「技巧」アピール文書

2023 年 3 月 31 日（5 月 5 日加筆修正）

出村　洋介

1. Rust と Python で全面的な書き直し

以前の「技巧」は C++を開発言語としていましたが、将棋プログラムの複雑化に伴い、ポインタ絡みの不具合などの発見が難しくなっているといった困難を感じていました。

そこで、今回は思い切って C++版の開発を停止し、安全性が高いと評判の Rust で全面的に書き直しました。また、学習部は GPU を安定して動かせる Python の PyTorch[1]を利用しています。Rust や Python の特徴を活かして、以下のような点を工夫しています。

①危険性の高いコード（ポインタやグローバル変数など）の利用を最小化

②Rust のユニットテスト機能の活用（Rust ではユニットテストが標準装備）

③Rust と Python 間では、NumPy 形式で安全にデータを受け渡し（PyO3 を利用[2]）

2. 自動ベクトル化の採用（Apple M1/M2 への対応）

従来の「技巧」は、Intel や AMD の CPU（x86-64）向けの intrinsic 関数を使って SIMD 演算を行うことで、Bitboard などの高速化が図られていました。そのため、従来の「技巧」のままでは、最近購入した Apple 社の M1 Mac で動作しないという問題がありました。

そこで、今回の「技巧」では、intrinsic 関数の利用を原則廃止し、自動ベクトル化のアプローチを新たに採用しています。これにより、Intel や AMD の CPU に加えて、Apple の M1/M2 CPU でも動作するようになりました。

自動ベクトル化（auto-vectorization）は、コンパイラが自動的に for ループ等を SIMD 演算に置き換えて高速化を図る仕組みです。コンパイラの自動ベクトル化では、プログラマ側で CPU ごとに個別のコードを書かなくても、SIMD 演算による高速化ができる利点があります。自動ベクトル化はコンパイラの対応が進んでおり、Rust の LLVM コンパイラ[3]だけでなく、C++の GCC[4]などでも利用可能です。

3. AlphaZero と NNUE の組合せをテスト開発中

今回の「技巧」では、AlphaZero の手法[1]を参考に、新たに深層学習の導入をテスト開発中です。

ところが、AlphaZero の深層強化学習の手法は、学習に要する計算量が大きいことが知られています。実際、Silver ら[1]によると、将棋で AlphaZero の学習を 1 回行うには、TPU

---

[1] https://pytorch.org.

[2] https://github.com/PyO3/pyo3.

[3] https://llvm.org/docs/Vectorizers.html.

[4] https://gcc.gnu.org/projects/tree-ssa/vectorization.html.

（Tensor Processing Unit）5000 台を用いて 12 時間を要したとされています。単純計算では、TPU 1 台で 7 年ほどかかるという膨大な計算量です。

そこで、今回の「技巧」では、AlphaZero の学習手法を参考にしつつ、AlphaZero よりも学習に要する計算時間を短縮することを目指して開発しています。

アイデアとしては、AlphaZero の深層強化学習に、軽量・高速な NNUE (Efficiently Updatable Neural-Network-based Evaluation Functions)[2]をうまく組み合わせることで、AlphaZero よりも学習に要する時間を減らすことができるのではと考えて、試行錯誤しながら開発を進めています。

3.5 （5 月 5 日追記）

開発期間の関係で、今回は NNUE の評価関数は利用せず、深層学習の評価関数とモンテカルロ木探索の組合せでの参加となりました。今回の主な工夫点は以下のとおりです。

・既存のデータセットを利用せず、自己対局のみによる強化学習（1000 万棋譜程度）
・比較的少ない探索量での強化学習[3]（今回は 1 局面あたり 16 シミュレーション）
・将棋固有の入力特徴の追加（影の利き、ピン、王手している駒なども考慮）

4. 使用ライブラリについて

コンピュータ将棋のライブラリは使用せずに、フルスクラッチで開発しています。

5. おわりに

「技巧」のアピール文書をご覧いただき、ありがとうございます。

2017 年に参加して以来久しぶりの参加となるため、初心に戻って開発していきたいと考えています。

参考文献

[1] D. Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, vol. 362, no. 6419, pp. 1140-1144, 2018.

[2] 那須悠. 高速に差分計算可能なニューラルネットワーク型将棋評価関数. 第 28 回世界コンピュータ将棋選手権アピール文書, 2018.

[3] I. Danihelka et al. Policy improvement by planning with Gumbel. *International Conference on Learning Representations*, 2022.