るとは一概に言えない。

浅い探索の評価値を深い探索の評価値に近づける。

これは、『NDF』(2014,2015)の手法。強化学習の一種だとみなせる。

ただし、真やねうら王では、評価値から推定される勝率に変換して、「浅い探索の評価値から推定される勝率を、深い探索の評価値から推定される勝率」に近づけるというように目的関数を設定している。

また、交差エントロピーを用いた目的関数なども試している。どの目的関数やどの最適化手法が優れているかは一概に言えない意味があり、色々試している。

ponanza化するApery、やねうら王

80億局面から学習させると棋風がponanza化するようである。真やねうら王でも、ponanzaが好んで指す、角交換型4八金戦法などを好んで指すようになった。

「depth6(6手読み)で80億局面を生成→そこからSGDやAdaGradで学習→自己対戦で強くなっているかのテスト」というのを1イテレーションとして、強さがある程度飽和するまでこれを繰り返した。数回目のイテレーションにおいても、棋力はR20程度上がるようである。(イテレーションが進むにつれ、徐々に上がる量は減衰する) このイテレーションを『雑巾絞り』と呼ぶ。雑巾はどこまで絞る余地があるのかは不明である。depth8や10でもやってみたいが、計算資源が足りない。

ゼロベクトルからの評価関数パラメーターの学習

次に実験として80億局面が生成できたので、評価関数パラメーターをゼロベクトルから学習させてみた。これは面白いことに簡単に以前の評価関数より強いものが出来た。どうも、以前の評価関数パラメーターから追加学習させることには、(雑巾絞りの早い段階においては)さほど意味がないようである。

人間の棋譜なしでの評価関数パラメーターの学習

最初にプロの棋譜からBonanzaメソッドで学習させた評価関数パラメーターを用いて、棋譜を生成していたのだが、以上の実験からすると、そのことにはあまり意味がなさそうである。結局、『雑巾絞り』をすることで3駒関係で表現できる限界に近づき、それがponanza風の棋風なのであろう。つまり、ある強さのソフトさえ最初にあれば、『雑巾絞り』には十分だ。

そこで、評価関数パラメーターをゼロ初期化して、駒割り(駒得)だけの評価関数で局面を80億生成した。ただし、depth 6だと生成に異様に時間がかかるため、depth 3とした。

「depth　3で80億局面を生成→そこからSGDで追加学習」というのを繰り返すことにより、やねうら王(2015)と同じ程度の強さの評価関数パラメーターが得られた。真やねうら王では、ここから追加学習させていく予定である。

まとめ

従来、『やねうら王』(2014,2015)では学習時に評価関数パラメーターの次元下げを行なっていた。『NDF』(2013)の相対KPP/KKPによる次元下げ、『AWAKE』(2014)の利きによる次元下げである。

このためソースコードが大変複雑化し、コードを書くのに長い時間を必要とした。ところが80億局面から学習させるようになってからは、次元下げが一切不要となったため、評価関数の形を改良することが容易になった。

教師局面である80億局面の生成には時間がかかるが、そこから学習させるときには浅い探索しかしないので40コアPCであれば、1日程度で完了する。また実験の結果から、追加学習には初期の『雑巾絞り』においてはあまり意味がないことがわかっているので、評価関数の形を大胆に変更してもそこからさほど『雑巾絞り』を繰り返す必要はなさそうである。

そこそこの計算資源(物理200コア)、CIツールの導入、hyperoptを用いた探索パラメーターの最適化、共有メモリを用いた評価関数、80億局面からの学習など、役者は一通り揃ったという感じだが、インフラ整備はまだまだ必要そうである。

いつになったら私は将棋ソフトの開発に着手できるのであろうか…。そして、去年からupしていない賞金額に心を折られることなくこのあと開発に立ち向かえるのだろうか…。ちなみにいま、このPR文書を書くのに開発モチベーションの80%ぐらいを消費した。そして今月の電気代が10万円を超えそうだ。俺はもう駄目かも知れん。ガチに開発すればするほど貧乏になっていく…。なんなのだ…。何故、このような罰ゲームに毎年自分が参加しているのか…。わからない。誰か教えて欲しい。