第 28 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
2018 年 5 月 1 日 更新

※下線部分は、第 5 回将棋電王トーナメントからの差分を示す。

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活用
- モンテカルロ木探索
- 並列化
- 自己対局による強化学習
- 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
- 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

2　使用ライブラリ

- elmo[1] (Commits on May 29, 2017)
- Apery(elmo の派生元) (14 Apr, 2017 commit:e3eb33ffa6aa840765d2e2efdacf1c618528a3be)

※学習データ生成、局面管理、合法手生成のために部分的に使用

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する方針である。そのため、学習局面を圧縮形式(hcpe)で生成する機能と読み込む機能

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

を備えており、合法手生成を高速に行える elmo(派生元 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 Network[2]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（入力層の持ち駒の面のみ 1)、フィルター枚数は 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。

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。

[2] https://arxiv.org/abs/1605.07146

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、AlphaGo の論文[3]に掲載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、末端ノードからプレイアウトを行った結果（勝敗）を報酬とするが、プレイアウトを行わず Value Network の値を使用する。

3.6 並列化

複数スレッドで並列化を行う。並列化の方式には、スレッド間でゲーム木のノード情報を共有するツリー並列化を採用する。

モンテカルロ木探索は、並列化が容易だが、Policy Network と Value Network を計算するための GPU の数以上の並列化を行うと GPU の使用で競合が発生する。

GPU は複数の計算要求をバッチで処理することが可能であるため、各スレッドからの要求をキューイングして、専用スレッドでバッチ処理することで競合を回避する。

3.7 自己対局による強化学習

Alpha Zero[4]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。

※教師ありより強くなっていないため、大会では使用しない

3.8 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する。

3.9 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] https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf
[4] https://arxiv.org/abs/1712.01815

3.10 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。

3.11 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。

3.12 詰み探索

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手を抜くことがある。

対策として、詰み専用の探索を行い、詰みの場合はその手を指す。

また、モンテカルロ木探索の末端ノードでも、数手の詰み探索を行い、詰みの局面を評価できるようする。Policy Network と Value Network の計算中に、CPU が待ち状態の間に詰み探索を行うため、探索速度が落ちることはない。

3.13 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。

定跡データには、局面に対して複数の手を記録し、それぞれの手に探索時の訪問回数を記録しておく。対局時に定跡を使用する際は、訪問回数に応じた確率で手を選択することで、ゲームの進行が固定されないようにする。

3.14 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。

GPU ごとにキューを処理する専用スレッドを割り当てる。キューを処理するスレッドに対

して、複数のモンテカルロ木探索を行うスレッドを割り当てる。ゲーム木のノード情報は、異なる GPU に割り当てた探索スレッド間で共有する。

3.15 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。

3.16 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、探索の精度を向上が期待できる。

4 学習について

4.1 自己対局による強化学習

4.1.1 学習データ、パラメータ

- 事前学習データ：elmo(wcsc27)で深さ 8 で生成した 4.9 億局面
- ミニバッチサイズ：64
- 学習アルゴリズム：Momentum SGD（学習率 0.01、慣性係数 0.9）
- 強化学習 1 サイクルで生成する局面：500 万局面
- 強化学習のイテレーション数：18（4 月末時点）

4.1.2 学習結果

【事前学習を行ったモデル】

- Policy Network の一致率：45.6%
- Value Network の一致率：78.1%

※ 既存将棋プログラムの自己対局データを増やすことでさらに一致率を上げられるが、Alpha Zero 方式の強化学習によって成果を上げたいため、収束する前のモデルから開始した。

4 月末時点で、18 サイクルの強化学習を実施した結果、学習開始モデルに対して、1 手 3 秒 50 回対局で勝率 81%となり、有意に強くすることができた。しかし、教師ありで収束するまで学習したモデルよりまだ弱いため、大会では強化学習したモデルは使用しないことにする。

4.2　教師ありによる学習

4.2.1 学習データ、パラメータ

- elmo(wcsc27)で深さ 8 で生成した 11 億局面
- ミニバッチサイズ：64
- 学習アルゴリズム：Momentum SGD（学習率 0.01～0.0001、慣性係数 0.9）

4.2.2 学習結果

- Policy Network の一致率：46.1%
- Value Network の一致率：78.1%

以上