平成３０年度　卒業論文

AI を用いた音楽生成の研究

函館工業高等専門学校　生産システム工学科　情報コース

１２番　佐藤　靖高

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
　　第２節 研究背景
　　第３節 研究目的
　　第４節 開発環境

第２章 関連技術
　　第１節 Tiny-DNN
　　第２節 ニューラルネットワーク

第３章 音楽生成プログラムの開発
　　第１節 プログラムの概要
　　第２節 教師データ
　　第３節 出力データ

第４章 音楽再生プログラムの開発
　　第１節 プログラムの概要
　　第２節 midiOut 関数

第５章 結果

第６章 課題

参考文献

付録
　　ソースコード
　　教師データ
　　出力データ

第１章　序論

第１節　　英文アブストラクト

It is a purpose to generate music using deep learning which is one of AI techniques.

In this research, development was done in c++ environment using Visual Studio. Tiny-DNN was used as the AI library. Create training data in the CSV file, load it and do learning. Similarly, output can be done with a CSV file, and the generated music can be played by loading the outputted CSV file.

In conclusion, although I was able to produce music, it is not accurate and I have to think about improving accuracy. Lack of training data and input perceptron is considered as a cause.

Key words :　　AI, deep learning, Tiny-DNN

第２節　　研究背景

近年、人口知能（AI）による技術が発展しており注目を集めている。

我々は、このAIの技術の１つであるディープラーニングの技術に注目した。

ディープラーニングの技術は以後や将棋、ポーカーなどに使われプロのプレイヤーをも破り最先端技術となっている。

そこで、この技術を用いることで誰でも簡単に音楽の生成をすることは出来ないだろうかと考えた。

第３節　　研究目的

本研究では、音楽知識に疎く作曲をすることのできない人の代わりにAIを用いて音楽生成することを研究目的とする。

第４節　　開発環境

・使用 PC

OS：　windows10 Home 64bit

CPU:　Intel® Core™2 Duo E7200 @2.53GHz

RAM:　4.00GB

・開発環境

Visual Studio 2017

・開発言語

C++

・使用ライブラリ

Tiny-DNN

第２章　関連技術

第１節　　Tiny-DNN

「Tiny-DNN」とはAIライブラリの１つである。これ以外のAIライブラリとしては、「Tensorflow」、「Keras」などが存在する。AIライブラリは主にPythonに対応しているものが多い。

「Tiny-DNN」は、C++環境においてディープラーニングの実装を可能にするライブラリで、ヘッダーファイルのみの構成となっており、C++をサポートしているコンパイラが存在する開発環境ならばどの環境においても用いることができる他環境に対応しているライブラリである。

特徴としてこのライブラリではGPUを使用せず、CPUのみで動作するため、GPUを新たに増設する必要性がないことがあげられる。

本研究ではC++を開発言語とするため、この「Tiny-DNN」を使用することとした[1]。

第２節　　ニューラルネットワーク

ニューラルネットワークとは人間の脳神経系のニューロンを数理モデル化したパーセプトロンの組み合わせである。

本研究では、入力層を２、中間層を１００、出力層を１８０とした３層パーセプトロンネットワークを使用する(図 1)。

入力値には BPM（Beats Per Minute）と作曲者識別番号の２つを、出力値は、ノート番号と音符データの組み合わせ（２×９０組＝１８０出力）とした。

ノート番号と音符データについては３章１節の教師データの項にて説明する。

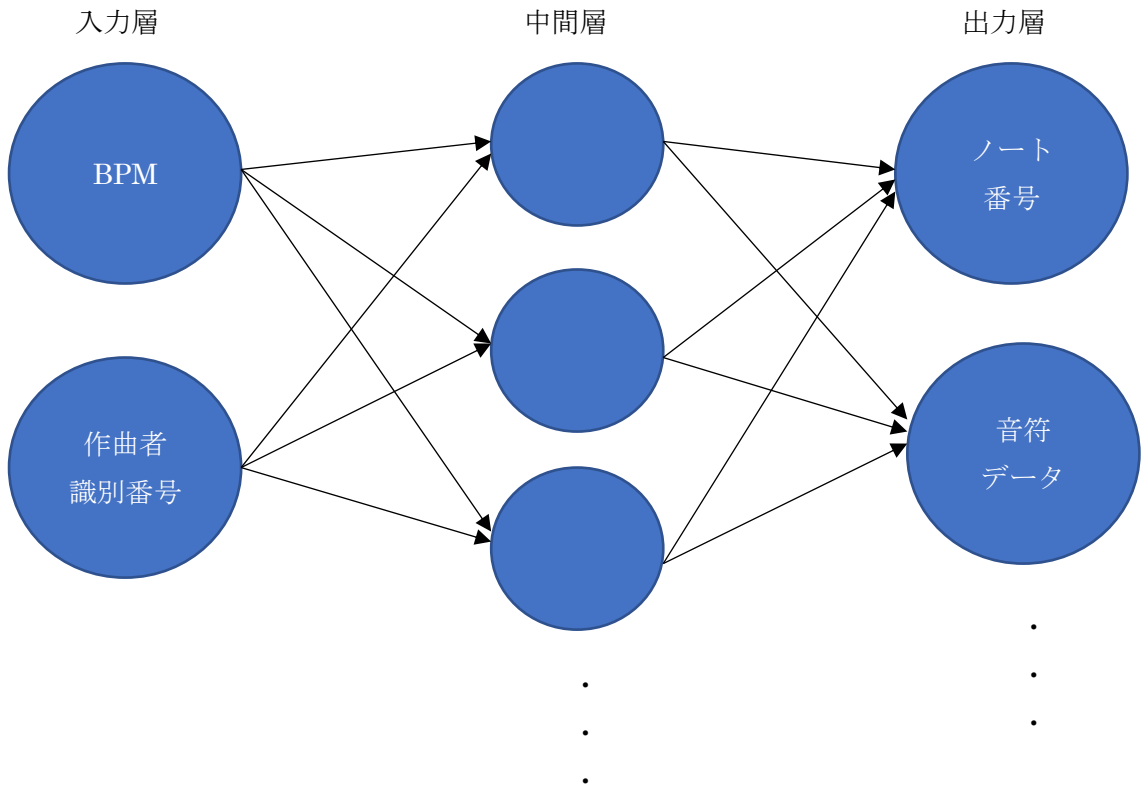

図１　ニューラルネットワークのモデル図

第３章 音楽生成プログラムの開発

第１節　　教師データ

プログラムの概要に先立ち第２章第２節でも触れたノート番号と音符データについて説明する。

まず、ノート番号とは正確にはMIDIノート番号（ノートナンバー）と言われるもので楽譜にある音符を元にして対応しているものをノート番号とした[2]。

次に音符データとは、先ほどのノート番号を割り当てた際の音符について２分音符や４分音符といった音符の違いを数字に置き換えて考えたもののことを音符データとした(表１)。

１つの音符からそれぞれ１つのノート番号と音符データを作ることができるのでこの２つを１つのセットと考え９０セット用意したものを教師データとして扱う。この教師データはCSVファイルとして作成する(図2)。

音符	音符データ
２分音符	２
付点２分音符	３
４分音符	４
付点４分音符	５
８分音符	６
付点８分音符	７
１６分音符	８

表１　音符と音符データの割り当て

```
110,0
70,6,72,6,70,6,72,6,70,6,72,6,70,6,68,6,
72,8,70,6,68,8,68,8,70,6,72,6,65,6,67,6,
70,6,69,6,70,6,69,6,65,6,65,6,70,6,72,6,
68,6,75,6,72,8,72,8,75,6,72,8,72,8,70,8,
65,6,67,6,68,6,70,6,68,6,70,6,72,6,70,6,
```

図 2　教師データ（一部抜粋）

第２節　　プログラムの概要

本プログラムではAIライブラリとして「Tiny-DNN」を使用し、ニューラルネットワークを構築し、教師データとして楽曲をいくつか学習させて楽曲を作成することを目的としている。

教師データのCSVファイルを読み込み、教師データから入力値に当たる、BPM(Beats Per Minute）と作曲者識別番号の２つと、出力値に当たる、ノート番号と音符データの組み合わせを識別し、入力と出力に当たる部分を合わせたものを学習させる。学習にはTiny-DNNを使用したニューラルネットワークを用いる。ここではあらかじめ決めていた入力層数、中間層数、出力層数に加えてエポック数と呼ばれる繰り返して学習する回数を決める必要がある。エポック数はプログラムを実行するPCによって最適な数が変化する。今回の研究では1000エポックが最適と判断した。この際、学習内容を任意のファイルに保存することで同じ学習データを活用できるようにした。

BPMと作曲者識別番号の２つを任意の値で与えることで先ほど学習した内容をもとに出力を得ることが出来る。この得られた出力データを教師データと同じくCSVファイルに保存する。この時ノート番号と音符データの組み合わせの他に音楽再生で必要になってくるBPMの値も保存する。

第３節　　出力データ

教師データと同じく、CSV ファイルで作成する。

出力データには出力する際に入力した BPM の値とノート番号と音符データの組み合わせが保存されている(図 3)。

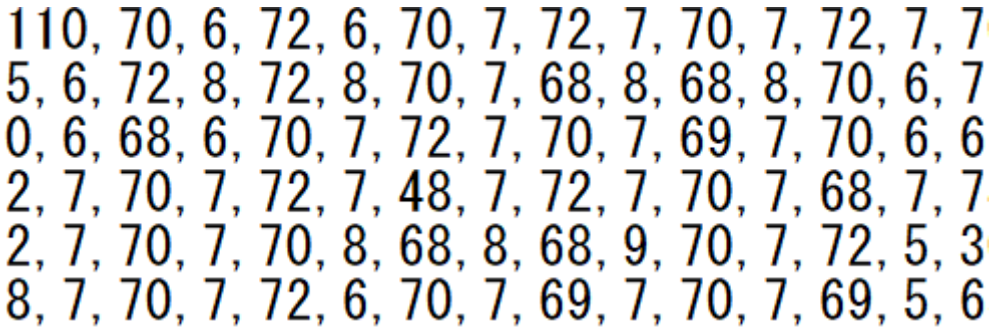

図 3　出力データ（一部抜粋）

第４章 音楽再生プログラムの開発

第１節　　プログラムの概要

音楽生成プログラムで作成された出力データの CSV ファイルを読み込み BPM とノート番号と音符データの組み合わせを所得する。所得した BPM に基づいて音符データをミリ秒に変換する。この処理の計算は音符によって式が決まっているためそちらを活用する。音符データをミリ秒に変換したら後述する midiOut 関数を使用しノート番号応じた鍵盤を押し、求めたミリ秒だけ音を鳴らし、終わったら鍵盤を離すことで１音を鳴らすことが出来る。これを繰り返し１つのメロディーとして音を流す。

第２節　　midiOut 関数

Windows 標準 API の１つで指定した音の鍵盤を押す、離すという処理ができる。押す際の処理と離す際の処理が必要で、押す処理と離す処理の間に Sleep 関数を挟むことで指定したミリ秒だけ鍵盤を押し続け、指定した時間が経過したら鍵盤を離すという処理を行うことが出来る(図 4)

この時値として与えてる部分の 1 byte 目の 0x90 は、鍵盤を押す又は離すことを表している。２byte 目の 0x30 は音階とオクターブを表しており、音階が１つ上がるごとに＋１、オクターブが１つ上がるごとに＋１２されていく。ここでは音階 C のオクターブ４、キーボードでいう真ん中のドの音を表している。３byte 目の 0x7f はベロシティと呼ばれる鍵盤を押す強さを表している [3] 。

```
midiOutShortMsg( h , 0x7f3090 );
sleep(1000);
midiOutShortMsg( h , 0x7f3090 );
```

図 4　midiOut 関数の使用例

第５章 結果

初めにAIによる学習を確認する目的で、教師データを１つ、入力値を教師データと同じものを使用して実行した。出力されたデータは教師データで与えたものと僅かな違いはあったがほぼ同様のデータを得ることが出来た。音として再生して比べてもほとんど遜色ないものだった。このことからAIによる学習は正常に行われている。

続いて教師データとして作曲者識別番号が同じでありBPMが離れている２つを与え、入力値をどちらか一方のデータに寄せて実行した。出力されたデータは教師データ１つの場合と同じくほぼ同様のデータを得ることが出来た。

最後に作曲者識別番号事に４つずつ２種類を合わせて８つの教師データを学習した。この時入力値にはBPMを60,120,180の３種類、作曲者識別番号には0,1,0.5の３種類の計９パターンで実行した。作曲者識別番号０．５に関しては作曲者識別番号０と１の両方の特徴を併せ持つデータを作成できないか確かめる目的で設定した。BPMが６０で作曲者識別番号が０，１の場合は。それぞれの作曲者識別番号の教師データの内容を内包したものとなった。BPMが１２０、180の場合は作曲者識別番号に応じてBPMが一番近い教師データを元にしたと思われるデータとなった。作曲者識別が0.5の場合はBPMの値にかかわらず全ての教師データとは関連性を感じることの出来ないデータが出力された(表2,3)。

ノート番号＼BPM	110	145	200	210
0	68～75	62～74	81～85	69～76
ノート番号＼BPM	138	170	183	192
1	76～81	58～72	72～79	74～86

表２　教師データのノート番号

ノート番号＼BPM	60	120	180
0	68～92	65～74	70～76
1	68～98	73～87	74～85
0.5	83～96	68～74	71～75

表３　出力データのノート番号

第６章 課題

本研究の今後の課題としては、より良い精度でのデータ生成が挙げられる。

精度が悪い原因としては、教師データ及び入力パーセプトロンの不足と考えられる。AI による作曲技術の知識を調べている際に作曲には膨大な教師データが必要とされており、本研究で用意できた８つという教師データはとても少なく、どの程度の教師データが必要かは明確ではないが、データ数が不足していたことは明白である。また、入力パーセプトロンについても現状では BPM と作曲者識別番号の２つだが、この２つの入力で識別できる範囲は小さく、限界があると思われるためいくつか別の要素を与える必要があると思われる。

参考文献

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

［2］ http://www.asahi-net.or.jp/~HB9T-KTD/music/Japan/Research/DTM/freq_map.html

［3］ http://yamatyuu.net/computer/program/vc2013/midi/index.html?note=C%2B%2CD-&oct=5&v=120&ch=1

付録

・音楽生成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 <random>

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

/*CSV ファイル読み込み*/
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 mirikun[30];
    const char *filename = "data5.csv";      //教師データの CSV ファイルの
名前
    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);

    /*読み込んだ CSV ファイルの名前、要素数、行、列の表示*/
/*  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);*/

    /* 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;
```

```
        for (int j = 0; j < 2; j++) {           //  j=入力のとして扱う値の個数。今回は BPM、番号の2つなので2
            const float x = data[i*cols + j];
            if (j != 1) {
            printf("BPM = %.0f, ", x);
            }
            else {
                printf("長短 = %.0f", x);
            }
            vx.push_back(x);
        }
        input.push_back(vx);
        printf("¥n");

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

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

    int i = 0;

    /*ニューラルネットワークの作成*/
    tiny_dnn::network<tiny_dnn::sequential> net;

    /*ここから*/

    net << fc(2, 100)                   //fc(入力数、中間層数)
        << tiny_dnn::tanh_layer();
    net << fc(100, 188)                 //fc(中間層数、出力数)
188:data5//182:data9
        << tiny_dnn::sigmoid_layer();
```

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

    adagrad opt;

    net.fit<mse>(opt, input, output, 4, 1000);  //(、、、サンプルの数・教師データの数、エポック数・学習回数)

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

        printf("¥n¥n");

    /*ここまで同じ教師データを使用する場合は 2 度目以降の実行時コメントアウトすると実行が早くなる*/

    //net.load("teacher");  //上記に従い上部をコメントアウトした際この行のコメントアウトを外す

    float bpm = 0.0;
    float no = 0.0;
    /*入力を与える。この値によって出力が変化する*/
    printf("入力 1>>");
    scanf("%f" ,&bpm);
    printf("入力 2>>");
    scanf("%f" ,&no);
    printf("¥n 入力値 : %.0f , %.0f¥n",bpm,no);

    tiny_dnn::vec_t input2 = { bpm, no };
    vec_t a = net.predict(input2);

    using namespace std;
    ofstream log;
    log.open("log.csv", ios::trunc);  // ofstream log("log.csv")でも可。第 2 引数の trunc は同名ファイルがあった場合には以前の内容を全て消す、というオプション。
    /*CSV ファイルに保存完了*/
```

```
    log << bpm;
    log << ",";
    for (int k = 0; k < cols; k++) {
        printf("%.0f,", round(a[k] * 100));  //a[k] k=0~cols 変換後の
値 *100 で普通の値になる。round()で囲うと整数にできる。
        log << round(a[k] * 100);
        if(k < cols - 1) {
            log << ",";
        }
    }
    log << "¥n";
    log.close();

    net.save("teacher");

    printf("¥n");

    return 0;
}
```

・音楽再生プログラムソースコード

```
/*音楽生成プログラム*/

#define _CRT_SECURE_NO_WARNINGS
#define _SCL_SECURE_NO_WARNINGS

#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <vector>
#include <windows.h>
#include <mmsystem.h>
#include <tchar.h>

#pragma comment(lib,"winmm.lib")

int save=0;
int v=0;
int b = 0;
int pop = 0;
int bpm = 0;

/*CSV ファイルの読み込み*/
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;
```

```
}

/*鍵盤処理の関数*/
int midi(int* v) {
    switch(save) {
            case 84:
                *v = 0x7f5490;  // 鍵盤を押す C7 0x54(84) 127 チェンネル 0
                break;
            case 83:
                *v = 0x7f5390;  // 鍵盤を押す B6 0x53(83) 127 チェンネル 0
                break;
            case 82:
                *v = 0x7f5290;  // 鍵盤を押す A+,B-6 0x52(82) 127 チェンネル 0
                break;
            case 81:
                *v = 0x7f5190;  // 鍵盤を押す A6 0x51(81) 127 チェンネル 0
                break;
            case 80:
                *v = 0x7f5090;  // 鍵盤を押す G+,A-6 0x50(80) 127 チェンネル 0
                break;
            case 79:
                *v = 0x7f4f90;  // 鍵盤を押す G6 0x4f(79) 127 チェンネル 0
                break;
            case 78:
                *v = 0x7f4e90;  // 鍵盤を押す F+,G-6 0x4e(78) 127 チェンネル 0
                break;
            case 77:
                *v = 0x7f4d90;  // 鍵盤を押す F6 0x4d(77) 127 チェンネル 0
                break;
            case 76:
                *v = 0x7f4c90;        // 鍵盤を押す E6 0x4c(76) 127 チェンネル 0//
                break;
            case 75:
                *v = 0x7f4b90;  // 鍵盤を押す D+,E-6 0x4b(75) 127 チェンネル 0
                break;
            case 74:
                *v = 0x7f4a90;  // 鍵盤を押す D6 0x4a(74) 127 チェンネル 0
                break;
```

```
case 73:
    *v = 0x7f4990;  // 鍵盤を押す C+,D-6 0x49(73) 127 チェンネル
    break;
case 72:
    *v = 0x7f4890;  // 鍵盤を押す C6 0x48(72) 127 チェンネル 0
    break;
case 71:
    *v = 0x7f4790;  // 鍵盤を押す B5 0x47(71) 127 チェンネル 0
    break;
case 70:
    *v = 0x7f4690;  // 鍵盤を押す A+,B-5 0x46(70) 127 チェンネル 0
    break;
case 69:
    *v = 0x7f4590;  // 鍵盤を押す A5 0x45(69) 127 チェンネル 0
    break;
case 68:
    *v = 0x7f4490;  // 鍵盤を押す G+,A-5 0x44(68) 127 チェンネル 0
    break;
case 67:
    *v = 0x7f4390;  // 鍵盤を押す G5 0x43(67) 127 チェンネル 0
    break;
case 66:
    *v = 0x7f4290;  // 鍵盤を押す F+,G-5 0x42(66) 127 チェンネル 0
    break;
case 65:
    *v = 0x7f4190;  // 鍵盤を押す F5 0x41(65) 127 チェンネル 0
    break;
case 64:
    *v = 0x7f4090;  // 鍵盤を押す E5 0x40(64) 127 チェンネル 0
    break;
case 63:
    *v = 0x7f3f90;  // 鍵盤を押す D+,E-5 0x3f(63) 127 チェンネル 0
    break;
case 62:
    *v = 0x7f3e90;  // 鍵盤を押す D5 0x3e(62) 127 チェンネル 0
    break;
case 61:
    *v = 0x7f3d90;  // 鍵盤を押す C+,D-5 0x3d(61) 127 チェンネル 0
```

```
    break;
case 60:
    *v = 0x7f3c90;  // 鍵盤を押す C5 0x3c(60) 127 チェンネル 0
    break;
case 59:
    *v = 0x7f3b90;  // 鍵盤を押す B4 0x3b(59) 127 チェンネル 0
    break;
case 58:
    *v = 0x7f3a90;  // 鍵盤を押す A+,B-4 0x3a(58) 127 チェンネル 0
    break;
case 57:
    *v = 0x7f3990;  // 鍵盤を押す A4 0x39(57) 127 チェンネル 0
    break;
case 56:
    *v = 0x7f3890;  // 鍵盤を押す G+,A-4 0x38(56) 127 チェンネル
    break;
case 55:
    *v = 0x7f3790;  // 鍵盤を押す G4 0x37(55) 127 チェンネル 0
    break;
case 54:
    *v = 0x7f3690;  // 鍵盤を押す F+,G-4 0x36(54) 127 チェンネル 0
    break;
case 53:
    *v = 0x7f3590;  // 鍵盤を押す F4 0x35(53) 127 チェンネル 0
    break;
case 52:
    *v = 0x7f3490;  // 鍵盤を押す E4 0x34(52) 127 チェンネル 0
    break;
case 51:
    *v = 0x7f3390;  // 鍵盤を押す D+,E-4 0x33(51) 127 チェンネル 0
    break;
case 50:
    *v = 0x7f3290;  // 鍵盤を押す D4 0x32(50) 127 チェンネル 0
    break;
case 49:
    *v = 0x7f3190;  // 鍵盤を押す C+,D-4 0x31(49) 127 チェンネル 0
    break;
case 48:
```

```
        *v = 0x7f3090;  // 鍵盤を押す C4 0x30(48) 127 チェンネル 0
        break;
    case 85:
        *v = 0x7f5590;
        break;
    case 86:
        *v = 0x7f5690;
        break;
    case 87:
        *v = 0x7f5790;
        break;
    case 88:
        *v = 0x7f5890;
        break;
    case 89:
        *v = 0x7f5990;
        break;
    case 90:
        *v = 0x7f5a90;
        break;
    case 91:
        *v = 0x7f5b90;
        break;
    case 92:
        *v = 0x7f5c90;
        break;
    case 93:
        *v = 0x7f5d90;
        break;
    case 94:
        *v = 0x7f5e90;
        break;
    case 95:
        *v = 0x7f5f90;
        break;
    default:
        *v = 0;
        break;
```

```
    }
    return *v;
}

/*音符番号からミリ秒への変換関数*/
int msc(int* b) {
    switch (pop) {
        case 1:
            *b = (60000 / bpm) * 4;
            break;
        case 2:
            *b = (60000 / bpm) * 2;
            break;
        case 3:
            *b = (60000 / bpm) * 2.5;
            break;
        case 4:
            *b = (60000 / bpm) ;
            break;
        case 5:
            *b = (60000 / bpm) * 1.5;
            break;
        case 6:
            *b = (60000 / bpm) * 0.5;
            break;
        case 7:
            *b = (60000 / bpm) * 0.75;
            break;
        case 8:
            *b = (60000 / bpm) * 0.25;
            break;
        case 9:
            *b = (60000 / bpm) * 0.25;
            break;
        default:
            *b = 0;
            break;
    }
```

```
    return *b;

}

void _tmain(int argc, char* argv[]) {
    HMIDIOUT h;
    const char *filename =
"C:/Users/satoy/Source/Repos/Project6/Project6/log.csv"; //生成プログラムでつくった結果の CSV ファイル
    //const char *filename = "出力データ.csv";   //ファイルを直下に持って来た時用
    double *data = NULL;
    double y[256] = { 0.0 };
    int elems = 0;  /* 全要素数 */
    int rows = 0;   /* 行の数 */
    int cols = 0;   /* 列の数 */


    /* 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);

    //const double y[60] = { 0.0 };

    for (int i = 0; i < rows; i += 2) {

        //printf("¥nNo.%d¥n", i / 2);

        for (int n = 0; n < cols; n++) {
```

```
            y[n] = data[n];
            printf("%.0f,", y[n]);
        }
        printf("¥n");
    }

    midiOutOpen(&h, MIDI_MAPPER, 0, 0, 0);

#if 1
    midiOutShortMsg(h, 0x0);    // 音色を定義 アコースティックピアノ 0x0(0)cc0

    bpm = round(y[0]);

        int count = 0;
        for (int i = 1; i < 183; i++) {//189:data5,183:data9

            if (y[i] == 0||y[i]==1) {
                pop = y[i+1];
                msc(&pop);
                Sleep(pop);
            }
            else if (y[i] > 20) {
                save = y[i];
                midi(&save);
                midiOutShortMsg(h, save);
                count++;
                i++;
                if (y[i] > 20) {
                    save = y[i];
                    midi(&save);
                    midiOutShortMsg(h, save);
                    count++;
                    i++;
                }
                if (y[i] > 20) {
                    save = y[i];
                    midi(&save);
```

```
                midiOutShortMsg(h, save);
                count++;
                i++;
            }
            pop = y[i];
            msc(&pop);
            Sleep(pop);
            for (count; count > 0; count--) {
                save = y[i] - count;
                midi(&save);
                midiOutShortMsg(h, save);
            }
        }

    }

    Sleep(1000);    //音がぶつっときれないようにするための処理

#endif
}
```

・教師データ

200	0						
85	4	85	4	85	4	85	6
83	4	85	6	83	6	81	6
85	6	83	6	83	4	81	6
83	4	85	6	83	4	85	6
83	6	81	6	81	6	85	6
85	4	85	4	85	4	85	6
83	4	85	6	83	6	81	6
85	6	83	6	83	4	81	6
83	4	85	6	83	4	85	6
83	6	81	6	81	8	85	4
85	4	85	4	85	6	83	4
85	6	83	6	81	6	85	6
83	6	83	4	81	6	83	4
85	6	83	4	85	6	83	6
81	6	81	6	85	6	85	4
85	4	85	4	85	6	83	4
85	6	83	6	81	6	85	6
83	6	83	4	81	6	83	4
85	6	83	4	85	6	83	6
81	6	81	8	85	4	85	4
85	4	85	4	85	4	85	4
85	6	83	4	85	6	83	6
81	6	85	6				
210	0						
73	4	73	4	73	6	71	6
69	6	69	6	76	4	71	6
73	5	73	4	71	6	69	6
71	6	69	6	71	6	71	6
73	6	73	6	71	4	69	6
69	4	71	4	73	4	73	4
73	6	71	6	69	6	69	6
76	4	71	6	73	5	73	4

71	6	69	6	71	6	69	6
71	6	73	6	73	6	71	6
71	6	69	6	69	6	71	4
73	4	73	4	73	6	71	6
69	6	69	6	76	4	71	6
73	5	73	4	71	6	69	6
71	6	69	6	71	6	71	6
73	6	73	6	71	4	69	6
69	4	71	4	73	4	73	4
73	6	71	6	69	6	69	6
76	4	71	6	73	5	73	4
71	6	69	6	71	6	69	6
71	6	73	6	73	6	71	6
71	6	69	6	69	6	73	4
73	4	73	6				
110	0						
70	6	72	6	70	6	72	6
70	6	72	6	70	6	68	6
75	6	72	8	72	8	75	6
72	8	72	8	70	6	68	8
68	8	70	6	72	6	65	6
67	6	68	6	70	6	68	6
70	6	72	6	70	6	69	6
70	6	69	6	65	6	65	6
70	6	72	6	70	6	72	6
70	6	72	6	70	6	68	6
75	6	72	8	72	8	75	6
72	8	72	8	70	8	70	8
68	8	68	8	70	6	72	6
65	6	67	6	68	6	70	6
68	6	70	6	72	6	70	6
69	6	70	6	69	6	65	8
70	6	72	6	70	6	72	6
70	6	72	6	72	6	70	6
72	6	70	6	72	6	70	6
68	6	75	6	72	8	72	8

75	6	72	8	72	8	70	6
68	8	68	8	70	6	72	6
65	6	67	6				
145	0						
62	6	62	6	65	6	67	4
70	6	69	4	67	6	67	6
65	6	65	6	62	6	65	6
67	4	67	6	69	6	70	6
70	6	74	4	72	4	70	6
69	6	62	6	62	6	65	6
67	4	70	6	69	4	67	6
67	6	65	6	65	6	62	6
65	6	67	4	67	6	69	6
70	6	70	6	74	4	72	4
70	6	69	6	62	6	62	6
65	6	67	4	70	6	69	4
67	6	67	6	65	6	65	6
62	6	65	6	67	4	67	6
69	6	70	6	70	6	74	4
72	4	70	6	69	6	62	6
62	6	65	6	67	4	70	6
69	4	67	6	67	6	65	6
65	6	62	6	65	6	67	4
67	6	69	6	70	6	70	6
74	4	72	4	70	6	69	8
62	6	62	6	65	6	67	4
70	6	69	4				
138	1						
77	6	79	6	81	6	86	6
84	4	81	6	79	6	77	6
74	6	77	6	82	6	81	4
79	4	77	6	79	6	81	6
77	6	84	6	79	6	77	6
79	6	81	6	79	6	77	6
77	4	76	8	77	6	79	6
81	6	86	6	84	4	81	6

79	6	77	6	79	6	81	6
82	6	81	6	79	6	77	4
77	6	79	6	81	6	84	4
81	6	79	6	77	6	77	6
77	6	81	6	79	5	77	8
77	6	79	6	81	6	86	6
84	4	81	6	79	6	77	6
74	6	77	6	82	6	77	6
79	6	81	6	86	6	84	4
81	6	79	6	77	6	74	6
77	6	82	6	81	4	79	4
77	6	79	6	81	6	77	6
84	6	79	6	77	6	79	6
81	6	79	6	77	6	77	4
76	8	77	6				
192	1						
74	4	78	4	86	2	85	2
83	4	81	3	79	4	81	4
81	3	74	4	76	4	78	2
78	2	76	4	79	3	78	4
76	4	76	4	74	4	74	4
78	4	86	2	85	4	83	4
83	4	81	3	79	4	81	2
81	4	85	4	86	4	78	4
78	4	79	4	81	5	79	8
79	4	78	6	76	6	74	4
74	4	76	4	78	2	81	6
78	6	78	4	78	6	78	6
79	6	81	4	74	4	74	4
79	6	86	6	86	8	85	4
86	6	85	4	86	8	74	4
78	4	86	4	74	4	78	4
86	2	85	2	83	4	81	3
79	4	81	4	81	3	74	4
76	4	78	2	78	2	76	4
79	3	78	4	76	4	76	4

74	4	74	4	78	4	86	2
85	4	83	4	83	4	81	3
79	4	81	2				
170	1						
65	4	68	6	65	6	70	4
70	4	70	6	68	6	65	6
61	6	63	4	61	6	61	6
63	6	63	6	65	4	65	6
65	6	65	6	68	6	65	6
63	6	61	4	58	4	65	6
65	6	63	6	63	6	63	6
63	6	63	6	63	6	65	6
65	6	68	4	65	6	63	4
61	6	61	6	61	6	63	6
65	4	61	6	61	4	70	6
68	5	65	6	65	6	68	6
65	6	70	4	70	6	70	6
70	6	68	6	65	6	61	6
63	4	61	6	61	6	63	6
63	6	65	4	65	6	65	6
65	6	70	6	72	6	61	5
58	6	58	6	65	6	65	6
63	6	61	6	61	6	61	6
61	6	61	6	63	6	65	6
63	6	61	6	61	6	61	6
63	6	61	6	61	4	61	4
61	6	61	4				
183	1						
72	4	74	4	72	4	76	4
84	4	83	2	81	6	79	4
81	4	79	4	74	6	74	6
77	4	76	6	74	6	76	4
76	4	77	4	79	4	72	4
72	6	76	6	74	4	72	4
72	4	74	6	76	3	79	2
72	4	74	4	72	4	76	4

84	4	83	2	81	6	79	5
81	4	79	4	74	6	74	6
77	4	76	6	74	6	76	4
79	4	72	4	79	4	72	4
79	4	72	4	79	4	72	4
79	4	72	4	83	4	72	4
72	4	74	4	72	4	76	4
84	4	83	2	81	6	79	4
81	4	79	4	74	6	74	6
77	4	76	6	74	6	76	4
76	4	77	4	79	4	72	4
72	6	76	6	74	4	72	4
72	4	74	6	76	3	79	2
72	4	74	4	72	4	76	4
84	4	83	2				

・出力データ

教師データ１つ、BPM110、作曲者識別番号０

110	70	6	72	6	70	7	72
7	70	7	72	7	70	5	68
7	75	6	72	6	72	8	75
6	72	8	72	8	70	7	68
8	68	8	70	6	72	5	28
7	65	6	42	7	68	6	70
6	68	6	70	7	72	7	70
7	69	7	70	6	69	7	65
6	32	5	65	7	70	7	72
7	70	7	72	7	48	7	72
7	70	7	68	7	74	7	72
9	72	9	47	5	72	7	72
7	70	7	70	8	68	8	68
9	70	7	72	5	30	7	65
6	67	5	68	7	70	7	68
7	70	7	72	6	70	7	69
7	70	7	69	5	65	7	29

2	1	4	30	2	30	2	30
2	31	4	30	4	29	4	30
4	29	2	29	4	29	3	29
2	29	4	28	4	30	4	28
4	28	4	1	4	29	4	30
4	30	4	28	4			

教師データ２つ、BPM210、作曲者識別番号 0

210	73	5	73	4	73	6	71
6	69	5	69	6	77	5	71
6	73	5	73	6	71	7	69
6	71	6	69	7	71	5	71
6	73	6	73	6	71	5	49
5	69	4	27	6	71	4	73
4	73	4	73	6	71	5	69
5	69	6	77	4	71	5	73
5	47	5	72	6	69	6	71
6	69	6	71	6	25	5	73
5	73	6	71	5	71	6	69
6	69	6	30	5	71	5	73
5	73	5	73	6	71	7	69
6	69	5	76	5	49	6	73
5	73	5	71	5	69	6	71
6	69	6	71	6	71	5	73
6	73	6	71	5	69	5	47
2	1	3	48	2	51	2	50
2	50	3	48	2	47	2	46
3	55	2	49	2	52	2	51
2	50	2	50	3	48	2	48
3	51	3	0	3	51	3	51
3	48	3	52	3			

教師データ８つ、BPM120、作曲者識別番号 0.5

120	71	6	72	6	72	6	76
5	76	5	74	6	72	6	70
6	69	6	72	7	74	7	75

5	73	6	72	7	73	6	74
7	71	7	77	5	75	6	71
6	72	6	70	6	69	6	70
6	71	5	72	7	72	5	72
6	73	6	75	6	72	5	71
6	72	6	72	5	72	6	74
6	74	6	74	6	74	5	72
5	74	6	74	7	72	6	76
5	74	7	72	7	72	6	72
6	71	7	74	7	73	5	73
7	68	6	70	6	71	5	76
6	74	5	73	6	73	6	73
6	69	6	70	6	75	6	68
7	70	6	73	6	77	6	76
5	73	6	72	6	71	6	70
6	70	6	72	7	73	5	71
5	70	6	75	6	75	7	73
7	79	6	75	7	72	7	70
6	70	7	69	7	69	6	71
5	68	7	69	5			