第 33 回世界コンピュータ将棋選手権
dlshogi with HEROZ 詳細アピール文章

山岡忠夫
加納邦彦
大森悠平
2023/5/19

1　独自に工夫した点

1.1　自動定跡作成

昨年の第 3 回世界将棋 AI 電竜戦で水匠が優勝した要因となった先手番角換わり定跡に後手番でどう対策するかが今大会の課題であった。

まず、角換わり先後同型腰掛け銀の 37 手目基本図からがコンピュータ将棋同士の対局で先手必勝であるかを検証した。dlshogi の候補手に加えて相手番の指す手に水匠 5 の指し手も考慮して、定跡を 1 か月近くかけて自動作成したところ、後手がどのように指しても先手番が評価値 400 以上になることがわかった。そのため、角換わりの基本図は回避が必要と判断した。

今回考案した新しい定跡作成の手法で、自動的に角換わりの基本図を回避することを目指していたが、技術的な課題の解決が間に合わなかったため、手動で後手からの角交換を負けとして扱った上で、自動で定跡を作成した。

その結果、二次予選第 4 回戦で先手やねうら王(水匠定跡搭載)に対して、後手番で、8 手目 1 四歩というこれまでほとんど指されていない手で後手番で引き分けることができた。

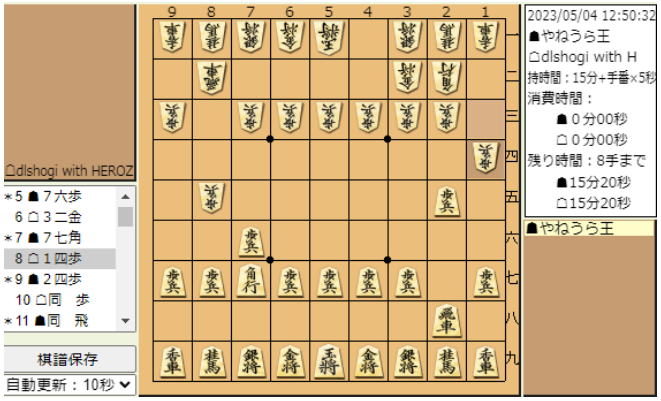

図 1　8 手目後手 1 四歩

1.2　モデル精度向上

前大会の 20 ブロック 256 フィルタのモデルに対して、今大会では ResNet 30 ブロック 384 フィルタというパラメータ数が多いモデルを学習した。それにより精度が向上しており、同一持ち時間での棋力も向上している。

2　開発動機

2016 年 3 月に行われた AlphaGo とイ・セドル九段の対局で、AlphaGo が従来と異なるディープラーニングという手法で勝利したことに衝撃を受けた。囲碁の盤面を画像として入力して指し手を予測するという仕組みを理解したいと考えて、論文を読み AlphaGo のクローンの実装を行った。プロの棋譜を学習して対局するところまで実装できたが、囲碁 AI は、すでにアメリカ、中国の巨大企業も取り組んでいたため、個人で囲碁 AI を開発するモチベーションは続かなかった。将棋 AI では、まだディープラーニングの手法が実験レベルでしか試されていなかったため、自分が開発する意義があると考えて取り組むことにした。

3　開発過程

2017 年からはじめは教師あり学習でのモデル学習から始め、PV-MCTS の実装をしたことで、GPSFish に勝てることできたため、ディープラーニングの手法が将棋 AI でも有効であることに確信が持てた。その後 AlphaZero が発表され、将棋でもトップレベルの強さにできることが報告された。AlphaZero は、膨大な計算リソースで実現されていたため、工夫を行うことで個人レベルの計算リソースでも強くできることを目標にして改良を続けた。

今大会では、HROZ の将棋 AI 開発者のチームで参加した。社内の計算リソースを使って強化学習を行った。チームメンバと共に、戦型・定跡といった大会に向けた作戦を準備した。

4　実験結果

第 3 回世界将棋 AI 電竜戦のモデルからレーティングが 40 程度向上している。

また、floodgate の 2017 年から 2018 年 6 月までの棋譜を元に、レーティング双方 3500 以上、評価値 3000 まで、千日手、最大手数を除外という条件で作成したテストデータに対する精度は、以下の通り。

	方策正解率	価値正解率
今大会のモデル	55.85%	77.65%

5　追試可能か

dlshogi の探索部、学習部のソースコードは GitHub[1]で公開している。学習方法については、ブログ記事[2]や書籍「強い将棋ソフトの創りかた[3]」で解説している。dlshogi の学習に使用しているデータも書籍に付属している。それらを元に、ある程度の追記を行うことは可能である。

[1] https://github.com/TadaoYamaoka/DeepLearningShogi
[2] https://tadaoyamaoka.hatenablog.com/
[3] https://honto.jp/netstore/pd-book_31311287.html

やねうら王チーム WCSC33 終了後の PR 文書

・独自に工夫した点について

１．leaf node での df-pn の効率化について（ｂｙ やねうらお）

選手権前には PR 文書にも書いたように leaf node での df-pn の効率化をしようと思っていたのだが、簡単な予備実験によるとどうも効果が薄いようなので、わりと実装が大変だということもあり、今回は取り組むのを断念した。引き続き、やっていきたい。

２．定跡作成の仕組み（当該項目における文責：たややん）

今回のやねうら王においては、以下のような定跡作成の仕組みを採用しました(2022 年末に行われた第 3 回電竜戦本戦における定跡作成手法をブラッシュアップしました)。

⑴ 指定局面から連続対局させ、指し手を全て定跡として登録する。

⑵ 任意の局面が定跡に登録されていた場合、ミニマックス法で定跡データ内を探索し、勝ちの枝があるか、負けの枝しかないか判別する。

⑶ 勝ちの枝があれば、その手を指し、負けの枝しかない、又は定跡に登録されていない局面であれば、探索エンジンで思考させる（探索エンジンにどれだけ思考させるかは、対局開始時にランダム（水匠 100 万ノードから 1 億ノードの範囲内）に決定する)。

このような仕組みを採れば、連勝を続ける限り、勝ちの枝は採用され続け、ノード数の異なる探索エンジンによって多数回その指し手が本当に勝ちか否かが挑戦され続けることになるため、定跡の精度は上がり続けます。

このような定跡作成ルーチンによって、作成された定跡は、2000 局面に満たない局面数しか登録されていないにもかかわらず、角換わり 37 手目基本図
(lr5nl/3g1kg2/2n1ppsp1/p1pps1p1p/1p5P1/P1PPSPP1P/1PS1P1N2/2GK1G3/LN5RL w Bb 38)
から後手番水匠に対して 2000 連勝以上できるような定跡となりました。

疑似コードは以下のとおりです。

```
book = dict()                           #定跡保存用。
                                         保存形式例：
                                         {"sfen lnsgkgsnl/1r5b1/pppppppppp/9/9/9/PPPPPPPPP/1B5R1/LNSGKGSNL b - 1":{"7g7f,"2g2f","resign"}}
board = Board()                         #将棋盤。手を進めたり(push)、戻せたり(pop) する。


def minmax(board, book):                #定跡内の指し手を簡易的なminmax法で探索し、末端局面において勝ち(+1)か、負け(-1) か
                                         を判定し、指し手とともに返す関数。
                                         末端が"resign"で終わるようにbook登録されていることが前提
    sfen = board.sfen()                 #現局面のsfen文字列を受け取る
    if sfen not in book:                #局面が定跡になければscore = 0, move = Noneを返す
        return 0, None
    for move in book[sfen]:
        if move == "resign":
            score = -1
        else:
            board.push(move)
            score = -minmax(board, book)[0]
            board.pop()
            if score == 1:              #勝つ枝が見つかればbreakしてよい（簡易的なアルファカット）
                break
    return score, move                  #scoreとその時の指し手を返す


board = ...                             #初期設定（連続対局開始局面をどうするか、投了値をどうするか等）

while True:                             #連続対局
    score, move = minmax(board, book) #定跡内ミニマックス探索

    if score == 1:                      #定跡内の指し手に勝つ枝があればそれを指す
        bestmove = move
    else:                               #勝つ枝がなければ探索して指す
        bestmove = ...
        book = ...                      #局面と指し手("resign"も含む)を定跡に登録

    if board == ...:                    #終局時処理。千日手の場合、後手勝ちとして、先手resignとしましたが、その他の実装もあると思います。
        ...
    else:
        board.push(bestmove)            #終局していない場合、1手進める
```

・開発動機

近年、評価関数モデルの機械学習が札束による殴り合いになっているので、そこから逃走すべく定跡を整備することにした。

・開発過程

上記２．の疑似アルゴリズムで示されるコードで自動生成した。

・実験結果

思考エンジンに水匠を用い、定跡不使用のものと上記手法で自動生成した定跡を使用したものとを対局させると定跡を使用した側が 100%勝つようにできた。また選手権では、定跡負けはほとんどしなかった。

・追試可能か

可能。

以上

W@nderER アピール文書（詳細）

2023/5/19 櫻井博光

・開発動機

WCSC29 頃だったと思うが、入玉して点数も充分なのに、わざわざ相手の玉を寄せに行き手数制限で勝負が紛れてしまう将棋ソフトをいくつも見かけた。その時に受けた印象から詰みよりも宣言勝ちを目指すエンジンがあれば面白いのではないかという考えに至り W@nderER の開発を行うようになった。最近は AobaZero や TMOQ、爆裂駒捨太郎など有力な後継も現れる中で、積極的な宣言勝ちが達成できていないことにもどかしさも感じている。

・開発過程

以前よりデファクトスタンダード型の NNUE (HalfKP 256x2-32-32-1 、以下標準型) を拡張する形で、玉の位置を考慮したネットワークアーキテクチャ型の NNUE を試している。[1]

昨年頃より、tttak さんが以前考案された様々なネットワークアーキテクチャ型の NNUE の一つである HalfKP-KingSafety Distinguish Golds[2] 型（以下 HKPKSDG 型 ）に興味を持ち、第 3 回電竜戦においては、0 より学習させた評価関数を用いてそれなりの成績を修めることができた。[3]

電竜戦から WCSC33 にかけては電竜戦にて使用した評価関数の追加学習と並行して、水匠 5[4] (標準 NNUE) の中身をコピーした HKPKSDG 型の水匠 5 を作成し、そこからの追加学習も実施した。HKPKSDG 型の水匠 5 においては、KingSafety 部分の Weight が 0 埋めされており、学習でこの領域に値が付いた上で強さも損なわないことを目標とした。その後選手権が近づくにつれ、水匠 5 (HKPKSDG) の追加学習と学習結果の比較に注力するようになった。

・開発工夫点

HKPKSDG 型の評価関数の学習において、やねうら王 V5.33[5] に Bonta さんの Gaussian_Lambda[6] を組みこんだ学習部を作成し、たややんさんの公開された Suishopsv-150m[7], floodgate_validation_sfen[8] で学習を繰り返すことで良さそうな学習パラメタについて検討した。ある程度パラメタに当たりを付けた後は、「強い将棋ソフトの創りかた」書籍[9]の付録データも利用し、教師データを混ぜての評価関数の学習を実施した。

試行した中で手ごたえを感じたいくつかの評価関数候補について、学習時のパラメタは以下となる。

候補名（A～D とする）	A	B	C	D
学習データ	Suishopsv-150m をシャッフルしたもの			付録データをシャッフルしたもの (dlshogi_with_GCT-001~012)
validation set	floodgate_validation_sfen から 10 万局面を抽出したもの			
loop	1		10	1
eval_limit	2700		2500	
Loss Function	ELMO_METHOD(WCSC27)			
mini-batch size	1000000		10000	
nn_batch_size	1000		10000	
eta	0.01			0.1
Iteration 毎の eta 可変	未使用			
newbob_decay	0.5			
newbob_num_trials	5	7	5	5
LAMBDA	0.6	0.01	0.77	0.7
LAMBDA2	0.33		0.55	0.33
LAMBDA_LIMIT	32000		436	32000
mirror_percentage	0			
eval_save_interval	70000000	50000000	50000000	500000
loss_output_interval	1000000	1000000	50000000	500000
候補とした関数番号	evalsave/0	evalsave/0	evalsave/1	evalsave/0

save_only_once, no_shuffle, nn_options, discount rate, reduction_gameply についてはデフォルトのままである。
またその他パラメタとして、Gaussian_Lambda の σ は 1000 、 Ponanza 定数は 600 、FV_SCALE は 16 で固定した。

・実験手法

上記で学習したいくつかの評価関数について、学習時と同じ validation データを使用した際の各種ロスと、水匠 5（標準型）＋やねうら王 v7.63 相手の勝率を確認した。
マシン環境は c6a.xlarge で、双方 Threads は 32、ResignValue は 300 、FV_SCALE は 24 とした。

対局開始時の局面については dlshogi チームの山岡さんが公開された山岡互角局面集[10]より、重複局面を省いた 37 手目を開始局面としたものと、平手を開始局面としたものを計測した。
また、計測においてはノード数を 1 手 1500 万ノードに固定して実施し、気になった評価関数については 1 手 1 秒での秒数固定での計測も追加で実施した。NPS が違うため同ノードでの計測は水匠 5（標準型）にとって不利ではあるが、WCSC33 の本番環境では c6a.metal を使う予定であったため、標準型 NNUE 相手の多少の NPS 差は埋められると予想し、この設定にて測定を実施した。なお、HKPKSDG 型の NPS は標準型の約 2/3 である。

・実験結果

	A	B	C	D	水匠 5 KSDG (学習元)	備考
hirate eval	107	86	71	25	58	
test_cross_entropy_eval	0.483826	0.48332	0.483097	0.487157	0.484081	
test_cross_entropy_win	0.387865	0.385548	0.393708	0.407379	0.393783	
norm	1.0963E+08	1.0965E+08	1.0168E+08	9.0479E+07	1.0058E+08	
move accuracy	36.00%	36.26%	36.90%	37.18%	37.10%	
initial loss	0.387866	0.385549	0.393709	0.40738	0.393784	
対水匠 5 勝率（平手）	52% (50-4-46)	48.5% (45-7-48)	50% (46-8-46)	53% (41-24-35)	-	ノード固定
対水匠 5 勝率（互角局面）	43% (38-10-52)	45.5% (43-5-52)	54% (44-20-36)	53.5% (45-17-38)	-	
対水匠 5 勝率（平手）	-	-	50% (16-8-16)	42.5% (13-8-19)	-	1 秒固定
対水匠 5 勝率（互角局面）	-	-	49.5% (45-9-46)	42% (38-8-54)	-	

※勝率欄の (X-Y-Z) は、水匠 5 相手に X 勝 Z 敗 Y 分したということを表す。

test_cross_entropy_eval, test_cross_entropy_win, initial loss は A~C の評価関数において学習元の HKPKSDG 化水匠 5 よりも若干の良化が見られたが、対戦結果としてみれば C のみが辛うじて標準型水匠 5 と同等レベルの強さを示唆するにとどまった。一方、D の評価関数では各種ロスは上昇しているが move accuracy が元の水匠 5 よりも上がり、ノード数固定の測定においてはほんのりとよい成績を見せた。しかし D は秒数固定において C と比べ結果が揮わなかったため、最終的に選手権では C の評価関数を使用した。

・考察

A~C, D は共に高品質の教師データで学習したものであるが、生成元の評価関数が水匠 4 か dlshogi_with_GCT かという違いがある。水匠 5 ベースの評価関数に対しては水匠系の教師データの相性が良かった可能性や、dlshogi 系の教師データでは中終盤の局面の認識が甘く、学習したことで読みの量が響く中終盤で NPS 差から水匠 5 に逆転され秒数固定では勝ちづらかった可能性などが考えられるが、検証はできていない。

・課題

今回、教師データを混ぜた学習ではロスが容易に上昇してしまい、うまく学習させることができなかったが、上記で考察した評価関数との相性や教師データの性質によるものなのか、もしくは教師データの混ぜ方で改善できるのかは試せていない。
また、今回の評価関数はいずれも FV_SCALE を 24 にして測定を行ったが、これも最適な値は見つけられていない。

・追試可能性

学習自体はランダム性があるため完全な再現は難しいと思われますが、教師データ・ソースコードは公開されているため、同様の学習を行うことは可能です。

・使用データ、プログラム等

・水匠開発者のたややんさんが公開された評価関数、教師データ
・dlshogi チームの山岡さん、加納さんが公開された「強い将棋ソフトの創りかた」書籍付録の教師データ
・探索エンジン、学習部としてやねうらおさんの公開されているやねうら王
・Bonta さんの Gaussian_Lambda
・tttak さんの HalfKP-KingSafetyDistinguishGolds アーキテクチャ
上記のどれか一つでもなければ WCSC33 の W@nderER はありませんでした。比較実験のしやすい環境を整えてくださった皆様にこの場を借りてお礼申し上げます。

・リンク

[1] HalfKPKrank 型評価関数 (WSCOC~WCSC31 W@nderER アピール文書)
https://www.apply.computer-shogi.org/wcsoc/appeal/W@nderER/wanderer_appeal-v2.html
https://www.apply.computer-shogi.org/wcsc31/appeal/W@nderER/WanderER_appeal_2.pdf
[2] HalfKP-KingSafety Distinguish Golds
https://github.com/tttak/YaneuraOu/releases/tag/V4.89_NNUE-features_20200406
https://github.com/tttak/YaneuraOu/tree/NNUE-features
[3] 第 3 回電竜戦本戦 大会結果
https://denryu-sen.jp/dr3/result.html
[4] やねうら王探索部および水匠 5 評価関数
https://github.com/yaneurao/YaneuraOu
[5] YaneuraOu v5.33
https://github.com/yaneurao/YaneuraOu/tree/40ae38e45092e81d6ed35deab54796284d2830f4
[6] Gaussian_Lambda
https://github.com/Bonta0729/Gaussian_lambda/tree/062d41e1fb5d47d58d4cdf9cc49ca74910b1977e
[7] Suishopsv-150m
https://twitter.com/tayayan_ts/status/1553348516281323520
https://drive.google.com/file/d/10RuQMETwYclRggui0eb2CZGNVNVNxgdh/view
[8] floodgate_validation_sfen
https://twitter.com/tayayan_ts/status/1337419485339242500
https://drive.google.com/file/d/1JBsUMe9LiYhL7d4YDqj6dvXCCKKTGJ5X/view
[9] 山岡忠夫・加納邦彦著 強い将棋ソフトの創りかた Python で実装するディープラーニング将棋 AI (マイナビ出版, 2021)
[10]山岡互角局面集
https://tadaoyamaoka.hatenablog.com/entry/2021/09/20/222018
https://drive.google.com/file/d/1aM7fkTD6_7U61IcrOG8BI__shrZK4bHb/view

Ryfamate WCSC33 大会後アピール文

Komafont*

1 開発動機

数年間にわたり病気で入退院を繰り返したが、自力で外の世界を歩けるまでに回復した時には世の中は変わり、機械学習の世界も大きく進歩していた。そこで、機械学習の最近の動向を勉強し、画像認識や自然言語処理の分野で目覚ましい成果をあげる技術の他分野への応用を研究するため、本大会に参加した。

近年、Deep Learning (DL) の発展に伴い、大規模な計算資源を利用できるチームの優位性がより拡大していると思われるが、初参加の 2021 年に DL と NNUE [*1] の合議を 1 台の PC で実現したところ、「おうちパソコンのパワーを最大限活用する」というコンセプトに賛同してくださる方から多くの応援をいただいた。このことから、単に多くの計算資源を投下することで強くするのではなく、個人が利用できる計算資源の範囲でも、ネットワークアーキテクチャの工夫によって優位性を生み出せることを示すことを目標とした。

2 独自の工夫

2.1 ネットワークアーキテクチャ (RyfcNet)

現在、コンピュータ将棋に用いられる DL 系評価関数は ResNet である[*2] 。ResNet は、NNUE と比べ、離れた位置にある駒の関係を認識するために多数の畳み込み層による演算が必要である上、駒の絶対座標の情報を局面の認識に生かすことが困難である。

Ryfamate は、これらの課題に対応するため、NNUE や Transformer 系の評価関数[*3] を参考に、Ryfamate Cross Network (RyfcNet) を開発した。RyfcNet は、畳み込みを、任意の次元について入力空間と同じ長さを持つカーネルをそれ以外の次元の方向に移動させながら適用する変換と捉え、その次元を層ごとに適切に選択することによって、少ない演算回数で上記の課題を解決した。例えば、既存の ResNet は、中間層に用いる畳み込み層はすべて 3×3 の部分空間ごとに特徴量を得るものであるが、RyfcNet では 1×9 や 9×1 の部分空間ごとに特徴量を得る畳み込み層を一部で採用することによって、離れた位置にある駒の関係を少ない回数の畳み込みで認識することが可能である。また、この畳み込み層は、既存の ResNet で用いられる畳み込み層に比べ、処理が高速かつ絶対座標に依存する局面の認識が容易であるという特徴がある。詳細は、事前に公開したアピール文[*4] に比較実験の結果とともに記載しており、追試可能である。また、要望が多ければサンプルコードを公開したいと考えている。

---

* 駒の書体 (Komafont) https://twitter.com/komafont

*1 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. 2018.

*2 山岡忠夫, 加納邦彦. 強い将棋ソフトの創りかた Python で実装するディープラーニング将棋 AI. マイナビ出版, 2021.

*3 Bichen Wu et al.. Visual transformers: Token-based image representation and processing for computer vision. 2020.

*4 https://www.apply.computer-shogi.org/wcsc33/appeal/Ryfamate/appeal_ryfamate_20230421.pdf

2.2 開発過程

新しいネットワークアーキテクチャを開発するためには、多くの試行錯誤を行う必要がある。しかし、本番用のモデルを 1 つ学習させるには、GPU 1 台で数か月かかる上、アーキテクチャごとに適切なハイパーパラメータが異なることから、個人で多くの試行錯誤を行うことは難しい。このため、本開発においては、実験用の学習では小さいサイズのモデルに対して、Optimizer に Adam や LAMB を、Scheduler に Warmup+CosineAnnealing を用いることで、ハイパーパラメータの学習結果に対する感応度を抑えつつ、早く誤差を収束させている。Adam 系の Optimizer は、SGD と比べて、十分な時間をかけて学習した際の汎化性能が低いという問題が指摘されるが、epsilon の値を高くするといったパラメータ調整によって、この問題は緩和することが可能である。[*5] これにより、相対的に短い時間でアーキテクチャの大まかな性能と性質を評価することができ、より大きいサイズのモデルで学習させるアーキテクチャの選定を行った。

なお、小さいモデルにおける比較実験の結果に比べ、大きいモデルでは既存の ResNet に対する優位性が拡大する傾向がある。本番では、20 ブロック 256 チャンネルの RyfcNet(Ryfc20b) を用いたが、学習途中のモデルで Ryfc20b 2.0s vs dr2-exhi 3.0s [*6]の対局を行ったところ、勝率は 64.75% [*7]であった。現在所有する計算資源のみで厳密な比較実験を行うことは現実的ではないが、事前の予想以上に強くなっており、ネットワークアーキテクチャの工夫が今回の大会で善戦した一因になっているのではないかと考える。

3 おわりに

Ryfamate は、DL と NNUE による合議制を採用したプログラムであるが、合議のもととなる探索部には dlshogi [*8] とやねうら王[*9] を一部改良して用いている。教師局面の作成や計測には、dr2-exhi や水匠をはじめ、公開された多くの評価関数を用いているほか、加納氏、山岡氏、たややん氏によって公開された大量の教師局面も学習に用いている。

また、初出場の 2021 年には、大会当日の Zoom による交流で、tanuki- 開発者の nodchip 氏に NNUE の学習に関する多くの質問に答えていただき、白ビール 開発者のたま氏には Multi Ponder の開発経験から合議エンジンに関してアドバイスをいただいた。このほかにも、大会出場に関連して、多くの開発者からアドバイスやご協力をいただいている。

そして、本大会の開催にご尽力いただいた、主催者の皆様、スポンサーの皆様をはじめ、多くの関係者の方々に厚くお礼を申し上げたい。

[*5] Dami Choi et al.. On Empirical Comparisons of Optimizers for Deep Learning. 2019.

[*6] dr2-exhi は、山岡氏によって 2021 年に公開された ResNet 15 ブロック 224 チャンネルのモデル。

[*7] 158 勝 86 敗 26 分。勝率は、引き分けを除くベース。その他の条件は、事前に公開したアピール文と同様。

[*8] https://github.com/TadaoYamaoka/DeepLearningShogi

[*9] https://github.com/yaneurao/YaneuraOu

二番絞り

二番絞りの誕生経緯については昨年のアピール文などを参考に頂ければ幸いであるが基本的には最高精度を目指す大きなモデルを作成しようとする試みである．

昨年度は幸いにも準優勝という好結果を得た．準備段階では手ごたえがなかったがその後の計測で大変高精度なものが完成していたことが分かった．今年度も似たような状況である．まともな計測に至っていないが，もし昨年度より弱いと判断した場合は昨年度版で出場する可能性がある．

ちなみに，昨年度版の局面評価精度は驚愕の域に達しており，既発表であるが一手の局面展開も行わず将棋倶楽部 24 でレート 2949，八段認定頂いている．前人未到の領域といって過言ではないだろう．
また，電竜戦ハードウェア統一戦においても準優勝となり，記念に 2017 年に行われたハードウェア統一戦の第 5 回電王トーナメントを思い起こし GTX1080Ti の二番絞りを floodgate に投入したところ短期レート 4500 台を記録した．（もちろん一時的なものでありしばらくしてレートは 4100〜4200 程度で落ち着いた）

今年度はこれを上回ることを目指しているが上記の通り昨年に続き未計測である．

参考：

芝，「将棋の PV-MCTS に向けた深層学習モデルの最適化」，第 45 回ゲーム情報学研究会
芝，「探索アルゴリズムに適した時間利用に関する研究」，第 46 回ゲーム情報学研究会
第 32 回世界コンピュータ将棋選手権，https://bleu48.hatenablog.com/entry/2022/05/06/145915
芝，「コンピュータ将棋における高精度な深層学習モデル」，ゲームプログラミングワークショップ 2022
二番絞り＠将棋倶楽部 24 の戦型分析，https://bleu48.hatenablog.com/entry/2023/03/08/062634
二番絞りの計測の件（GX1080Ti 編），https://bleu48.hatenablog.com/entry/2023/03/09/132936

選手権後の追記：

本年の二番絞りは昨年モデルに対して追加学習を行った．昨年のモデルは大変な高精度であることが確認されたがこれを上回るためには，単純に更なる高精度の教師データが必要であると考えた．昨年一部流用した書籍「強い将棋ソフトの創りかた」提供データを上回る精度を目標に 5000 ノード探索の自己対局を行った．そのため新規に RTX4090 を 2 枚購入し既存の PC に搭載することで計算機リソースの増強を図った．しかしながら，昨年主催した電竜戦マイナビニュース杯ハードウェア統一戦の運営期間中に電力的な制限が生じるなどの理由により保有する計算機を十分に回すことができなかった．そのため予定より大幅に少ない約 1 億局面程度の自己対局教師データしか生成できなかった．元々2020 年に開始した二番絞りプロジェクトの当初の 20 ブロックモデルであっても 1 億局面の教師データでは不足し過学習となることは確認していたため本年のモデルは過学習となることは学習前から予想の範囲であった．

1 億局面の学習に約 1 週間要し，その後の 1PC1GPU 程度の計測により昨年モデルとほぼ同程度の強さであることを確認し，本年本番投入した．しかしながら，やや過学習気味は否めなく得意の先手番はともか

く後手番を苦手としたようである．例えば初期局面の評価値は昨年モデルが約 200 としたのに対し約 250 程度となっている．

二番絞りプロジェクト 20 ブロック当初でも強化学習の 1 ステップの周期が 2 カ月程度であったが，それが 40 ブロックとなって約半年程度となっていた。現在の計算機リソースではとうとう 1 ステップが 1 年を超えてしまったことが判明したため今後上位を目指すなら大幅な方向転換が必須と思われる．今後は二番絞りらしい対応を検討中である．

選手権運営側の追試可能かという質問についてはハードウェアを用意頂ければ対応可能である．コスト面でも相当厳しいがクラウド利用はある程度利用実績が無いとハイエンドインスタンスの起動許可すら得られない．もちろんチームとしても選手権運営負担で追試できるのであれば幸いである．

dlshogi のソースを一部利用している点についてコメントを入れて欲しいとの要望があった．dlshogi に関しては比較的初期からの Contributor である．つまりライブラリの利用者ではなく開発者の一員と考えて頂ければ適当に思う．そもそもの経緯は 2017 年 4 月に始まり，同月には本家より高精度の学習を行った結果を blog にも掲載し交流が開始している．また，大きなものとしては 2020 年二番絞りプロジェクト開始後には AMP（Automatic Mixed Precision）対応を施すことにより学習時間を約半分にするフィードバックを行ったり学習精度を上げるヒントとなる情報を共有したりなど劇的な変化へ繋がったと考えている．私が考えるには電竜戦プロジェクトがきっかけとなり，二番絞りを含む複数の協力者が互いに切磋琢磨し情報交換したため現状のオープンソースの dlshogi があると言うのが実情であろう．選手権のライブラリ規定の改定経緯を御存知の方には言うまでもないが，オープンソース開発と言う意味ではやねうら王も同様に多くの Contributor の賜物であることも加えておく．

参考：

電竜戦マイナビニュース杯ハードウェア統一戦開幕，https://bleu48.hatenablog.com/entry/2022/12/24/142241

第 1 回マイナビニュース杯電竜戦ハードウェア統一戦の戦型分析，

https://bleu48.hatenablog.com/entry/2023/03/07/170532

第 33 回世界コンピュータ将棋選手権，https://bleu48.hatenablog.com/entry/2023/05/05/160055

東横将棋　詳細アピール文書

東横コンピュータ将棋部

概要及び開発動機

W SC33 版の東横将棋は、やねうら王探索部＋標準 NNUE 型評価関数ベースのコンピュータ将棋エンジンです。ディープラーニング系およびそのハイブリッドが全盛の現在において、これはかなり枯れた構成と言えるでしょう。一般家庭で検討用途に使用される際には、まずこれが選択されると思われます。
この構成と定跡の強化で現在の主流ソフトにどこまで対抗できるかというのが主な開発動機です。
WCSC33 版の東横将棋は、二次予選の初戦以外、電竜戦さくらパイルール 2023 で優勝した標準 NNUE 型評価関数を使用しています。探索部についてはあえて修正すべき部分はないと判断したのと、予期せぬ不具合を回避するため、従来のやねうら王探索部をそのまま使用しました。

使用ハードウェア

二次予選および決勝リーグでは、AMD Ryzen Threadripper 3990X を使用しました。このハードウェアは「高級スリッパ」とも称され、かつて最強の家庭用パソコンとして名を馳せましたが、発売から 3 年が経過し、昨年時点でも既に時代遅れのものとなっていました。
WCSC33 において決勝リーグではクラウド利用を前提にしていましたが、少なくとも二次予選までは Ryzen Threadripper 3990X を使用することは事前に決めておりました。
二次予選を突破した時点で、今度は決勝リーグで上位チームのモンスターマシンにどこまで対抗できるかに興味が湧き、決勝リーグでも結局は Ryzen Threadripper 3990X の使用を継続することになりました。

評価関数について

昨年から標準 NNUE 型評価関数の強化に注力しております。しかし、標準 NNUE 型評価関数は既に限界に達しているとされています。その中でできる限りの強化学習手法を試みましたが、結果としては従来の評価関数から僅かながらしか強化することができませんでした。
前述した通り、標準 NNUE 型評価関数はこれ以上の強化は困難と考えていましたが、大会後に強力な評価関数が公開され、標準 NNUE 型評価関数にもまだまだ伸びしろがあることを再認識した次第です。

定跡について

角換わりや相掛かりを重視した定跡の作成に力を入れました。相掛かり定跡と言えば、s-book_black が有名ですが、昨年の大会以降、ライセンスの関係でどこのチームも使用できない状況にあります。そのため、ほぼゼロから定跡を作成しました。この過程で、ディープラーニング系の中でライセンスでゴリゴリガッチガチに固めているソフトの使用は避け、s-book_black フリーおよびライセンスフリーの原則を遵守しました。
角換わり定跡については水匠定跡及び floodgate の棋譜を参考に、相掛かり定跡については主に floodgate の棋譜等を参考に作成しました。しかし、結果的に大会までに納得のいく定跡が完成したとは言えませんでした。

実験結果等

既存の各種評価関数との大量の自己対戦及び floodgate での対戦結果、電竜戦さくらパイルール 2023 での優勝により、ある程度の結果は望めると判断いたしました。

追試の可否について

ハードウェア及び使用したソフトウェアは全て手元にありますので、いつでも追試可能です。

謝辞

大会中は本当に楽しい経験をさせていただきました。運営の皆様、スポンサー様、他の参加者の方々には深く御礼申し上げます。
多種多様な将棋ソフトとの対戦は刺激的で、充実した時間を過ごすことができました。次回の大会では、貴重な対戦経験を活かし、さらに技術を磨いて上位入賞を目指したいと強く思っています。
大会に参加することは私にとって非常に貴重な経験であり、皆様の温かいサポートや応援があったからこそ、6 位入賞と新人賞受賞という素晴らしい結果を得ることができました。心からの感謝を込めて、改めてお礼申し上げます。コンピュータ将棋界でのさらなる成果や活躍をお届けできるよう、精一杯努力し続けます。引き続き皆様のご支援と応援をよろしくお願い申し上げます。

大将軍 詳細アピール文書

横内健一・横内靖尚

概要

昨今のコンピュータ将棋において評価関数は劇的な変化を遂げてきている。Bonanza で始まった従来の 3 駒関係をベースとした評価関数から NNUE を経て、Deep Learning を用いた評価関数も登場し、その精度の高さは GPU の性能向上と相まって飛躍的に進歩している。

大将軍は、そのようなトレンドのなか、評価関数については今更ながら従来の 3 駒関係（15 年以上前の技術）をベースに大会に参加している。

開発動機

大将軍は以前に N4 や N4S という名前で大会に参加していたことがあるが、4 駒関係を用いた評価関数を使用していた。その際の経験では、序盤はそれなりに戦えるものの、中盤以降、深い読みができず、敗戦するケースが多かった。評価関数の表現力を高め大局観の精度を高めることは大事ではあるものの、将棋の場合、最終的には詰む・詰まない、を正確に読み切ることが重要であり、具体的な手順を深く読む必要がある。そのため将棋では 4 駒関係のような重い評価関数は不向きとの考えがあった。

そこで、あえて比較的軽い 3 駒関係の評価関数と最新の探索アルゴリズムを組み合わせた場合、どの程度大会で戦うことができるかをモチベーションに大会に参加した。

開発過程

元祖の Bonanza 型の 3 駒関係を出発点として、4 駒関係などの開発を進めてきたが、これらの評価関数では、手番が評価に含まれないため、先後同一局面では評価値が 0 になってしまうという課題があった。そのため、3 駒関係をベースとして、さらに手番を考慮した 3 駒のモデル（kkpt-kpp）で大会に参加していた。昨年からは一般的な手番を考慮した 3 駒のモデル（kppt 型）とした。

使用ライブラリとその選定理由

やねうら王の選定理由

ソースコードがわかりやすく、ベースのエンジンとして使用

水匠 2,3,4,4 改の選定理由

学習の棋譜生成に使用（強いソフトで生成した棋譜を利用）

独自に工夫した点

これらは過去からの工夫で、今では常識の技術ではあるが、3 駒関係のモデルでは、評価関数の差分評価（動いた駒のみの評価値を更新）をいち早く取り入れて採用している（4 駒関係の計算では、すべての駒の配置に対して評価値を計算すると組み合わせが多く膨大な計算量が必要となるため、差分計算を取り入れると 3 駒分の計算量で済む経験を利用）。また、学習においてプロの棋譜からの学習時においてミニバッチ方式を採用した。これにより、パラメータの更新回数を相対的に増やすことができ、学習を効率よく行うことができた。

今回の新しい取り組みとして、学習対象とする局面に対し、状況に応じた学習量のパラメータ調整を試みた。やねうら王を用いた学習では、深い探索の評価値や勝敗をもとに損失を計算する。また、学習の対象とする局面は、現局面ではなく、現局面から浅い探索（静止探索）を行い、その末端の局面に対して学習を行うのがよいとされている。

しかし、深い探索の進行で学習を行うとき、深い探索の進行と浅い探索の進行が連続して一致する場合、例えば図 1 において、深い探索の進行が、局面 A から局面 B、局面 C と進行し、浅い探索の進行も同様だった場合、局面 C について、3 回学習が行われることとなる。これが本当によいのか疑問があった。

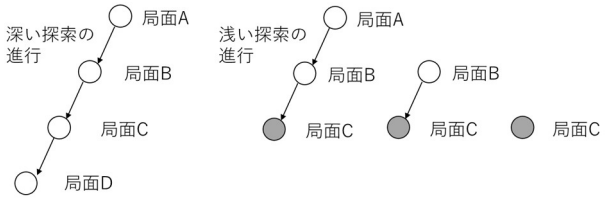

図 1　深い探索と浅い探索が一致するケース

次に、図 2 のように、深い探索の進行と、浅い探索の進行が異なる場合に、浅い探索の末端に対して、深い探索の評価値や勝敗を反映させることがよいのかという疑問もあった。

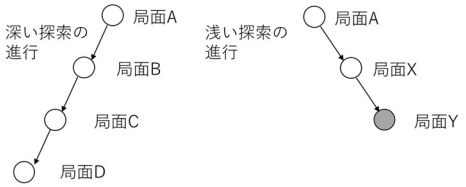

図 2　深い探索と浅い探索が一致しないケース

そこで、上記のケースにおいて、学習量に係数を設けパラメータとし、調整を試みた。

実験結果

深い探索と浅い探索の進行が一致する場合の影響について、浅い探索の末端局面の学習量に対して式 1 の係数 factor を乗算することで調整した。この式の pv.size()は浅い探索の深さを意味する。探索が深いほど、その後に同じ局面が複数回学習される可能性が高いと仮定し、学習量を減らすことを意図している。

$$factor = \frac{p}{pv.size() + p} \qquad (式 1)$$

p をパラメータとし、この係数を考慮せず学習したものと 1000 局対戦した勝率を図 3 に示す。

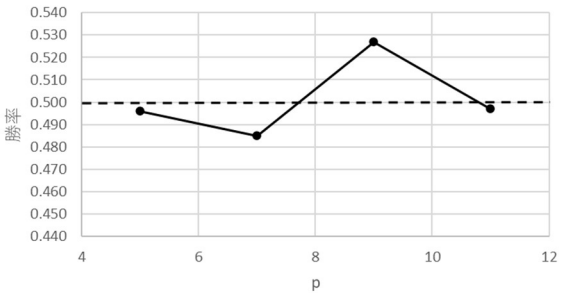

図3 パラメータpと勝率

微妙ではあるが有利な条件があり、探索の深さに応じて学習量を減らすことを採用した。

また、深い探索と浅い探索の進行が一致しないケースについては、単純に学習量に係数 q を乗算した。係数 q をパラメータとし、この係数を考慮せず学習したものと 1000 局対戦した勝率を図 4 に示す。

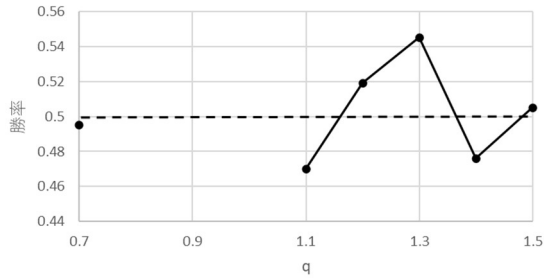

図4 パラメータqと勝率

係数 q は 1 未満のほうが勝率は良くなると考えていたが、1 より大きい条件で有利なものがあった。ただし、不安定な結果となった。

追試可能か

学習は、教師局面をランダムにシャッフルしているため、全くの再現は難しいと考える。

また、本アイデアの効果は、学習モデルの違い、学習の成熟度、学習データ、各種パラメータの設定など様々な要素により異なると思われ、同様の結果が再現するかはよくわからない。ただ、現局面ではなく、静止探索の末端局面に対して学習させると強くなる理由は解明されておらず、今後も研究対象となると思われる。

以上

「アストラ将棋」アピール文書

令和５年５月
恒岡 正年

概要：

アストラ将棋 は、山岡氏著「強い将棋の造りかた」という本の内容に沿って dlshogi をベースに作成されたコンピュータ将棋エンジンです。

自前のモデル作成を行い、探索パラメータを本番で使用するハードと持ち時間に合わせて調整しました。

UctSearch.cpp と usi.cpp を改造しましたが、それほど大きな改造ではありません。

学習時に使う data_loader.py や特に train.py は大きく改造しています。

定跡を作成しましたが考時間の短縮と後手番での特定の局面を避けるのが目的で規模は小さいです。

ベンチマークの相手には主に水匠５を使用しました。

ベンチマークの開始局面には、たややん氏・山岡氏の公開されたもの、自前で用意した物等幅広く使用しました。ベンチマークは棋譜生成を兼ねており、より多くの学習局面を生成するためです。

開発動機：

以前に電通大で開催された囲碁セミナー(松原氏,山下氏が講師をされました)に参加して囲碁 AI の開発を始めたのですが、アルファ碁の登場で桁違いの物量を生かした開発が主流になり撤退。

「強い将棋の造りかた」の本に触発されて将棋 AI の開発に参入。

ディープラーニング(DL)の特に学習部に興味があり、DL の勉強を兼ねて自前の model を作ろうと思いました。そのため、model の学習に最も注力しています。

独自に工夫した点：

独自構造の model の模索・作成、学習方法(マクロバッチ実装、他)、生成モデルのチューニング、探索パラメータの調整方法等。

開発過程：

2022 年１月から「強い将棋の造りかた」(*1)に沿って pytorch の勉強を兼ねて勉強開始。

2022 年５月頃まで色々な構造の model の生成と共に、python-dlshogi2 を改造して floodgate へ投入。GPU に GTX1080ti を使って R3400～R3500 位だったと記憶しています。

2022 年５月以降「強い将棋の造りかた」の第７章「GCT 電竜を超える強い将棋 AI を創る」へ移行。

2023 年２月後半まで、色々な model の実験・学習に注力、及び棋譜生成を続ける。

マクロバッチを実装し、疑似的に GPU のメモリ量が多くなったのと同じような結果が得られる様にした。(Batch Normalization 部分の挙動が異なるので全く同じにはならない)

最終的に 110 番目のネットワークの model を今回使用。学習途中で中断した物が殆どで、収束するまで学習させたのは 20 個程度。使用する model はこの時点で決定。

2023 年４月末まで、探索パラメータの調整(2 カ月)及び定跡生成(手作業で 2 週間ほど)

ネットワーク構造等に関しましては、既提出済のアピール文書(4 月 29 日版)をご参照ください。

実験結果：

有意差を理論的に示せるほどの比較実験を行う計算資源が無かったのと、新しい事を多く試したかったので主観的な判断で変更項目の採用・不採用を決めました。

これだけでは面白くないので、パラメータの調整の手法について書きます。
6656po(RTX3090 で 0.5 秒程度)でベンチを取って 3 つの有力なパラメータセットが得られたとします。 3 つのベンチの勝率の差は 1～ 2 %程度の差でした。これらのパラメータセットを 15s+F1.2s (探索量が 2 倍以上多い)で評価すると勝率の差が 8%程度出るケースを確認しています。パラメータセットによって探索時間を大きくした時の伸びしろが異なるという結果でした。

そのためパラメータの調整を行った 2 カ月間は、6656po (optuna を利用、パラメータ 3~4 個単位で調整) => 15s+F1.2s(手作業のベンチで調整) => 20s+F5.0s(同左) と 3 段階で探索量を増やしながら評価を行い、パラメータセットも最良の物だけではなく 2nd,3rd...best の物も候補として評価し、段階的に候補を絞っていきました。
パラメータセットを 2 つに絞ったあと、さらに Softmax_temperature と C_base, C_base_root (設定の粒度が小さい)の調整を再度行い、最終的に 1 つを選びました。(もう一つは予備へ)
この 2 カ月の探索パラメータの調整で R100 程度強くできたと思います。
(2 月末時点で使用していた手動調整したパラメータの出来が不十分だったとも言えます。)

3 月中頃に floodgate に流した結果では、13056po(RTX3090 で 1 秒弱の探索時間) の探索量で R3800 程度ありました。本番のハード(RTX4090)と 900s+F5s の持ち時間及び ponder 有りの条件では 32 倍強の探索量と推定。探索量が 2 倍になるとレーティングが R+80 増加すると仮定すると R4200(～4300)程度になったと予想しました。(Top クラスは R4800～R5000over でしょうか？)

追試可能か：

同一のネットワーク構造を用意し「強い将棋の造りかた」の内容に沿って学習し、パラメータの調整を行えば、同様のモデルが得られると思います。
ただ、短時間で学習する手法(RTX3090 で 3 週間弱)とか、学習後のモデルの調整、optuna を使ったパラメータの効率的な調整は経験に依存する部分も多いと思います。

最後に：

dlshogi の成果を利用した上での参加ではありましたが、1 台の PC で参加し決勝戦に出場できたのは幸運だったと思います。今回のモデルは定跡を抜けてから 80 手程度までは上位ソフトと互角に近い状態を維持できましたが、90 手目前後以降で読みの精度が低いことを確認しました。これは、KL 情報量が多い範囲の局面を意図的に多く学習させ終盤に近い局面を減らした事、学習時に重複局面を除去しなかった事(序盤の局面が増える)によると推定しています。
モデルの出来は満足できる物ではありませんでしたが、2 月末以降残り時間を探索パラメータの調整に費やしたのは結果的に良かったと思います。
今回、モデルの学習方法や探索パラメータの調整方法の知見が得られました。反省点を生かした次のモデルではすでに終盤力の向上を確認できております。(序盤の精度が低下している可能性はある。)

謝辞：

将棋 AI に関わるきかっけとなった本を出版された山岡様、加納様 には深く御礼申し上げます。
特に加納様には SNS 上で多くのアドバイスをいただきました。

また、WCSC 開催に御尽力されている関係者・スポンサー様にお礼を申し上げます。

参考文献：

*1: 山岡忠夫, 加納邦彦. 強い将棋ソフトの創りかた Python で実装するディープラーニング将棋 AI. マイナビ出版, 2021.