平成 30 年度卒業研究
AI を用いたゲームの自動プレイシステムの開発

函館工業高等専門学校

生産システム工学科　情報コース　5 年

21 番　中村　柊太

指導教員　東海林　智也

目次

第1章 序論

第1節　英文アブストラクト

Machine learning, which is one of the AI technologies, has penetrated various fields of industry. When solving problems involving random elements by machine learning, reinforcement learning is generally used. However, since reinforcement learning has a complicated algorithm, execution speed of the program may decrease. In this study, we aim to develop a learning system that is fast, lightweight and comparable to reinforcement learning by using deep learning instead of reinforcement learning. In conclusion, learning is not perfect, and it is difficult to compete with reinforcement learning without further devising.

Key words : deep learning , reinforcement learning, execution speed

第2節　研究目的

本研究の目的は、深層学習のアルゴリズムを用いて高速・軽量でかつ強化学習に匹敵する機械学習システムを開発することである。

また、身近にあるランダム要素のあるゲームを自動プレイさせることで、ランダム要素の強い問題に深層学習を応用する方法を考察する。

第3節　研究背景

今日、AI 技術の一つである機械学習は、産業の様々な分野に浸透している。しかしながら、現実世界の問題には単純な数値計算では解決できないランダム要素が含まれた問題も数多く存在する。

そういった問題を機械学習によって解決する場合、強化学習が一般的であるが、強化学習は、その複雑さからプログラムの実行速度が低下する可能性がある。

そこで、深層学習の機能を拡張し、ランダム性に強く、高速・軽量な機械学習システムを開発できないかと考えた。

第4節　開発環境

研究室 PC

- CPU：Intel(R)Core(TM) i5-3470 CPU @ 3.20GHz
- 実装メモリ・RAM：4GB
- OS：Windows 10 Education

開発環境：Visual Studio 2017

使用言語：C++

AI ライブラリ：tiny-dnn

第2章 関連技術

第1節　　深層学習

　深層学習とは、ニューラルネットワークの階層を深めた機械学習の手法である。ニューラルネットワークとは、人間の脳の神経系を模した数値計算のモデルである。

　教師データが存在し、一度学習が済んでしまえばそのデータを参照し続けるために、強化学習に比べ軽量である。

第2節　　強化学習

　強化学習とは、エージェントの行動によって報酬を獲得し、繰り返し学習することで報酬を最大化する機械学習の手法である。

　教師データはなく、内部報酬を自動生成する。

　一般的には再帰関数を頻繁に用いるため、メモリを大量に消費する可能性がある。

第3節　tiny-dnn

tiny-dnn は C++向けヘッダオンリーの深層学習ライブラリである。導入コストが他の深層学習ライブラリよりも低く、フォルダのコピーのみで導入が完了する利点がある。言語は C++と python に対応している。

外部依存がないので、C++をサポートしているコンパイラがある開発環境であれば動作する。

本研究では深層学習ライブラリとしてこの tiny-dnn を使用する[1][2]。

第3章 プログラムの開発

第1節 プログラムの概要

本研究では、ランダム性を持つ単純なゲームとして、プレイヤーと AI の対戦型三目並べを取り扱うこととした。ゲームのプレイ画面のイメージを図 1 に示す。

三目並べは、互いに最善手を打ち続けると、必ず引き分けになる法則がある。したがって、このゲームの学習の目標を「引き分けにすること」とし、現在の盤面の状態と、どこに石を置けばいいかの組をニューラルネットワークに教師データとして与え、学習させる。その後、学習済みモデルにランダムな盤面の状態を与え、どこに石を置くかを出力として得るプログラムを開発する。

ニューラルネットワークは、入力層 9 ユニット、中間層 300 ユニット、出力層 1 ユニットとする 3 層構造ニューラルネットワークと、入力層 9 ユニット、中間層 1,中間層 2 をそれぞれ 100 ユニット、出力層 1 ユニットとする 4 層構造ニューラルネットワークの 2 つを併用する。

このニューラルネットワークに教師データを与えて学習済みモデルを作成し、そのモデルにゲーム中の盤面の状態を与えることで、AI が石を置くべきだと判断したマスの番号を出力させる。学習の繰り返し回数は 20000 回とする。

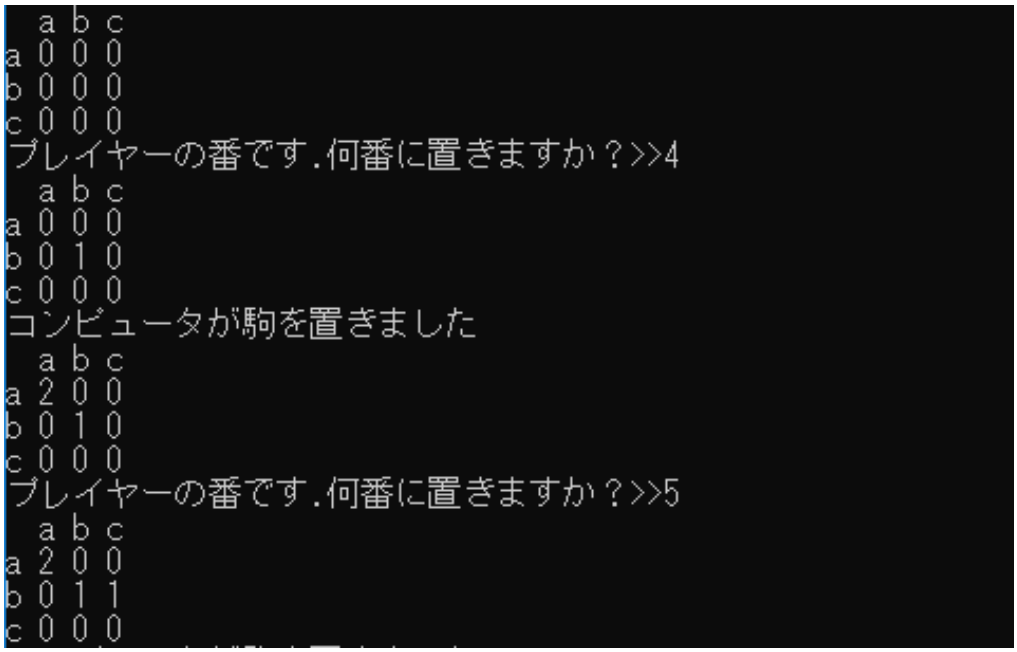

図１　三目並べプレイ画面イメージ

第2節　教師データ

盤面の状態を board 配列に格納する。配列の要素番号は、上一列を左から 0,1,2、中央一列を左から 3,4,5、下一列を左から 6,7,8 とする。プレイヤーの石が置かれている状態なら 1、AI の石が置かれている状態なら 2、石が置かれていない状態なら 0 として、対応する要素に数字を入れていく。board 配列の値が{ 0,0,1,0,2,0,0,0,0 }であった場合、盤面右上にプレイヤーの石、盤面中央に AI の石が置かれている状態ということになる。

教師データは CSV ファイル形式で与える。board 配列の値を入力データ、その場合に石を置くべき場所の要素番号を出力データとして与えてニューラルネットワークを学習させた。

教師データは 20 セット用意した。教師データセットの例を図 2 に示す。

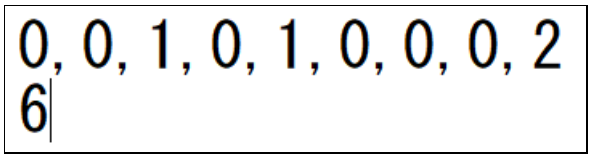

図 2　教師データセットの一例

第4章 結果

3層構造ニューラルネットワークの学習済みモデルに盤面の状態{1,2,2,0,1,0,0,1,0}を入力として与えたときの出力を図3に示す。

図3の一番下を除いた列に書いてある数字は、サンプルデータの学習結果である。左側が正しい出力として求められている値、右側が学習によって導出された値である。双方の数値が一致していることから、サンプルデータの学習は成功しているといえる。一番下にある数字が、入力に対しての出力で、図の場合は要素番号8に石を置くのが最善手とAIが判断したことになる。盤面と照合して確認すると、AIの判断は最善手であることがわかる。

しかしながら、意図しない出力もあった。特に、盤面の石が少ない状態、つまり、入力データに0が多い状態のときは期待した出力が得られないことが多かった。

そののち、4層構造ニューラルネットワークを用いて学習させた結果、学習にかかる時間が倍加したものの、学習の精度は向上した。それでも意図しない出力は無くならなかった。

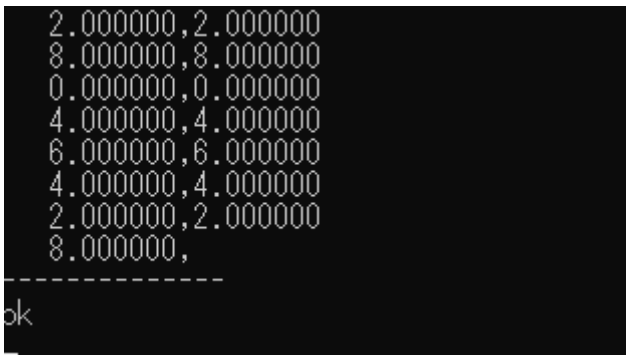

図3　　実行画面

第5章 課題と考察

実行結果より、現状では学習の精度は実用的なほど高くはないといえる。教師データとして与えた入力であれば間違えることはないが、未知の入力に対しては「引き分けにする」という目標を達成できないこともあった。

入力データに0が多い状態のときに期待した出力が得られないことが多かった理由は、情報に法則性を見出すには単純すぎる入力の情報量であったことが考えられる。入力データを盤面の状態のみにする場合は、序盤はテンプレートを作り、決まった行動をとったほうが動作は安定すると思われる。

改良の余地として、隠れ層の増加と教師データとして与える入力データの再考察が考えられる。

現在、深層学習の隠れ層は1層・2層である。これを更に増やしていくことで、計算に複雑さが生じ、法則性を見出す契機になる可能性がある。

また、入力データに9つの数列を与えるだけでは、3×3の盤面を完全に表現しているとは言えず、それがパターンの認識に支障をきたしていると推測できる。よって、入力に情報を追加したり、出力を1つとせずにいくつか候補を出させたりして、学習を効率化すべきであると思われる。

また本研究を通じて、ニューラルネットワークの構造を拡張しただけでは強化学習に取って代わることは難しいということが分かった。強化学習には、繰り返すごとに精度が増す強みがあるが、深層学習には精度の伸びが強化学習ほど見込めないため、ランダム性に強い機械学習をさせるには強化学習が最適であると言える。

逆に深層学習が得意とする分野は、自動翻訳の様に教師データを指定しやすく、ある程度数を確保しやすいアプリケーションであるということが分かった。現在、口語や慣用句を正しく翻訳することは非常に難しい。特に日本語は、同じ意味を指す別の言葉が数多く

存在する。そこで、翻訳が明らかに違った場合、手直しをしてもらうシステムを導入し、その手直しを教師データとして学習させる。同じ翻訳ミスが複数あった場合、どのように翻訳すべきかを導出し、自然な翻訳になるまで手直しを加え続けることで正しい翻訳ができるアプリケーションになるだろう。

参　考　文　献

[1] Github・tiny-dnn
https://github.com/tiny-dnn/

[2] C++ヘッダだけで Deep Learning、tiny-dnn の紹介
https://qiita.com/nyanp/items/11c6bb6fb539486c5069

付録

・深層学習プログラムソースコード

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void output(int x[9]);

int main(void)
{
	int board[9] =  { 0,0,0,
			  0,0,0,
			  0,0,0 };
	int p, c=0;

	output(board);//盤面の表示
	while (1) {
		printf("プレイヤーの番です.何番に置きますか？>>");
		scanf("%d", &p);//プレイヤーに入力を促す、番号はあとでわかりやすくする(今は添え字)

		while (board[p] != 0) {
			printf("そこには駒が置かれています.再度選んでください>>");
			scanf("%d", &p);//すでに駒が置かれていたら再度選択させる
		}
		board[p] = 1;//プレイヤーの指定した場所を1とする
```

```
                output(board);

                if (board[0] != 0 && board[1] != 0 && board[2] != 0 &&
                    board[3] != 0 && board[4] != 0 && board[5] != 0 &&
                    board[6] != 0 && board[7] != 0 && board[8] != 0)//盤面が
1か2で埋まったら終了
                {
                        printf("終了です¥n¥n");
                        system("pause");
                        return 0;
                }

                printf("コンピュータが駒を置きました¥n");


                while (board[c] != 0) {
                        /*本来はここにディープラーニングのプログラムを入れ
る
                          いまは乱数を生成して自動的においているだけ*/
                        srand((unsigned)time(NULL));
                        c = rand() % 9;// 0 ～ 8 の乱数を生成,すでに駒が置か
れているなら再抽選
                }
                board[c] = 2;//コンピュータの選んだ場所を2とする
                output(board);
        }
}

void output(int x[9]) {
        int i;
        printf("   a b c¥n");//横abcの表示
        for (i = 0; i < 3; i++) {
                printf("%c %d %d %d¥n", i+97, x[i * 3], x[i * 3 + 1], x[i * 3 +
2]);//縦abc、盤面の表示
        }
}
```

・深層学習プログラムソースコード

```
#define _CRT_SECURE_NO_WARNINGS
#define _SCL_SECURE_NO_WARNINGS

#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <vector>
#include <cmath>
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>


#include "tiny_dnn/tiny_dnn.h"
using namespace tiny_dnn;
using namespace tiny_dnn::layers;
using namespace tiny_dnn::activation;


char* readTextFile(const char* filename)
{
	FILE *fp = NULL;
	char buf[256];
	char* str = NULL;
	int fsize;

	/* ファイルのオープン */
	fp = fopen(filename, "r");
	if (fp == NULL) {
		printf("Can't read '%s'.¥n", filename);
		exit(1);
```

```
        };

        /* ファイルサイズの取得 */
        fseek(fp, 0L, SEEK_END);
        fsize = ftell(fp);

        /* メモリの動的確保 */
        str = (char*)malloc(fsize * sizeof(char));
        if (str == NULL) {
                printf("Can't allocate memory. 'str' is NULL.¥n");
                fclose(fp);
                exit(1);
        }

        /* シーク位置を先頭に戻す */
        fseek(fp, 0L, SEEK_SET);

        /* テキストデータの読み込み */
        str[0] = '¥0';
        while (fgets(buf, sizeof(buf), fp) != NULL) {
                strncat(str, buf, strlen(buf) + 1);
        }

        /* ファイルのクローズ */
        fclose(fp);

        return str;
}

void readCSVData(
        double** pArray,        /* CSVデータ格納用配列 */
        int* pElems,            /* CSVデータの要素数 */
        int* pRows,             /* CSVデータの行数 */
        int* pCols,             /* CSVデータの列数 */
        const char* filename    /* csvファイルの名前 */
) {
```

```
char *str = NULL;
char elem[15];        /* 要素を文字列として格納 */
char *ep = NULL;
int countNL = 0;      /* 改行の数をカウント */
int countSep = 0;     /* カンマの数をカウント */
int cols = 0;         /* 各行の列数をカウント */
int i, j, k;
double val;

/* テキストファイルの読み込み */
str = readTextFile(filename);

/* 要素数，行数，列数の取得 */
for (i = 0; i < strlen(str); i++) {
        switch (str[i]) {
        case ',': countSep++; cols++; break;
        case '¥n':
                countNL++; cols++;
                *pCols = (*pCols > cols) ? *pCols : cols;
                cols = 0;
                break;
        }
}
*pRows = countNL;
*pElems = countSep + countNL;

/* CSVデータ格納用配列の生成 */
*pArray = (double *)calloc(
        (*pRows) * (*pCols) * sizeof(double), sizeof(double));
if (*pArray == NULL) {
        printf("can't allocate memory. '*pArray' is NULL. ¥n");
        free(str); str = NULL;
        exit(1);
}

/* CSVデータを格納用配列へコピー */
```

```
        i = j = 0;
        k = 0; elem[0] = '0'; elem[1] = '¥0';
        while (j < (*pRows) * (*pCols)) {
                if (i < strlen(str)) {
                        switch (str[i]) {
                        case ',': case '¥n':
                                val = strtod(elem, &ep);
                                if (*ep != '¥0') {
                                        printf("Warning (%d, %d): "
                                                "Conversion may be incorrect.
¥n",
                                                j / (*pCols), j % (*pCols));
                                }
                                (*pArray)[j] = val; j++;
                                if (str[i] == '¥n') {
                                        while (j % (*pCols) > 0) {
                                                (*pArray)[j] = 0.0; j++;
                                        }
                                }
                                k = 0; elem[0] = '0'; elem[1] = '¥0';
                                break;
                        default:
                                if (k + 1 < sizeof(elem)) {
                                        elem[k] = str[i]; elem[k + 1] = '¥0';
k++;
                                }
                                else if (k + 1 == sizeof(elem)) {
                                        printf("Warning (%d, %d): "
                                                "Too many digits. ¥n",
                                                j / (*pCols), j % (*pCols));
                                        k++;
                                }
                                break;
                        }
                        i++;
                }
```

```
                else {
                        (*pArray)[j] = 0.0; j++;
                }
        }

        /* メモリを動的に確保するので解放が必要 */
        free(str); str = NULL;

        return;
}


int main(int argc, char* argv[]) {
        const int num = 500;
        //      double x_train[2];
        //      double y_train[60];
        double mirikun[30];
        const char *filename = "teacher.csv";           //教師データのCSVフ
ァイル 1行目入力(盤面の状況)、２行目数値(AIの次の入力先)
        double *data = NULL;
        int elems = 0;  /* 全要素数 */
        int rows = 0;   /* 行の数 */
        int cols = 0;   /* 列の数 */
        std::vector<tiny_dnn::vec_t> input;
        std::vector<tiny_dnn::vec_t> output;

        /* 引数からcsvファイルの名前を取得 */
        if (argc >= 2) {
                filename = argv[1];
        }

        /* CSVデータの読込 */
        readCSVData(&data, &elems, &rows, &cols, filename);


        printf("¥n");
```

```
        printf("filename = %s¥n¥n", filename);
        printf("要素数  = %d¥n", elems);
        printf("行数  = %d¥n", rows);
        printf("列数  = %d¥n", cols);

        system("pause");
        //教師データの確認のためポーズ画面表示

        /* data[i*cols+j] = i行j列の成分  */
        for (int i = 0; i < rows; i += 2) {

                printf("¥nNo.%d¥n", i / 2);

                //  入力
                tiny_dnn::vec_t vx;
                printf("盤面の状態  =");
                for (int j = 0; j < 9; j++) {                   //        j=入力
の値の数 0 ～ 8 の 9 つ
                        const float x = data[i*cols + j];
                        printf("%.3f, ", x);
                        vx.push_back(x);
                }
                input.push_back(vx);
                printf("¥n");

                //  出力
                tiny_dnn::vec_t vy;
                for (int n = 0; n < 1; n++) {                   //    n=出力の個
数=1固定
                        const float y = data[(i + 1)*cols + n];
                        printf("%.3f,", y);

                        vy.push_back(y / 100);
                }
                output.push_back(vy);
                printf("¥n");
```

```
	}


	int i = 0;


	tiny_dnn::network<tiny_dnn::sequential> net;
	//		network<sequential> net;
	net << fc(9, 100)
	//fc(入力数、中間層数)
		<< tiny_dnn::tanh_layer();
	net << fc(100, 100)
		<< tiny_dnn::sigmoid_layer();
	net << fc(100, 1)
	//fc(出力数、中間層数)
		<< tiny_dnn::sigmoid_layer();


	printf("¥n--------------¥n");

	auto loss = net.get_loss<cross_entropy>(input, output);
	printf("loss -> %lf¥n", loss);

	adagrad opt;

	//net.fit<mse>(opt, input, output, 5, 30);
	net.fit<mse>(opt, input, output, 20, 20000);
		//(、、、サンプルの数・教師データの数、エポック数・学習回数)(変更有)

	loss = net.get_loss<cross_entropy>(input, output);
	printf("loss -> %lf¥n", loss);

	printf("¥n");

	//float bpm = 165.0;
```

```
	for (int k = 0; k < rows/2; k++) {
		vec_t a = net.predict(input[k]);
		printf("	%lf,%lf¥n", round( output[k][0] * 100 ), round( a[0] * 100));
	}

	tiny_dnn::vec_t input2 = { 2,0,1,1,1,2,0,0,0 };
	//実際に学習する値・盤面の状況を入力→どこに置けばいいのかの出力が得られます(変更有)
	vec_t a2 = net.predict(input2);
	printf("	%lf,", round(a2[0] * 100));

	printf("¥n----------¥n");

	net.save("test.csv");

	//net.load("test.csv");

	printf("ok¥n");
	getchar();


	return 0;
}
```

・教師データ

1,0,0,1,2,2,0,0,1
6
0,0,1,0,1,0,0,2,0
6
0,1,0,0,1,2,0,0,0
7
1,0,0,0,2,0,2,1,1
2
2,0,0,1,2,0,1,0,1
7
1,0,1,0,2,1,0,0,2
1
2,0,0,1,1,0,2,0,1
5
1,0,2,0,1,0,0,0,0
6
0,1,1,0,0,0,0,2,0
0
2,0,2,1,1,2,1,0,1
1
0,0,0,2,1,1,1,2,0
2
1,0,2,0,1,1,0,0,2
3
0,2,0,0,1,1,2,1,0
3
1,1,0,2,2,1,1,2,0
2
0,0,0,0,1,0,0,0,0
8
0,2,2,1,1,0,1,1,2
0

0,0,1,0,0,0,0,0,0
4
0,0,0,1,0,0,0,0,0
6
0,1,0,0,0,0,0,0,0
4
0,0,0,2,1,1,1,0,2
2