

第 29 回世界コンピュータ将棋選手権

dlshogi アピール文章

山岡忠夫
2019/5/5 更新

※下線部分は、第 28 回世界コンピュータ選手権からの差分を示す。

1 特徴

- ディープラーニングを使用
- 指し手を予測する Policy Network
- 局面の勝率を予測する Value Network
- 入力特徴にドメイン知識を積極的に活用
- モンテカルロ木探索
- GPU によるバッチ処理に適した並列化
- 自己対局による強化学習
- 詰み探索結果を報酬とした強化学習
- 既存将棋プログラムの自己対局データを使った事前学習
- REINFORCE アルゴリズムによる Policy Network の学習
- ブートストラップ法による Value Network の学習
- マルチタスク学習
- 末端ノードでの短手順の詰み探索
- ルートノードでの長手順の詰み探索
- 序盤局面の事前探索 (定跡化)
- マルチ GPU 対応
- CUDA、cuDNN を直接使用
- GPU ごとに異なるモデルの読み込み
- Optuna による探索パラメータの最適化
- 確率的な Ponder
- レイヤー融合

2 使用ライブラリ

- elmo¹ (Commits on May 29, 2017)
→事前学習用の訓練データ生成に使用
- Apery² (WCSC28)

¹ https://github.com/mk-takizawa/elmo_for_learn

→局面管理、合法手生成のために使用

2.1 ライブラリの選定理由

本プログラムは、将棋におけるディープラーニングの適用を検証することを目的としており、学習局面生成、局面管理、合法手生成については、使用可能なオープンソースがあれば使用の方針である。そのため、学習局面を圧縮形式(hcpe)で生成する機能を備えている `elmo`、及び、合法手生成を高速に行える `Apery` を選定した。

3 各特長の具体的な詳細（独自性のアピール）

3.1 ディープラーニングを使用

DNN(Deep Neural Network)を使用して指し手を生成する。

従来の探索アルゴリズム(α β 法)、評価関数(3 駒関係)は使用していない。

3.2 Policy Network

局面の遷移確率を Policy Network を使用して計算する。

Policy Network の構成には、Wide Residual Network³を使用した。

入力の畳み込み 1 層と、ResNet 10 ブロック(畳み込み 2 層で構成)と出力層の合計 22 の畳み込み層で構成した。フィルターサイズは 3 (入力層の持ち駒のチャンネルのみ 1)、フィルター数は 192 とした。

3.3 Value Network

局面の勝率を Value Network を使用して計算する。

Value Network は、Policy Network と出力層以外同じ構成で、出力層に全結合層をつなげ、シグモイド関数で勝率を出力する。

3.4 入力特徴にドメイン知識を積極的に活用

Alpha Zero では、入力特徴に呼吸点のような囲碁の知識を用いずに盤面の石の配置と履歴局面のみを入力特徴とすることで、ドメイン知識なしでも人間を上回ることが示された。しかし、その代償として、入力特徴にドメイン知識を活用した AlphaGo Lee/Master に比べて倍のネットワークの層数が必要になっている。AlphaGo Zero の論文の Figure 3 によると、ネットワーク層数が同一のバージョンでは Master を上回る前にレーティングが飽和している。

強い将棋ソフトを作るという目的であれば、積極的にドメイン知識を活用した方が計算リソースを省力化できると考えられる。

² <https://github.com/HiraokaTakuya/apery>

³ <https://arxiv.org/abs/1605.07146>

そのため、本ソフトでは、入力特徴に盤面の駒の配置の他に、利き数と王手がかかっているかという情報を加えている。それらの特徴量が学習時間を短縮する上で、有効であることは実験によって確かめている。

3.5 モンテカルロ木探索

対局時の指し手生成には、Policy Network と Value Network を活用したモンテカルロ木探索を使用する。

ノードを選択する方策に、Policy Network による遷移確率をボーナス項に使用した PUCT アルゴリズムを使用する。PUCT アルゴリズムは、AlphaZero の論文⁴の疑似コードに記述された式を使用した。

また、末端ノードでの価値の評価に、Value Network で計算した勝率を使用する。

通常のモンテカルロ木探索では、末端ノードから複数回終局までプレイアウトを行った結果（勝率）を報酬とするが、将棋でランダムなプレイアウトは有効ではないため、プレイアウトを行わず Value Network の値を使用する。

3.6 GPU によるバッチ処理に適した並列化

複数回のシミュレーションを順番に実行した後、それぞれのシミュレーションの末端ノードの評価をまとめて GPU でバッチ処理する。その後、評価結果をそれぞれのシミュレーションが辿ったノードにバックアップする。以上を一つのスレッドで行うことで、マルチスレッドによる実装で課題となる GPU の計算後にスレッドが再開する際にリソース競合が起きる問題（大群の問題）を回避する。

GPU で計算中は、CPU が空くため、同じ処理を行うスレッドをもう一つ並列で実行する。2つのスレッドが相互に CPU と GPU を利用するため、利用効率が高い処理が可能となる。

3.7 自己対局による強化学習

事前学習を行ったモデルから開始して、AlphaZero⁵と同様の方式で強化学習を行う。自己対局により教師局面を生成し、その教師局面を学習したモデルで、再び教師局面を生成するというサイクルを繰り返すことでモデルを成長させる。

2018年の大会で使用した elmo で生成した教師局面で収束するまで学習したモデルに比べて、自己対局による強化学習によって有意に強くすることができた。

3.8 詰み探索結果を報酬とした強化学習

自己対局時に終局まで対局を行うと、モンテカルロ木探索の特性上、詰むまでの手順が長くなる傾向がある。勝率予測に一定の閾値を設けることで、終局する前に勝敗を判定するこ

⁴ <http://science.sciencemag.org/content/362/6419/1140>

⁵ <https://arxiv.org/abs/1712.01815>

とで対局時間を短縮できるが、モデルの精度が低い場合は誤差が大きいため、学習精度に影響する。

この課題の対策として、df-pnによる高速な長手数の手み探索の結果を報酬とした。単純にすべての局面で手み探索を行うと、自己対局の実行速度が大幅に落ちてしまう。自己対局は複数エージェントに並列で対局を行わせ、各エージェントからの手み探索の要求をキューに溜めて、手み探索専用スレッドで処理するようにした。エージェントがGPUの計算待ちの間に手み探索が完了する。エージェントが探索している局面は別々のため、時間のかかる手み探索の要求が集中することは少ない。これにより自己対局の速度を大幅に落とすことなく長手数の手み探索を行えるようになった。

3.9 既存将棋プログラムの自己対局データを使った事前学習

本プログラムを使用して、Alpha Zeroと同様に、ランダムに初期化されたモデルから強化学習を行うことも可能だが、使用可能なマシンリソースが足りないため、スクラッチからの学習は行わず、既存将棋プログラムの自己対局データを教師データとして、教師あり学習でモデルの事前学習を行う。

教師データには、elmoで生成した自己対局データを使用した。

3.10 REINFORCE アルゴリズムによる Policy Network の学習

単純に自己対局の指し手を学習するのではなく、学習局面の価値と勝敗データと関連付けて学習を行う。良い局面から負けになった手は、悪手として負の報酬を与え、悪い局面から勝ちになった手は善手として正の報酬を与える。学習アルゴリズムには、AlphaGoの論文に掲載されているREINFORCEアルゴリズムを使用した。

3.11 ブートストラップ法による Value Network の学習

Value Networkの学習の損失関数は、勝敗を教師データとした交差エントロピーと、探索結果の評価値を教師データとした交差エントロピーの和とした。

このように、本来の報酬（勝敗）とは別の推定量（探索結果の評価値）を用いてパラメータを更新する手法をブートストラップという。

経験的にブートストラップ手法は、非ブートストラップ手法より性能が良いことが知られている。

3.12 マルチタスク学習

Policy NetworkとValue Networkのネットワーク構成が同じ層を共通化し、出力層を分けることで、同時に学習を行う。

関連する複数のタスクを同時に学習することをマルチタスク学習という。タスク間に関連がある場合、単独で学習するよりも精度が向上する。

また、対局時に Policy Network と Value Network を同時に計算できるため、高速化の効果もある。

3.13 末端ノードでの短手順の詰み探索

モンテカルロ木探索の末端ノードで、7 手の詰み探索を行い、詰みの局面を正しく評価できるようにする。並列化の方式により、GPU で計算中の CPU が空いた時間に詰み探索を行うため、探索速度が落ちることはない。

3.14 ルートノードでの長手数詰みの探索

モンテカルロ木探索は最善手よりも安全な手を選ぶ傾向があるため詰みのある局面で駒得になるような手を選ぶことがある。

対策として、詰み探索を専用スレッドで行い、詰みが見つかった場合はその手を指すようにする。

詰み探索は、df-pn アルゴリズムを使って実装した。優越関係、証明駒、反証明駒、先端ノードでの 3 手詰めルーチンにより高速化を行っている。

3.15 序盤局面の事前探索（定跡化）

出現頻度の高い序盤局面は、対局時に探索しなくても、事前に探索を行い定跡化しておくことができる。また、事前に探索することで、対局時よりも探索に時間をかけることができる。

ゲーム木は指数関数的に広がるため、固定の手数までの定跡を作成するよりも、有望な手順を選択的に定跡に追加する方が良い。自分が指す手は、1 つ局面につき最善手を 1 手（または数手）登録し、それに対する応手は、公開されている定跡や棋譜の統計情報を使って確率的に選択する。その手に対して、また最善手を 1 手（または数手）登録する。この手順により、頻度の高い局面については深い手順まで、頻度の低い局面については短い手順の定跡を作成することができる。

3.16 マルチ GPU 対応

複数枚の GPU を使いニューラルネットワークの推論を分散処理する。

「3.6 GPU によるバッチ処理に適した並列化」の方式により、GPU ごとに 2 つの探索スレッドを割り当てることで、GPU を増やすことでスケールアウトすることができる。ノードの情報は、すべてのスレッドで共有する。

アムダールの法則により、増やせる GPU には上限があるが、確認できている範囲で 4GPU まで線形で探索速度を上げることができている。

3.17 CUDA、cuDNN を直接使用

モデルの学習にはディープラーニングフレームワークとして Chainer を使用しているが、対局プログラムには、ディープラーニングフレームワークを用いず CUDA、cuDNN を直接使用する。Chainer で学習したモデルを読み込み、推論を行う処理をスクラッチで開発した。

高速化と対局の実行環境にディープラーニングフレームワークの環境構築を不要とすることを目的とする。

3.18 GPU ごとに異なるモデルの読み込み

モデルごとに誤る確率が独立である場合、複数モデルが同時に誤る確率は、単一のモデルを使用する場合より低くなる。GPU ごとに異なるモデルを読み込むことで、探索の精度の向上が期待できる。

3.19 Optuna による探索パラメータの最適化

PFNにより公開された Optuna⁶を使用して、モンテカルロ木探索の探索パラメータ (PUCT の定数、方策の温度パラメータ) を最適化した。

Optuna は、主にニューラルネットワークの学習のハイパーパラメータを最適化する目的で利用されるが、将棋エンジン同士の連続対局の勝率を目的関数として、探索パラメータの最適化に使えるようにするスクリプト⁷を開発した。Optuna の枝刈り機能により、少ない対局数で収束させることができる。

3.20 確率的な Ponder

モンテカルロ木探索は確率的にゲーム木を成長させる。その特性を活かして、相手が思考中に、相手局面からモンテカルロ木探索を行うことで、確率的に相手の手を予測して探索を行うことができる。予測手1手のみを Ponder の対象とするよりも、効率のよい Ponder が実現できる。

3.21 レイヤー融合

BatchNormalization と畳み込み層のフィルターの行列を事前に計算して、畳み込み層のフィルターに BatchNormalization の演算を含める。これにより、推論時のニューラルネットワークの計算量を減らすことができる。

⁶ <https://optuna.org/>

⁷

https://github.com/TadaoYamaoka/DeepLearningShogi/blob/master/utils/mcts_params_optimizer.py

4 学習について

4.1 事前学習

- 事前学習データ：elmo(wesc27)で深さ 8 で生成した 4.9 億局面

4.2 自己対局による強化学習

4.2.1 学習データ、パラメータ

- ミニバッチサイズ：64~2048 (段階的に増やす)
- 学習アルゴリズム：Momentum SGD (学習率 0.01→0.001、慣性係数 0.9)
- 強化学習 1 サイクルで生成する局面：250~500 万局面
- 強化学習 1 サイクルで学習する局面：直近 10 サイクル分
- 強化学習のサイクル数：179

4.2.2 学習結果

2017 年~2018 年 6 月の floodate のレート 3500 以上の棋譜に対する損失と一致率で評価

【事前学習を行ったモデル】

- Policy Network 損失(交差エントロピー)：0.933、一致率：40.9%
- Value Network 損失(交差エントロピー)：0.587、一致率：68.8%

【自己対局による強化学習を行ったモデル】

- Policy Network 損失(交差エントロピー)：1.122、一致率：44.0%
- Value Network 損失(交差エントロピー)：0.526、一致率：71.7%

以上