手抜きについて

手抜きチーム

2021 年 3 月 28 日

手抜きは CSA プロトコルで対局を行うコンピュータ将棋プログラムです。開発者らが将棋プログラムの仕組みを理解するために作っています。
リポジトリ：`https://github.com/hikaen2/tenuki-d`

名前の由来

かっこいい名前が思いつかなかったため『手抜き』にしました。

特長

- 初段〜二段くらいの棋力
- αβ探索
- D 言語で実装

開発者

鈴木太朗：自転車と合唱と Emacs が好きです。誰か df-pn 教えてください。Twitter: @hikaen2
玉川直樹：好きなコードは add9 です。将棋ウォーズ 2 段。Twitter: @Neakih_kick

1 今年の目標

NNUE を理解する。

2 使用ライブラリ

- 『どうたぬき』(tanuki- 第 1 回世界将棋 AI 電竜戦バージョン) の評価関数ファイル nn.bin[*1]

[*1] `https://github.com/nodchip/tanuki-/releases/tag/tanuki-denryu1`

表 1　どうたぬきの nn.bin の内訳

No.	内容	開始位置	サイズ (バイト)	備考
1	version	0	4	0x7af32f16
2	hash	4	4	0x3e5aa6ee
3	size	8	4	0x000000b2(architecture のサイズ．可変)
4	architecture	12	178	-
5	header	190	4	-
6	FeatureTransformer.biases	194	512	$\vec{\boldsymbol{b}}_1$: int16_t が 256 個
7	FeatureTransformer.weights	706	64,198,656	$\boldsymbol{W}_1$: int16_t が 256×81×1548 個
8	header	64,199,362	4	-
9	HiddenLayer1.biases	64,199,366	128	$\vec{\boldsymbol{b}}_2$: int32_t が 32 個
10	HiddenLayer1.weights	64,199,494	16,384	$\boldsymbol{W}_2$: int8_t が 32×512 個
11	HiddenLayer2.biases	64,215,878	128	$\vec{\boldsymbol{b}}_3$: int32_t が 32 個
12	HiddenLayer2.weights	64,216,006	1,024	$\boldsymbol{W}_3$: int8_t が 32×32 個
13	OutputLayer.biases	64,217,030	4	$\vec{\boldsymbol{b}}_4$: int32_t が 1 個
14	OutputLayer.weights	64,217,034	32	$\boldsymbol{W}_4$: int8_t が 1×32 個
			合計: 64,217,066	

3　メモ：nn.bin の構造について

どうたぬきの評価関数ファイル nn.bin は 64,217,066 バイトある。その内訳を表 1 に示す。値はすべてリトルエンディアンで格納されている。

表 1 の No.4: architecture には人間に可読な形式でニューラルネットワークの構造が書かれている。どうたぬきの場合は

```
Features=HalfKP(Friend)[125388->256x2],Network=AffineTransform[1<-32](ClippedReLU[32](
AffineTransform[32<-32](ClippedReLU[32](AffineTransform[32<-512](InputSlice[512(0:512)])))))
```

と書かれている。この文字列の長さはニューラルネットワークの構造によって変わるので，その長さ（バイト数）が No.3 size に格納されている。どうたぬきの場合は 0x000000b2 = 178 である。

No.7: FeatureTransformer.weights ($\boldsymbol{W}_1$) は 256×125388 行列である：

$$\boldsymbol{W}_1 = \begin{pmatrix} w_{1(0,0)} & \cdots & w_{1(0,125387)} \\ \vdots & \ddots & \vdots \\ w_{1(255,0)} & \cdots & w_{1(255,125387)} \end{pmatrix}$$

この行列は 125388 次元の特徴ベクトルを 256 次元ベクトルに変換する。ここで $125388 = 81 \times 1548$ は KP（K:自玉と P:玉以外の駒 の組み合わせ）が取りうる場合の数である。81 は K の取りうる場合の数（玉が 1 一から 9 九まで。盤面のアドレスを図 1 に示す），1548 は P の取りうる場合の数（内訳を表 2 に示す）である。この特徴ベクトルは駒のあるところが 1，駒のないところが 0 になる二値ベクトルである。将棋の駒は玉を除くと 38 枚あるので，特徴ベクトルは 125388 次元のうち 38 ヶ所だけが 1 で，のこりがすべて 0 のベクトルとなる。特徴ベクトルの具体例については次節に記す。

9	8	7	6	5	4	3	2	1	
72	63	54	45	36	27	18	9	0	一
73	64	55	46	37	28	19	10	1	二
74	65	56	47	38	29	20	11	2	三
75	66	57	48	39	30	21	12	3	四
76	67	58	49	40	31	22	13	4	五
77	68	59	50	41	32	23	14	5	六
78	69	60	51	42	33	24	15	6	七
79	70	61	52	43	34	25	16	7	八
80	71	62	53	44	35	26	17	8	九

図 1　盤面のアドレス

$\boldsymbol{W}_1$ は nn.bin に column-major で格納されている。すなわち最初の値は $w_{(0,0)}$，次の値は $w_{(1,0)}$，$\cdots$ という順番で格納されている。一方，後述する $\boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{W}_4$ は row-major で格納されている。すなわち最初の値は $w_{(0,0)}$，次の値は $w_{(0,1)}$，$\cdots$ という順番で格納されている。$\boldsymbol{W}_1$ とそれ以外で格納順が異なることに注意されたい。

No.6: FeatureTransformer.biases ($\vec{\boldsymbol{b}}_1$) は 256 次元ベクトルである：

$$\vec{\boldsymbol{b}}_1 = \begin{pmatrix} b_{1(0)} \\ \vdots \\ b_{1(255)} \end{pmatrix}$$

このベクトルは $\boldsymbol{W}_1$ の出力である 256 個のニューロンに足すバイアスである。

$\boldsymbol{W}_1$ に KP の 125388 次元の特徴ベクトル $\vec{\boldsymbol{x}}$ を掛けて，$\vec{\boldsymbol{b}}_1$ を足すと，256 次元のベクトル $\vec{\boldsymbol{y}}$ が得られる：

$$\underbrace{\begin{pmatrix} y_0 \\ \vdots \\ y_{255} \end{pmatrix}}_{\vec{\boldsymbol{y}}} = \underbrace{\begin{pmatrix} b_{1(0)} \\ \vdots \\ b_{1(255)} \end{pmatrix}}_{\vec{\boldsymbol{b}}_1} + \underbrace{\begin{pmatrix} w_{1(0,0)} & \cdots & w_{1(0,125387)} \\ \vdots & \ddots & \vdots \\ w_{1(255,0)} & \cdots & w_{1(255,125387)} \end{pmatrix}}_{\boldsymbol{W}_1} \times \underbrace{\begin{pmatrix} x_0 \\ \vdots \\ x_{125387} \end{pmatrix}}_{\vec{\boldsymbol{x}}}$$

この 256 次元ベクトル $\vec{\boldsymbol{y}}$ を先手分と後手分あわせて 512 次元にしたベクトルを $\vec{z}_1$ とする。$\vec{z}_1$ を No.10: HiddenLayer1.weights ($\boldsymbol{W}_2$：32×512 行列) と No.9: HiddenLayer1.biases ($\vec{\boldsymbol{b}}_2$：32 次元ベクトル) に入力すると 32 次元ベクトル $\vec{z}_2$ が得られる：

$$\underbrace{\begin{pmatrix} z_{2(0)} \\ \vdots \\ z_{2(31)} \end{pmatrix}}_{\vec{z}_2} = \underbrace{\begin{pmatrix} b_{2(0)} \\ \vdots \\ b_{2(31)} \end{pmatrix}}_{\vec{\boldsymbol{b}}_2} + \underbrace{\begin{pmatrix} w_{2(0,0)} & \cdots & w_{2(0,511)} \\ \vdots & \ddots & \vdots \\ w_{2(31,0)} & \cdots & w_{2(31,511)} \end{pmatrix}}_{\boldsymbol{W}_2} \times \sigma \underbrace{\begin{pmatrix} z_{1(0)} \\ \vdots \\ z_{1(512)} \end{pmatrix}}_{\vec{z}_1}$$

表2　P の取りうる場合の数

No.	内容	開始序数	場合の数
1	味方の持駒の歩の 0 枚目〜18 枚目	0	19
2	相手の持駒の歩の 0 枚目〜18 枚目	19	19
3	味方の持駒の香の 0 枚目〜4 枚目	38	5
4	相手の持駒の香の 0 枚目〜4 枚目	43	5
5	味方の持駒の桂の 0 枚目〜4 枚目	48	5
6	相手の持駒の桂の 0 枚目〜4 枚目	53	5
7	味方の持駒の銀の 0 枚目〜4 枚目	58	5
8	相手の持駒の銀の 0 枚目〜4 枚目	63	5
9	味方の持駒の金の 0 枚目〜4 枚目	68	5
10	相手の持駒の金の 0 枚目〜4 枚目	73	5
11	味方の持駒の角の 0 枚目〜2 枚目	78	3
12	相手の持駒の角の 0 枚目〜2 枚目	81	3
13	味方の持駒の飛の 0 枚目〜2 枚目	84	3
14	相手の持駒の飛の 0 枚目〜2 枚目	87	3
15	味方の歩が１一〜９九にいる	90	81
16	相手の歩が１一〜９九にいる	171	81
17	味方の香が１一〜９九にいる	252	81
18	相手の香が１一〜９九にいる	333	81
19	味方の桂が１一〜９九にいる	414	81
20	相手の桂が１一〜９九にいる	495	81
21	味方の銀が１一〜９九にいる	576	81
22	相手の銀が１一〜９九にいる	657	81
23	味方の金 *が１一〜９九にいる	738	81
24	相手の金 *が１一〜９九にいる	819	81
25	味方の角が１一〜９九にいる	900	81
26	相手の角が１一〜９九にいる	981	81
27	味方の馬が１一〜９九にいる	1062	81
28	相手の馬が１一〜９九にいる	1143	81
29	味方の飛が１一〜９九にいる	1224	81
30	相手の飛が１一〜９九にいる	1305	81
31	味方の龍が１一〜９九にいる	1386	81
32	相手の龍が１一〜９九にいる	1467	81
			合計: 1548

* と・成香・成桂・成銀は金として扱う

ここで σ はベクトルの各要素を 0〜127 の範囲にクリップする活性化関数 clipped ReLU である。具体的には 0 以下の値は 0 にクリップする。127 以上の値は 127 にクリップする。

次に $\vec{z}_2$ を No.12: HiddenLayer2.weights ($\boldsymbol{W}_3$：32×32 行列) と No.11: HiddenLayer2.biases ($\vec{\boldsymbol{b}}_3$：32 次元ベクトル) に入力すると 32 次元ベクトル $\vec{z}_3$ が得られる：

$$\underbrace{\begin{pmatrix} z_{3(0)} \\ \vdots \\ z_{3(31)} \end{pmatrix}}_{\vec{z}_3} = \underbrace{\begin{pmatrix} b_{3(0)} \\ \vdots \\ b_{3(31)} \end{pmatrix}}_{\vec{\boldsymbol{b}}_3} + \underbrace{\begin{pmatrix} w_{3(0,0)} & \cdots & w_{3(0,31)} \\ \vdots & \ddots & \vdots \\ w_{3(31,0)} & \cdots & w_{3(31,31)} \end{pmatrix}}_{\boldsymbol{W}_3} \times \sigma \underbrace{\begin{pmatrix} z_{2(0)} \\ \vdots \\ z_{2(31)} \end{pmatrix}}_{\vec{z}_2}$$

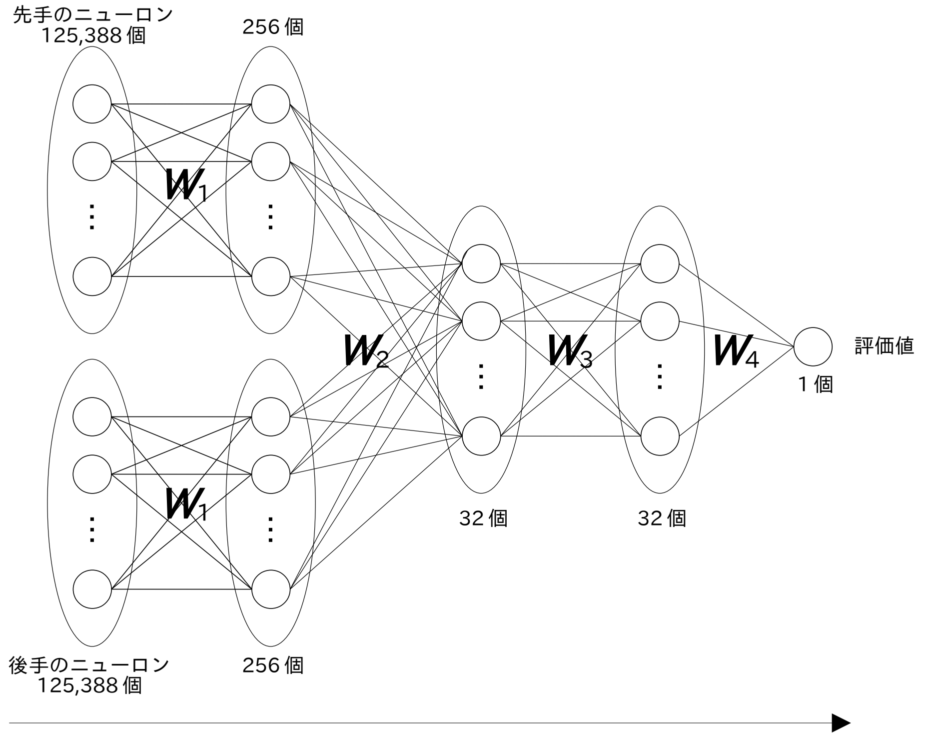


図 2　ニューラルネットワークの図（先手の評価値を求めるとき）

最後に $\vec{z}_3$ を No.14: OutputLayer.weights ($\boldsymbol{W}_4$：1×32 行列) と No.13: OutputLayer.biases ($\vec{\boldsymbol{b}}_4$：1 次元ベクトル) に入力すると 1 次元ベクトル $\vec{z}_4$ が得られる。これが評価値である：

$$\underbrace{\begin{pmatrix} z_{4(0)} \end{pmatrix}}_{\vec{z}_4} = \underbrace{\begin{pmatrix} b_{4(0)} \end{pmatrix}}_{\vec{\boldsymbol{b}}_4} + \underbrace{\begin{pmatrix} w_{4(0,0)} & \dots & w_{4(0,31)} \end{pmatrix}}_{\boldsymbol{W}_4} \times \sigma \underbrace{\begin{pmatrix} z_{3(0)} \\ \vdots \\ z_{3(31)} \end{pmatrix}}_{\vec{z}_3}$$

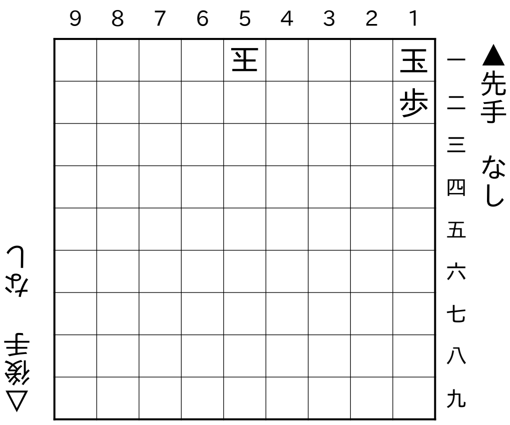
以上の計算を図に表すと図 2 となる（バイアス $\vec{\boldsymbol{b}}_n$ と活性化関数 σ の記載は省略している)。

以上の計算を行うサンプルプログラムを `https://github.com/hikaen2/nnue-test` に作成した（D 言語)。

3.1　特徴ベクトルの具体例

例 1

以下の局面を考える。



先手の特徴ベクトルを考える。先手玉が１一にいる。図 1 を見ると１一のアドレスは 0 なので，このときの K の序数は 0 になる。１二に（先手から見て）味方の歩がいる。表 2 を見ると味方の歩の序数は 90 から始まっている。これに１二のアドレスである 1 を足すと $90+1=91$ になり，１二にいる味方の歩を表現できる。K と P を合わせると $0\times 1548+91=91$ となる。これは 125388 次元ベクトルの（0 から数えて）91 番目の要素に 1 を立てることを意味する。したがって先手の特徴ベクトルはこのようになる：

$$\vec{\boldsymbol{x}}=\begin{pmatrix} x_0 & = & 0 \\ \vdots & & \\ x_{90} & = & 0 \\ x_{91} & = & 1 \\ x_{92} & = & 0 \\ \vdots & & \\ x_{125387} & = & 0 \end{pmatrix}$$

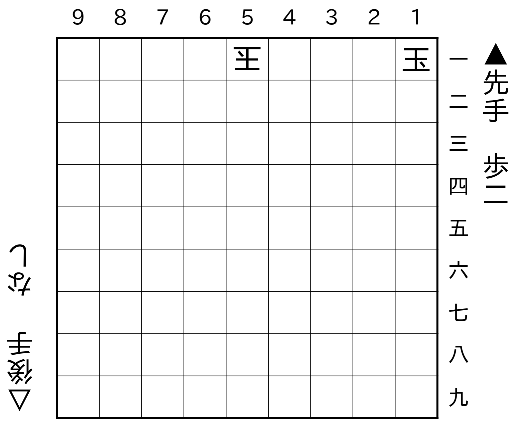
次に後手の特徴ベクトルを考える。後手の特徴ベクトルを考えるときは盤面を 180 度回転する。すると５一の後手玉は５九の位置にいることになる。図 1 を見ると５九のアドレスは 44 なので，このときの K の序数は 44 になる。つぎに１二に（後手から見て）相手方の歩がいる。これは盤面を 180 度回転すると９八の位置にいることになる。表 2 を見ると相手方の歩の序数は 171 から始まっている。これに９八のアドレスである 79 を足すと $171+79=250$ になる。K と P を合わせると $44\times 1548+250=68362$ となる。これは 125388 次元ベクトルの（0 から数えて）68362 番

目の要素に 1 を立てることを意味する。したがって後手の特徴ベクトルはこのようになる：

$$\vec{\boldsymbol{x}} = \begin{pmatrix} x_0 & = & 0 \\ \vdots & & \\ x_{68361} & = & 0 \\ x_{68362} & = & 1 \\ x_{68363} & = & 0 \\ \vdots & & \\ x_{125387} & = & 0 \end{pmatrix}$$

例 2

以下の局面を考える。



先手の特徴ベクトルを考える。先手玉が 1 一にいる。図 1 を見ると 1 一のアドレスは 0 なので，このときの K の序数は 0 になる。味方の持ち駒に歩が 2 枚ある。表 2 を見ると味方の持ち駒の歩の序数は 0 から始まっている。そこで 1 枚目の歩は $0+1=1$ となる。2 枚目の歩は $0+2=2$ となる：

$$\vec{\boldsymbol{x}} = \begin{pmatrix} x_0 & = & 0 \\ x_1 & = & 1 \\ x_2 & = & 1 \\ x_3 & = & 0 \\ \vdots & & \\ x_{125387} & = & 0 \end{pmatrix}$$

4　私のための Q&A

Q1　NNUE は KP を入力とするニューラルネットワークということですか？

A1　そうです。厳密に言えばどうたぬきの NNUE についてはそうです。これは KP 以外を入力とする NNUE も考えられるということです。

Q2 なんで KPP より強いんですか？

A2 KPP の線形和より，KP の非線形和のほうが表現力があるということだと思います。言い換えると線形和がよほどよろしくないのかもしれません。たとえばカツカレーがおいしいのはカツがおいしくて，なおかつカレーがおいしいからと考えることもできますが，だからといっておいしいプリンをおいしいカレーに入れてもおいしくなるとは限らないということだと思います。

Q3 線形和・非線形和ってなんですか？

A3 雑な理解として，線形和はとりあえず全部足したやつだと思っています。非線形和はとりあえず全部足したやつじゃない足し方をしたやつだと思っています。

Q4 ベクトルってなんですか？

A4 プログラマとしては雑な理解として配列だと思っています。だから 125388 次元のベクトルは 125388 要素の配列だと思っています。

Q5 行列ってなんですか？

A5 2 次元配列だと思っています。

Q6 バイアス ($\vec{\boldsymbol{b}}$) ってなんですか？

A6 1 次関数で言うところの切片だと思っています。つまり $y = ax + b$ の b にあたるものだと思っています。

Q7 線形代数はなにが線形なのですか？

A7 雑な理解でいうと，ベクトルを行列によって変換したときの変換され具合が線形（1 次関数によって計算される）なのだと思います。たとえば図形を行列で変形すると，どの座標の点も一様に変形されます。突然図形がぐにゃぐにゃになったりはしないということです。

Q8 NNUE のニューラルネットワークの学習ってどうやるんですか？

A8 わかりません。これから調べます。

5 参考文献

- 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，https://www.apply.computer-shogi.org/wcsc28/appeal/the_end_of_genesis_T.N.K.evolution_turbo_type_D/nnue.pdf