Ari Shogi WCSC33アピール文書

兵頭優空

最終更新: 2023/05/15 21:00

1.概要

Ari Shogi(以降Ari)はディープラーニングを使用した将棋AIである。

今までの大会に出場していたバージョンとは異なり、今回はpython-dlshogi2をベースにしている。(なので割と強い)

今選手権では、1次予選で強運を発揮して10位に滑り込み、目標であった「2次予選に進出する」を達成する事ができた。

Ari Shogiは、今選手権において、ビール工房HFT支店様からのご寄付により参加費を免除していただきました。心より感謝申し上げます。

2.使用ライブラリ

プログラムの一部として呼び出されたりしているライブラリ

・python-dlshogi2: ベースとして使用

- 採用理由1: シンプルで分かりやすい実装でかつ強く、改造しやすそうだから。
- 採用理由2: 私がほぼ唯一使えるPythonで書かれているから。

・cshogi: 合法手生成などで使用

- 採用理由1: python-dlshogi2で使われているから。
- 採用理由2: 私がほぼ唯一使えるPythonで利用できる高速な将棋ライブラリだから。

・KomoringHeights: root局面での詰み探索で使用

- 採用理由: USIプロトコル経由で利用できて、性能も高そうだったから。

・dlshogi: 探索部のパラメータを自動で最適化するコードを使用

　- 採用理由: Optunaを使用したパラメータの自動調整がほぼ改造なしで利用できたから。(Optunaを使用したパラメータの自動調整の部分以外は使用していません)

定跡生成 / パラメータ自動調整用の対局で使用したソフト

・水匠5: 定跡生成とパラメータ調整とテストに使用

　- 採用理由1: CPUのみで動き、かつ強いから

　- 採用理由2: 多くのNNUE評価関数の基になっているから

・技巧2: パラメータ調整とテストに使用

　- 採用理由1: CPUのみで動き、かつ強いから

　- 採用理由2: floodgateでの基準ソフトになっているので、強さの指標として用いるには都合が良いと考えたから

　- 採用理由3: floodgateでのレートがAriと同じくらいだったから

・Apery(WCSC30): 定跡生成とパラメータ調整とテストに使用

　- 採用理由1: CPUのみで動き、かつ強いから

テスト対局で利用したソフト

・LesserKai

・Kifuwarabe(WCSC31)

・TakeWarabe

など(主な採用理由: PCに既に入っていたから)

これらのライブラリ / ソフトを公開してくださっている方々に心より感謝申し上げます。

3. 教師データ

・書籍「強い将棋ソフトの創りかた」の付録の教師データ

- dlshogi_with_gct-のデータを利用

- 採用理由：質が非常に高いうえに量も結構あるデータセットだから

・floodgateのデータ(http://wdoor.c.u-tokyo.ac.jp/shogi/index.html)

- 2018年 ～ 2022年までの対局から以下の条件で抽出

- 50手以上の棋譜 & 対局者の低い方のレートが3500以上 & 投了か千日手か入玉宣言で終局している

- 採用理由：様々な年代の様々なソフトによる様々な対局が入っているから

・AobaZeroの公開されている教師データ(http://www.yss-aya.com/aobazero/)

- arch000061110000 ～ arch000063180000から以下の条件で抽出。

- 51手以上続いた対局 & 投了か千日手か入玉で終局している

- 採用理由1：質が十分でかつかなりの量があるから。

- 採用理由2：AobaZeroは「序盤に強い」と言われており、学習データに加えれば序盤が強化できるのではないかと思ったから。

・dlshogi with GCTのWCSC31バージョンの公開されている教師データ
(https://tadaoyamaoka.hatenablog.com/entry/2021/05/06/223701)

- selfplay-unique-90001.hcpe ～ selfplay-unique-90303.hcpe を使用

- 採用理由：質も良くや量も十分あるから。

・Aoba駒落ちの公開されている教師データ(http://www.yss-aya.com/komaochi/index.html)

- arch000010800000 ～ arch000012990000 から以下の条件で抽出。

- 51手以上続いた対局 & 投了か千日手で終局している(入玉関連は平手と点数の

計算などが違うため除外)

- 採用理由1: 駒落ちの対局データでは量も質も高い部類に入るから。

- 採用理由2: 駒落ちのデータを学習データに加える事で学習データの多様性が増し、強くなるのではないかと思ったから。(効果は未検証)

これらの教師データを公開してくださっている方々に心より感謝申し上げます。

(自己対局による教師データ生成はリソースの都合で断念した)

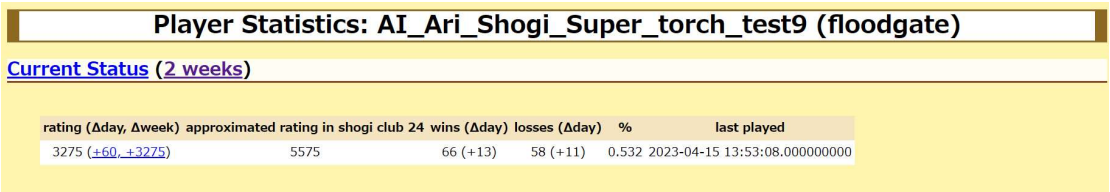
4. 評価関数

電竜戦さくらパイルール2023(https://golan.sakura.ne.jp/denryusen/dr4_sakura/dr1_live.php | 以降、さくらリーグ)に出場した時のモデルにAoba駒落ちのデータを加えて追加で学習を行ったもので出場した。(しかし精度はほぼ伸びなかった)

さくらリーグで使用したモデルは、ResNetのResBlockの後の部分にSeBlockを入れた構造になっていること以外は基本的にpython-dlshogi2のオリジナルとは変わらないもので、チャンネル数は128、ブロック数は5、全結合層のユニット数は32、SeBlockのボトルネック係数は16、総パラメータ数は約190万
(https://rheinmetall.hatenablog.com/entry/2021/05/14/000000 のコードで算出)となっている。

さくらリーグで使用したモデルを、(さくらリーグで使用した探索部より古い)モデル完成時の探索部に搭載し、当時の定跡を付けてfloodgateで計測したところ、レート3200後半、3300近くをつけ、レート3300の基準ソフトgikou2_1cなどにも勝ち越せていた。

(coduck_pi2_600MHz_1c相手に時間切れ負けしていなかったら3300を超えていた可能性もあると思われる)



Player Statistics: AI_Ari_Shogi_Super_torch_test9 (floodgate)

Current Status (2 weeks)

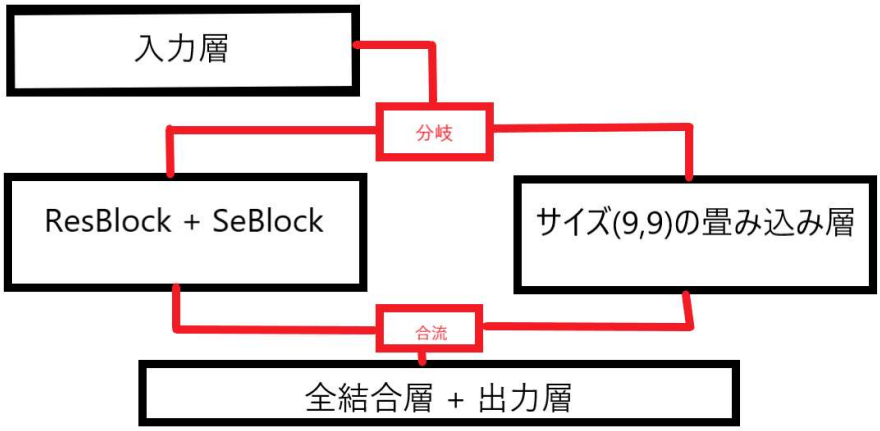
rating (Δday, Δweek)	approximated rating in shogi club 24	wins (Δday)	losses (Δday)	%	last played
3275 (+60, +3275)	5575	66 (+13)	58 (+11)	0.532	2023-04-15 13:53:08.000000000

注. この画像が撮影された時、将棋倶楽部24レート換算の数値は壊れていたので、24レート換算の値は全く参考になりません

v.s.	wins	(Δday)	losses	(Δday)	%	(today)
Incinerator	0	-	1	(+1)	0.00	(0.00)
dia_sd3bk_ry4500U	0	-	1	-	0.00	-
CSSP	0	-	1	(+1)	0.00	(0.00)
D01-DAI	0	-	1	(+1)	0.00	(0.00)
KT-1	0	-	1	-	0.00	-
YO_763_Celeron	0	-	1	-	0.00	-
D01-KAI	0	-	1	-	0.00	-
testf3_allGPU	0	-	1	-	0.00	-
i3_4160t	0	-	2	(+1)	0.00	(0.00)
Kristallweizen-Core2Duo-P7450	0	-	3	-	0.00	-
Suisho5_750_473stb_1000k	1	-	9	(+1)	0.10	(0.00)
AobaZero_w4263_n_p800	0	-	6	(+3)	0.00	(0.00)
AobaZero_w4258_n_p800	0	-	7	-	0.00!	-
gikou2_1c	8	(+2)	5	(+1)	0.62	(0.67)
Krist_483_473stb_1000k	14	(+5)	6	(+1)	0.70+	(0.83)
yo-nn_r0-ep10_2m-n3350	2	-	3	-	0.40	-
elmo_WCSC27_479_1000k	17	(+4)	2	(+1)	0.89	(0.80)
Yss1000k	5	-	0	-	1.00	-
coduck_oci_1c	9	(+1)	0	-	1.00	(1.00)
coduck_pi2_600MHz_1c	4	(+1)	1	-	0.80!!!	(1.00)
test	1	-	0	-	1.00	-

その後、「Pythonのような遅い言語で書かれた探索部では多少速度を犠牲にしても精度を上げたほうが強いのでは」と思うようになって、色々と試そうとしたが、学習が間に合いそうになかったので断念した。(今後の大会などでは使うかもしれない。)

以下の図のようなネットワークを試していた。



9 * 9の畳み込み層の部分の効果で盤面全体を見れるようになり、CNN系の弱点といわれている「遠くの駒の関係を学習するにはそこそこの層数が必要」がマシになる効果が得られる事を期待している。

5. 定跡

今回のAri Shogiの定跡は「そこそこな手を指しつつ時間を温存する」という事を目標にしている。

序盤に出てきそうな局面を強い外部のエンジン(今回は水匠5)で調べて最善手を登録していくシンプルな方式で作った。

1. floodgateの棋譜で出てくる局面

2. 適当なAIとの対局で定跡が抜けた局面(対局自体は定跡を抜けた局面で打ち切る)

といった局面を登録して定跡を作っていった。

登録局面数は40,000ほどとなっている。

登録する手は、水匠5を約2億ノード思考させた時の最善手とした。

水匠5を採用した理由は

1:強い

2:CPUだけで動く

3:既にPCに入っている

の3つ。

DL系のAIの方が序盤が優れているといわれているが、GPUは空けておきたかったので見送った。

6. 探索部

Ari Shogiの思考の手順

1. 定跡や投了や入玉宣言などのチェックを行う。

2. 「詰め将棋エンジン1」にroot局面(思考対象の局面)を送って詰みを探させる

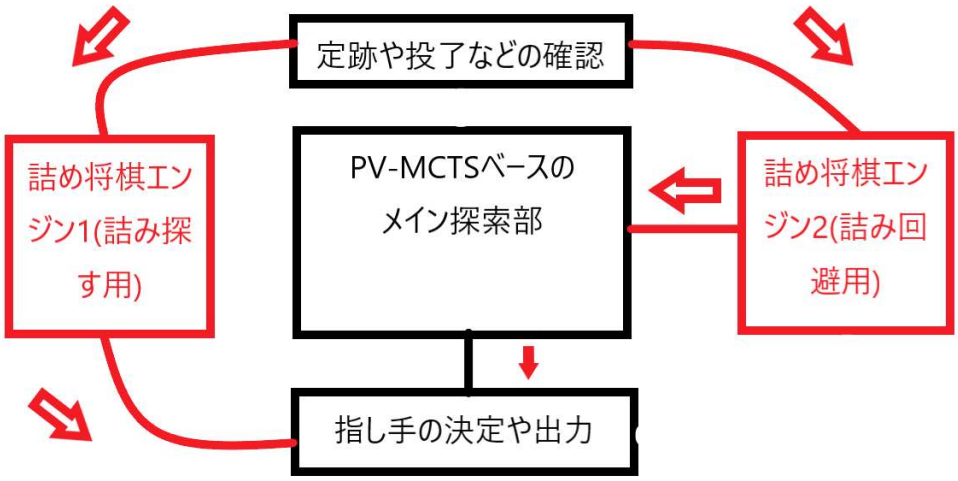
3. 詰みを回避するためのプログラムを動かす。(「詰め将棋エンジン2」はここで使う)

4. メイン探索部で思考する

5. メイン探索部が最善手を決めたあたりのタイミングで「詰め将棋エンジン1」を止める

6. 「詰め将棋エンジン1」が詰みを見つけた場合はその手を指す。それ以外の場合はメイン探索部の最善手を指す。

[参考図]



探索部に施した改造

- root局面で外部の詰め将棋エンジンを呼び出す

・投了チェックなどを済ませ、Ari本体が探索を開始する前に外部の詰め将棋エンジンに局面とgoコマンドを送信し、Ari本体が最善手を決定したあたりでstopコマンドを送信、不詰みだった場合はAri本体の手を採用、そうでない場合は詰め将棋エンジンの指し手を採用するようにしている。

・手軽な改造だが長い手数の詰みが見つけられるようになるのが良い。

・外部エンジンの呼び出しにはsubprocessを使っている。

- 外部の詰め将棋エンジンを使った詰み回避用プログラム

・通常の探索を始める前に、root局面の全ての子ノードに対して外部の詰め将棋エンジンを走らせる。

・詰みが見つかった局面は探索されない & 指し手として選ばれない

・ここで走らせる詰め将棋エンジンは、root局面で走らせているものとは異なり探索ノード数などに制限がかけてある。(選手権では1万ノードで制限していた)

- 複数種類のAIとの対局によるパラメータの自動調整

・dlshogiのUSIエンジンのオプションの値をOptunaで調整するプログラム
(https://tadaoyamaoka.hatenablog.com/entry/2019/01/06/215203 | https://github.com/TadaoYamaoka/DeepLearningShogi/blob/master/dlshogi/utils/usi_params_optimizer.py)を改造し、1種類のエンジンとの対局だけでなく、複数種類のエンジンとの対局でパラメータを調整するように変更した。

・1種類のエンジンとの対局で求める場合より汎用的なパラメータが得られる事を期待しているが、効果は検証できていない。

・持ち時間の条件も複数用意する(秒読み1秒の対局だけでなく、3秒や5秒の対局も一定数混ぜる、など)のも考えたが、リソースの関係で断念した。

- CPUCTの設定を自分の手番か相手の手番化で変える

・CPUCTの設定では、ざっくり言うと「良さそうな手を詳しく調べるか、あまり調

べてない手も調べるか」というバランスをとっているので、その局面が自分の手番かどうか(その局面の手番がroot局面の手番と同じかどうか)で設定を変えれば強くなるんじゃないかと思ったので実装した。

・その局面の手番がroot局面の手番と同じ時に使う変数と、その局面の手番がroot局面の手番とは違う時に使う変数を用意して、Optunaで調整できるようにした。

・試しに軽くOptunaで調整したところ、その局面の手番がroot局面の手番と同じ時に使う変数と、その局面の手番がroot局面の手番とは違う時に使う変数では、値に結構な差が出た。(評価関数などの他の要素にも結構左右されそうだが)

- AND / OR木を使った終盤強化

・dlshogiが行っていた実験の1つ。(参考: https://tadaoyamaoka.hatenablog.com/entry/2019/08/10/212840)

- root局面の推定勝率を利用して末端の局面で呼び出すN手詰めルーチンの設定を変える

・末端の局面でN手詰めルーチンを呼び出す際、序盤～中盤ではNを小さめに、終盤では大きめに設定したいと思っている。

・「進行度」などを計算する方が良いと思うが、大変なのでとりあえずroot局面の推定勝率を使って変える事にした。

・具体的には、「root局面の推定勝率がどれだけ50%から離れているか」でNを5と7で切り替えるようにした。

・切り替えを判断する数値はOptunaで調整できるようにした。

- そこまで形成を損ねずに手を散らす機能

・dlshogiが行っていた実験の1つ(参考: https://tadaoyamaoka.hatenablog.com/entry/2021/12/05/223131)

・選手権本番では、「開始局面から32手内で、かつ定跡などにhitしていない局面」で有効になっていた

- 色々メモ化して高速化

・cshogiのzobrist_hashを利用して色々メモ化する事でそこそこ高速化した。

- その他の改造

・持ち時間制御の部分を多少改造した

7. floodgateでの計測

WCSC33版のAri Shogiを、本番と同じ構成(ハード含む)でfloodgateで対局させた。

http://wdoor.c.u-tokyo.ac.jp/shogi/view/show-player.cgi?event=LATEST&filter=floodgate&show_self_play=1&user=Ari_Shogi_WCSC33%2B56c2698facb0e087dd1d2b606a325555

Player Statistics: Ari_Shogi_WCSC33 (floodgate)

Current Status (2 weeks)

rating (Δday, Δweek)	approximated rating in shogi club 24	wins (Δday)	losses (Δday)	%	last played
2938 (+229, +2938)	5238	48 (+12)	54 (+17)	0.471	2023-05-07 16:44:29.000000000

Kristallweizen-Core2Duo-P7450	0	-	1	-	0.00	-
DQmA-wcsc33-tanuki-5600k	0	-	4	(+1)	0.00	(0.0
KfnU-wcsc33-tanuki-2000k	0	-	3	(+2)	0.00	(0.0
jbFG-wcsc33-tanuki-1400k	0	-	1	(+1)	0.00	(0.0
1-1-k-wcsc33	0	-	3	-	0.00	-
Suisho5_750_473stb_1000k	0	-	3	-	0.00	-
ZWJs-wcsc33-tanuki-500k	0	-	1	-	0.00	-
AobaZero_w4268_n_p800	0	-	2	-	0.00	-
tQgF-wcsc33-tanuki-700k	0	-	2	(+2)	0.00	(0.0
gikou2_1c	1	(+1)	3	(+1)	0.25	(0.5
NanohaWCSC33	0	-	3	-	0.00	-
Krist_483_473stb_1000k	2	-	0	-	1.00++	-
jAEz-wcsc33-tanuki-250k	2	(+2)	3	(+3)	0.40	(0.4
elmo_WCSC27_479_1000k	4	(+1)	5	(+2)	0.44	(0.3
XWrg-wcsc33-tanuki-350k	2	(+2)	0	-	1.00	(1.0
qinoa-wcsc33	4	(+3)	2	-	0.67	(1.0
Yss1000k	3	-	0	-	1.00	-
AobaZero_p200_book	1	-	1	-	0.50	-
coduck_oci_1c	10	(+2)	0	-	1.00	(1.0
coduck_pi2_600MHz_1c	4	-	0	-	1.00	-
gakkari_yuchan_neun	1	-	0	-	1.00	-
Yowa_LesserKai	3	-	0	-	1.00	-
test	7	(+1)	1	-	0.88!!!	(1.0

Suisho5_750_473stb_1000k	3452	297	304	0.494	on line	N/A
XXX_32core	3448	16	4	0.800	2023-05-02	N/A
AobaZero_w4263_n_p800	3404	68	71	0.489	2023-04-26	N/A
ZWJs-wcsc33-tanuki-500k	3394	19	35	0.352	on line	N/A
AobaZero_w4268_n_p800	3373	244	234	0.510	on line	N/A
tQgF-wcsc33-tanuki-700k	3365	22	27	0.449	on line	N/A
gikou2_1c	3300	257	327	0.440	on line	N/A
NanohaWCSC33	3269	28	33	0.460	on line	N/A
Krist_483_473stb_1000k	3243	262	369	0.415	on line	N/A
jAEz-wcsc33-tanuki-250k	3169	24	30	0.444	on line	N/A
elmo_WCSC27_479_1000k	3018	252	376	0.401	on line	N/A
XWrg-wcsc33-tanuki-350k	2991	19	30	0.388	on line	N/A
Ari_Shogi_WCSC33	2990	48	54	0.471	2023-05-07	N/A
qinoa-wcsc33	2834	50	70	0.417	2023-05-07	N/A
AobaZero_p200_book	2661	39	36	0.520	2023-05-05	N/A
Yss1000k	2637	285	323	0.469	on line	N/A
John	2633	148	135	0.523	2023-05-01	N/A
AobaZero_p100_book	2586	15	5	0.750	2023-05-04	N/A
test	2409	347	159	0.686	on line	N/A
coduck_oci_1c	2364	143	390	0.269	on line	N/A
coduck_pi2_600MHz_1c	2187	123	448	0.215	on line	N/A
gakkari_yuchan_acht	2185	4	31	0.114	on line	N/A
gakkari_yuchan_neun	2183	3	25	0.107	on line	N/A
Yowa_LesserKai	2011	52	335	0.135	2023-05-06	N/A
ayame_rps_probability_lv8	1755	1	32	0.030	2023-05-04	N/A

(画像は5/7の17:30頃に撮影)

計測の最初の方でレートが低いプレーヤーに1回負けたせいで、一度はレート2700ほどが付いていたが、100戦ほどした後は2900～3000あたりになっていた。

(4月半ばよりも強化されているはずなのに4月半ばに計測した時より弱くなっている

のはちょっとショック）

1次予選11位のきのあ将棋さんとあまり変わらないレートがついているので、WCSC33の1次予選のボーダーラインはこの辺だったのかもしれない。

（マッチングや先後の運、計測誤差や、プレーヤー同士の相性もあるので、あくまで参考値）

定跡OFFバージョン（ http://wdoor.c.u-tokyo.ac.jp/shogi/view/show-player.cgi?event=LATEST&filter=floodgate&show_self_play=1&user=Ari_Shogi_WCSC33_no_book%2B3444fe680f33a865fb00d95d3da98bb9 ）

python-dlshogi2の公開されているモデル
（ https://tadaoyamaoka.hatenablog.com/entry/2021/12/29/122554 の件で触れられている"checkpoints/checkpoint.pth"の事 ）を搭載したバージョン
（ http://wdoor.c.u-tokyo.ac.jp/shogi/view/show-player.cgi?event=LATEST&filter=floodgate&show_self_play=1&user=Ari_Shogi_WCSC33_pydl2_model%2Be6949611b754200335ee5b1948a69ec3 ）

についても計測を行った。

XXX_32core	3442	16	4	0.800	2023-05-02	N/A
tQgF-wcsc33-tanuki-700k	3435	26	31	0.456	2023-05-07	N/A
ZWJs-wcsc33-tanuki-500k	3406	21	40	0.344	2023-05-07	N/A
AobaZero_w4263_n_p800	3378	40	47	0.460	2023-04-26	N/A
AobaZero_w4268_n_p800	3366	256	250	0.506	2023-05-08	N/A
NanohaWCSC33	3331	50	61	0.450	2023-05-08	N/A
gikou2_1c	3300	249	329	0.431	on line	N/A
Ari_Shogi_WCSC33_no_book	3282	11	12	0.478	2023-05-08	N/A
Krist_483_473stb_1000k	3236	252	379	0.400	on line	N/A
jAEz-wcsc33-tanuki-250k	3164	27	35	0.435	2023-05-07	N/A
Ari_Shogi_WCSC33_pydl2_model	3148	6	8	0.433	2023-05-08	N/A
XWrg-wcsc33-tanuki-350k	3057	24	34	0.414	2023-05-07	N/A
elmo_WCSC27_479_1000k	3019	241	384	0.386	2023-05-08	N/A
Ari_Shogi_WCSC33	3016	49	54	0.476	2023-05-07	N/A
qinoa-wcsc33	2891	66	104	0.389	on line	N/A
AobaZero_p200_book	2670	39	36	0.520	2023-05-05	N/A
John	2640	122	116	0.513	2023-05-01	N/A
Yss1000k	2635	266	341	0.438	on line	N/A
AobaZero_p100_book	2580	15	5	0.750	2023-05-04	N/A
coduck_r6c	2541	6	19	0.240	on line	N/A
test	2503	353	146	0.707	2023-05-08	N/A
coduck_oci_1c	2361	136	397	0.256	2023-05-08	N/A
gakkari_yuchan_neun	2233	13	55	0.191	on line	N/A
gakkari_yuchan_acht	2164	11	66	0.143	2023-05-08	N/A
coduck_pi2_600MHz_1c	2127	110	459	0.193	2023-05-08	N/A

(画像は5/8の21:20頃に撮影)

何故か定跡OFFの方が強かった。

python-dlshogi2のモデルを搭載したバージョンは、モデルファイル以外は定跡OFF版と同じなのだが、定跡OFF版よりレートが低かった。

パラメータを調整しなかったのもあるが、モデルの大きさによる速度の低下が原因

でレートが下がっていると思われる。

(注.floodgateでの計測はノイズが大きいうえに、今回は対局数も非常に少ないのでそこまで信頼できる値ではない。なのであくまで参考値程度に見てほしい)

ちなみに、選手権前にfloodgateで対局していた、

SplatBrella_wcsc33_1(http://wdoor.c.u-tokyo.ac.jp/shogi/view/show-player.cgi?event=LATEST&filter=floodgate&show_self_play=1&user=SplatBrella_wcsc33_1%2B23d38970923b1165915709e05326d2c6)

SplatBrella_wcsc33_2(http://wdoor.c.u-tokyo.ac.jp/shogi/view/show-player.cgi?event=LATEST&filter=floodgate&show_self_play=1&user=SplatBrella_wcsc33_2%2B1da785ade1299722ea5094d2ff0c4dbe)

は、定跡をOFFにして、「そこまで形成を損ねずに手を散らす機能」を32手目まで有効にしたAri Shogiである。

その他

・来年の選手権は可能であればリアル会場に行きたいと思っている。(無理かもしれないが)

・Ari Shogiは、6/30、7/1に開催予定の「第4回世界将棋AI電竜戦TSEC指定局面戦」にも参加する予定。

リンクなど

GitHubアカウント: https://github.com/YuaHyodo

Kaggleアカウント: https://www.kaggle.com/yuahyodo

lichessアカウント: https://lichess.org/@/UA_2419006

以下、おまけ

注.おまけの文章はダラダラ適当に書いている上に推敲もほぼしていないので、文章がおかしいところが通常より多い。

アイディアとして出たが、まだちゃんと試してないものを書く。

既出のものも多いかもしれない。

ゴミみたいなアイディアばかりだと思うが、(私だけでは試しきれないので)気になった人はぜひ試してほしい。(可能であれば結果も教えてほしい)

一覧

案1.複数入力のニューラルネット

案2.ニューラルネットの継ぎ足し学習

案3.外部ソフトを用いたMultiPonder

案4.合議の案いろいろ

案5.生成系のAIを使った局面生成とその利用法に関して

案6.アンチコンピュータ戦略を自動で見つけてそれを自動で潰すシステム

案7.頓死を誘発する局面を目指すAI

案1.複数入力のニューラルネット

Self-Attentionを使ったニューラルネットを評価関数に使おうという試みがいくつかある。

Self-Attentionには、CNNの「遠くの駒の関係を学習するためには層がある程度深くなければならない」という弱点が無いという強みがあると思うのだが、未だにResNetをはじめとするCNN系は超えれていない。(ように見える)

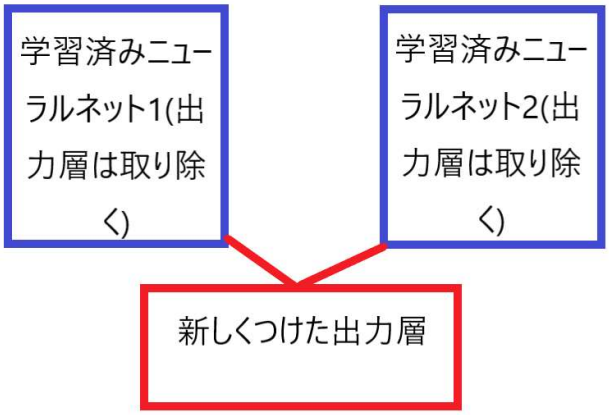
これは、CNNにはSelf-Attention系にはない強みがあるからだと思われる。

そんな事を考えていた私はふと、「アンサンブルすれば良いのでは」と思った。

Kaggleにちょろっと参加してアンサンブルの威力を知った自分は、近い将来、将棋AIはアンサンブルを多用するようになると思っているし、そう思っている人も結構いると思う。

「Self-Attention系にない強みを持つCNN系」と「CNN系にはない強みを持つSelf-Attention系」を混ぜれば(強いかどうかはともかく)精度自体は上がると思う。

CNN系とSelf-Attention系のモデルを別で学習させ、それぞれの出力層を取っ払って新しい出力層をくっつけて1つのニューラルネットとして学習させる。(参考図_1)



(学習の際、学習済みニューラルネットの部分の重みは固定した方が良さそう)

こうする事によって強いかどうかはともかく精度の高い評価関数ができると思う。

「精度は非常に高いが対局には使えないレベルで遅い」という評価関数にも、定跡や教師データ生成などの用途はあると思うので、(Self-Attention云々関係なく)アンサンブルは今後重要になってくると思う。

(そうなるとアンサンブルの素材を求めたチーム合併が行われるようになるかも)

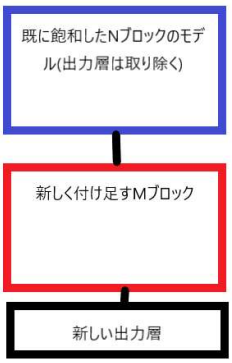
案2.ニューラルネットの継ぎ足し学習

上位陣のDL系将棋AIのニューラルネットは今後も巨大化していくのではないかと自分は考えている。(軽いモデルを使ったチームもある程度あると思うが)

しかし、巨大なニューラルネットを学習させるのは難しいし、かつ物凄いリソースが必要だ。

持っているリソースにもよると思うが「この評価関数の学習が失敗したら次の大会は終わる」というようなギャンブルをする必要も出てくるかもしれない。

そんなギャンブルのリスクを低減できないか考えていた時、ふと「徐々にネットワークを大きくしていけば良いのでは」と思った。(プログレッシブGANかなんかにも似た案があった気がする)



まず、Nブロックのモデルを用意して、飽和するまで学習させる。

飽和したら、出力層を取っ払ってNブロックの後にMブロックと新しい出力層を付けたし、また飽和するまで学習させる。

これをリソースがある限り繰り返す。

大会に出るときは、大会前に学習済みのやつのなかから最強のやつを選べばOK。

こうする事で、より低リスク・低コスト(&低い心理的負担)で巨大な評価関数を学習させる事ができるかもしれないと自分は思っている。(上手くいくかは知らん)

案3.外部ソフトを用いたMultiPonder

MultiPonderの改良版。他のソフトも利用する事でhit率の向上を目指す。

Ponderの相手の予想手を複数用意するとした場合、MultiPVなどで出す方法も考えられるが、今回は複数のソフトを用いる事にする。

3個の候補手を決める場合、通常のやつだと、候補1:本体のAIが考える最善手, 候補2:本体のAIが考える次善手,候補3:本体のAIが考える3番目の手

とするが、このシステムでは、

候補1:本体のAIが考える最善手, 候補2: 外部エンジンAが考える最善手, 候補3: 外部エンジンBが考える最善手

という様にする。

本体のAIがあまり考えない応手も予想できるので、「特定の相手には全然Ponderが

刺さらない」という事が減るかも。

案4. 合議の案いろいろ

合議系の将棋AIの案をつらつら書いておく。

- DL系将棋AIを極端な性能にして、それを他のAI(NNUE系とか)で補う合議が強いかもしれない。

  ・現状言われている「DL系で序盤 / 中盤に優位を築き、NNUE系の終盤力で押しつぶす」というのをもっと極端にしたい

  ・DL系のAIを超極端な性能(弱点作ってでもどこかに特化するイメージ)にして、その弱点をNNUE系で補うようなのが良いかも。

  ・ピーキーな奴らを集めて弱点を補わせる。それでも埋まらない穴はオールラウンダー(悪く言うと器用貧乏)で埋める。という思想。

- 「単体ではそうでもないけど、他のと混ぜた時の効果は絶大」というAIや評価関数はあると思うので、それを活かしたい。

  ・探すの難しそう。あらゆるパターンを試すリソース / 運 / 開発者の勘の勝負になるかも。

  ・そういうAIは意外なところにもあるかもしれない。(少し古い時代のAIの中に有用なものが眠ってたりして)

- 学習する合議AI

  ・普通の評価関数の出力層を取っ払って転移学習させるとか？

  ・[局面, [AI_1の勝率, AI_2の勝率, AI_3の勝率]]というデータセットを作って、転移学習させる？(ゼロからやるよりは低コストだが、それでも大変そう)

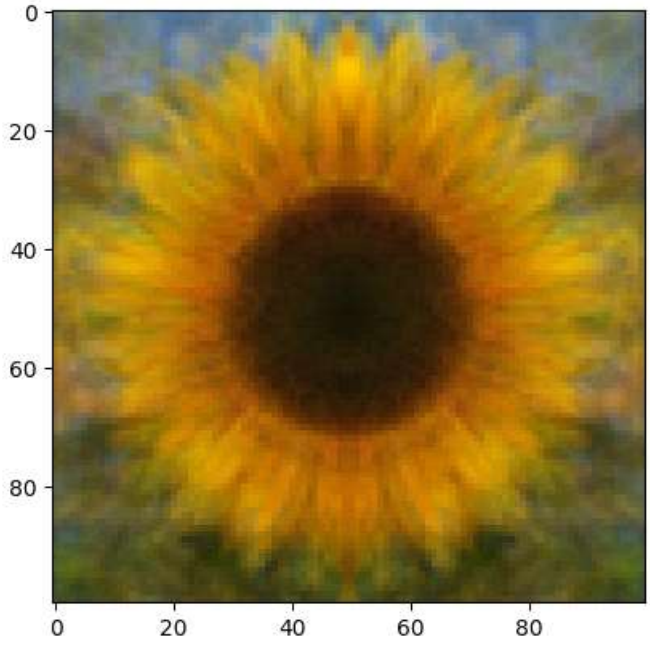
  ・合議に使うAIや、対局相手のAIの選定が大変そう。

案5. 生成系のAIを使った局面生成とその利用法に関して

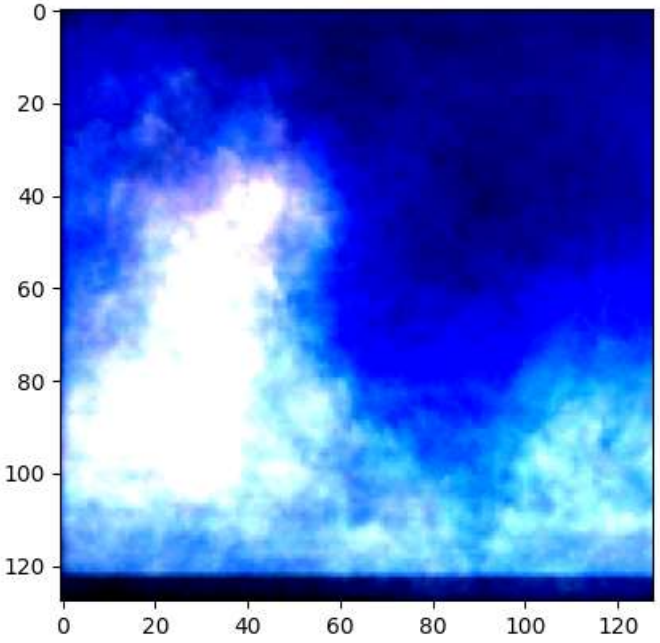
しばらく前からGANの実験を色々やっている。

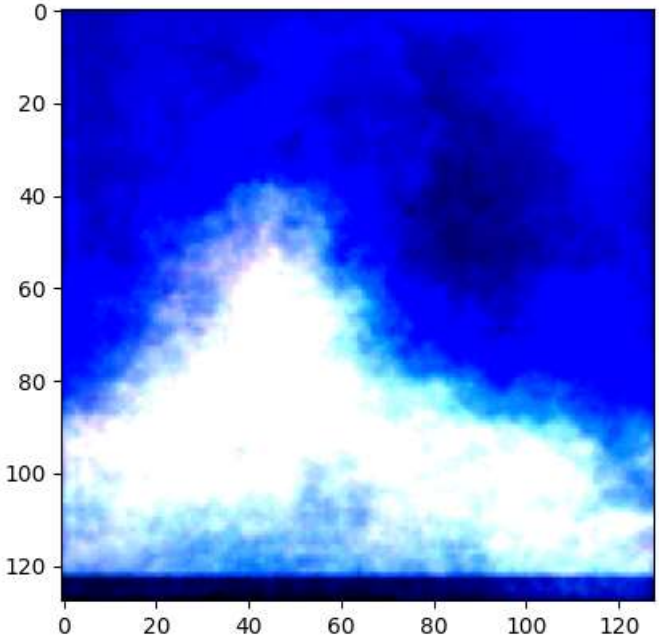
空き時間にやっているので進捗はそんなでもないが、一応進んではいる。

［旧バージョンで生成した向日葵］



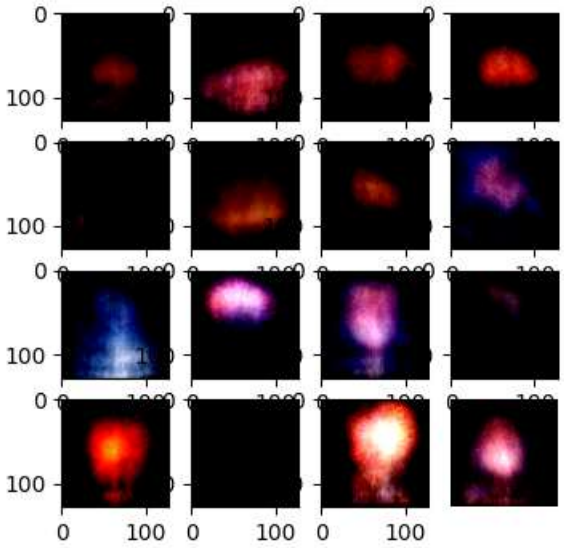
［新バージョンで生成した積乱雲］





（データセットの画像の下部に要らん部分があるせいで下に要らん部分ができてしまっている）

［新バージョンで生成した花火］



しばらく前に「将棋やオセロの盤面を生成する」というのを試していた。

当時は失敗したが、今ならいけるかもしれない。

自分は、将棋の局面の生成に関しては

1:画像のように扱う

2:文字列として扱う

というのを考えている。

GAN以外にも、生成系の言語モデルや画像生成AIの技術が転用できると思うが、あまり詳しくないのでここでは詳細は触れない。

私は、局面生成AIの使い道として

1: 教師データの生成(高リスク)

2: 教師データの元の生成(低リスク)

などを考えている。

(1は詳細を考えれてないので置いておいて)2は割と実用的なんじゃないかと考えている。

ここでいう「教師データの元」というのは「教師データに収録する局面(ラベル付け

などは既存の将棋AIで行う)」や「教師データ生成用の対局の開始局面」を指している。

(種類を問わず)合法な局面を生成できるAIを学習させた後、「互角な中盤局面」や「入玉が絡む局面」や「振り飛車に関する局面」などの「特定要素を持った局面」を生成できるAIを学習し、(色々なチェックをした後)そこから対局させたりして教師データを生成する、というのを考えている。(上手くいくかは知らん)

(ちなみに、私は教師データの局面に関しては別に開始局面から合法的にたどりつく事ができる必要はないと考えている。)

案6.アンチコンピュータ戦略を自動で見つけてそれを自動で潰すシステム

少し前、囲碁AI界隈でかなり影響が大きいアンチコンピュータ戦略が発見されたという話を聞いた。

(実行には多少のコツがいるらしいが)上手くいけば様々な強豪AIが錯覚を起こして負けるらしく、修正もそこそこ大変なようだ。

この話を聞いて「アンチコンピュータ戦略(または脆弱性)を自動で見つけて自動で潰すシステムは作れないのか」と思った人も多いと思う。

私もそう思って色々考えたので、それについて考えた事とかをメモっておく。(まぁ、こんなリソースを投じてまで自動化する事でもないと思うが。)

・詳細は忘れたが、KataGOの脆弱性をつくようにAIを学習させるという論文があった気がする。詳細を知らないので何とも言えないが応用できるかもしれない。

・AlphaStarにも少し違うが似たような仕組みがあった気がする。あれは戦略同士の相性問題に関するものだったが、これも応用できるかも。

・「本体のAIに対しては脆弱性をつかなければ勝てないレベル」にネットワークサイズや探索局面数などを調整すると良いかもしれない。

・勝敗以外に「終局までの手数」や「1局を通した評価値の変化」なども考慮して学習させると良いかも。

・アンチコンピュータ戦略や脆弱性を見つけるAIができれば、対策用の教師データを生成して学習するだけでいずれ解決しそう。

この件についていろいろ考えていたら第2回TSECの直前にAN Shogiが起こした1件を

思い出した。「強いAI(その時は技巧2を使った)の攻撃をより長く耐える」という方向に学習させたのだが、行き着いた先は「特定の動きをする事で相手のAIに穴熊を組ませて攻めるのを遅らせる」というところだった。

その時はどうしようもなかったのでバックアップを復元して大会に出した。

案7.頓死を誘発する局面を目指すAI

将棋には「頓死」という用語があるらしい。

よく知らないが、だいたい「最善手を指せば詰みを回避できる局面でミスして詰まされる」という状況の事らしい。

そして、流行りのDL系将棋AIは割と頓死しやすいらしい。

そこで思った、「頓死を誘発するような局面を目指すAIは、DL系に対する有効な武器になるんじゃないか」と。

そこで、頓死を誘発するような局面を目指すAIの仕組みと、その利用法考えてみた。

※ここでは、「頓死」を「合法手の中に1つ以上詰みを回避できる(注.現実的な思考時間と棋力での話)手がある局面で、詰みを回避できなくなる手を指してしまう」と定義して考えていく。

1.全合法手中、詰みが回避できない手(不正解の手)が占める割合

2.勝敗が確定してから終局するまでの手数

3.候補手の数

などの情報を使って「その局面でどれくらい頓死しやすいかの指標、〝頓死度〟」を計算する評価関数を作り、それを探索部と組み合わせる事で、頓死を誘発するAIを作れると思う。

あとは、これを合議に加えたりして「適切な場面(相手がDL系で、こちらが劣勢の時など)に頓死を誘発する」「相手の〝頓死度〟が高くなる局面」になるように手を指していくようにすれば良いんじゃないかと思っている。