プログラミング演習1 (2022年7月20日)

今日の内容

- 構造体

プログラムと復習用の動画はこちら

- プログラムはこちら.
- 復習用の動画はこちら.

構造体

例題（このような問題が解けるようになるのが目標）

深層学習で使われている人工ニューラルネットワークでは，行列A，ベクトルb，活性化関数と呼ばれる関数$\sigma : \mathbb{R} \to \mathbb{R}$を使って，入力ベクトル$x$に対して

$$f(x) = \sigma(Ax + b)$$

という計算を繰り返し行う．ただし，上の計算において，σは入力されたベクトルの各成分に同じ関数を適用する：

$$\sigma(x) = \begin{pmatrix} \sigma(x_1) \\ \sigma(x_2) \\ \vdots \\ \sigma(x_n) \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

そこで，行列とベクトルをまとめたものを簡易的なニューラルネットワークと考え（このようなまとまりを「層」と呼ぶことがある），これを一つの変数で表したい．また，そのようにして表されたものを利用して，上記の$f(x)$を計算する関数を作りたい．そのようになるように，以下の書きかけのプログラムを完成させよ．

```c
#include <stdio.h>

    struct{
  double A[ ][ ]; // 簡単のため，行列は 3x3 とする.
  double b[ ]; // 簡単のため，ベクトルは 3 次元とする.
} nnlayer;
```

```
double sigma(double x){//relu と呼ばれる活性化関数.
    if(x > 0){
        return x;
    }
    return 0;
}

void func(nnlayer nn, double x[], double f[]){
    // f に sigma(A x + b) を代入.
    return;
}

int main(void){
    double A[3][3]={{1,2,3},{4,5,6},{7,8,9}};
    double b[3]={1,2,3};
    double x[3]={0.1,-0.2,0.3};
    double f[3];
    int i,j;
    nnlayer nn;

    nn ... ; //nn が保持している A や b を，上で宣言した A，b で初期化.

    func(nn, x, f);// ニューラルネットワークの計算を実行.
    for(i=0;i<3;i++){
        printf("%lf \n", f[i]);//計算結果を表示.
    }

    return 0;
}
```

※ 実際にニューラルネットワークを使うには，やりたいことに合わせたデータを沢山集め，ここで言う行列A，bをうまく調節する必要があります．そのためには，最適化のアルゴリズムや確率･統計などの数学，データをうまく扱うためのプログラミング技術など，沢山の勉強が必要となります．また，関数$f(x)$としても，やりたいことに合わせて，もっと複雑なものを設計します．

解答例

```
//struct0.c
#include <stdio.h>

typedef struct{
  double A[3][3]; // 簡単のため，行列は 3x3 とする.
  double b[3]; // 簡単のため，ベクトルは 3 次元とする.
} nnlayer;

double sigma(double x){//relu と呼ばれる活性化関数.
    if(x > 0){
        return x;
    }
```

```
    return 0;
}

void func(nnlayer nn, double x[], double f[]){
    // f に sigma(A x + b) を代入.
    int i,j;
    for(i=0;i<3;i++){
        f[i]=0;
        for(j=0;j<3;j++){
                f[i]=f[i]+nn.A[i][j]*x[j];
        }
    }
    for(i=0;i<3;i++){
        f[i] = sigma(f[i]);
    }
    return;
}

int main(void){
    double A[3][3]={{1,2,3},{4,5,6},{7,8,9}};
    double b[3]={1,2,3};
    double x[3]={-2,-2,3};
    double f[3];
    int i,j;
    nnlayer nn;

    //nn が保持している A や b を，上で宣言した A，b で初期化.
    for(i=0;i<3;i++){
        nn.b[i] = b[i];
    }
    for(i=0;i<3;i++){
        for(j=0;j<3;j++){
           nn.A[i][j] = A[i][j];
        }
    }

    func(nn, x, f);// ニューラルネットワークの計算を実行.
    for(i=0;i<3;i++){
        printf("%lf \n", f[i]);//計算結果を表示.
    }

    return 0;
}
```

構造体の基本的な文法

例題のような問題を解くためには，行列Aとベクトルbのような，**複数の変数を一つにまとめた新しい型を作成**できると便利である．C言語では，そのような機能として構造体が用意されている．

このように変数をまとめられると，データ解析などで，データをまとめて1つの変数としてみなしたい場合などに便利である．

文法

```
struct 自分で決めた新しい型の名前{
  1つ目の変数の宣言;
  2つ目の変数の宣言;
  3つ目以降の変数の宣言
}; //最後にセミコロン ; をつける.

int main(void){
    struct 自分で決めた新しい型の名前 変数名;//変数の宣言
    新しい型の変数の名前.新しい型の変数がもつ各変数の名前 = ... ;//新しい型に含まれる各変数を使うときには「.」をつける
    return 0;
}
```

あるいは

```
typedef struct{
  1つ目の変数の宣言;
  2つ目の変数の宣言;
  3つ目以降の変数の宣言
} 自分で決めた新しい型の名前; //最後にセミコロン ; をつける.

int main(void){
    自分で決めた新しい型の名前 変数名;//変数の宣言
    return 0;
}
```

- 構造体の中に含まれている変数を**メンバー**と呼ぶ.
- 構造体の変数に含まれている各メンバーを使うためには，変数名に　.　をつけて，その後にメンバー変数の名前を書く.

例1）長方形とその面積をまとめた型

```
// struct1.c
#include <stdio.h>

struct rect1{// rect1 という名前の新しい構造体を作った.
        double tate; // この構造体は，3つの double 型変数をまとめたもの.
```

```
        double yoko; // 構造体の中の変数の型は異なるものでもよい .
        double menseki; // また, 配列などが含まれていてもよい.
};

typedef struct{// 新しい構造体を作り, rect2 という新しい型として登録した.
        double tate;
        double yoko;
        double menseki;
} rect2; // 名前を後ろに書く. セミコロン ; をつけるのを忘れずに.

int main(void){
        double tate = 2.0;
        double yoko = 3.0;

        struct rect1 r1;//1番目の方法で宣言した型の変数はこのように作る.
        rect2 r2;//2番目の方法で宣言した型の変数を作るときには, struct を書かない.

        r1.tate = tate;
        r1.yoko = yoko;
        r1.menseki = r1.tate*r1.yoko;
        printf("%lf\n", r1.menseki);

        r2.tate = tate;
        r2.yoko = yoko;
        r2.menseki = r2.tate*r2.yoko;
        printf("%lf\n", r2.menseki);

        return 0;
}
```

例2）構造体の中の変数の型は混ざっていてもよい.

```
// struct2.c
#include <stdio.h>
#include <string.h>

struct student{
    char namae[50];
    char bangou[8];
    int gakunen;
};

int main(void){
    struct student stu;
    strcpy(stu.namae,"uriko");//文字列は（配列なので） stu.namae = "uriko" のように代入できないことに注意.
    strcpy(stu.bangou,"123456t");
    stu.gakunen=1;

    printf("学籍番号 %s の学生 %s の学年は %d です. ", stu.bangou, stu.namae, stu.gakunen);
    return 0;
```

```
}
```

練習問題

上のプログラムについて，

- 構造体を定めている部分を typedef を使った書き方に書き換えてみよ．
- 身長を表す double 型の変数 height を追加してみよ．
- stu の height を100に設定せよ．
- 新しく student 型の変数 uribo を作成し，それに stu をコピーせよ．

  ```
  uribo = stu;
  ```

- 変数 uribo のメンバー変数 namae を "uribo" に変更せよ．

構造体のポインタや配列

構造体についても，これまでと同様，ポインタや配列を使うことが出来る．

```
// struct3.c
#include <stdio.h>
#include <string.h>

struct student{
    char namae[50];
    char bangou[7];
    int gakunen;
};

int main(void){
    struct student uriko;
    struct student *puriko;
    struct student stu[3];

    strcpy(uriko.namae,"uriko");
    strcpy(uriko.bangou,"123456t");
    uriko.gakunen=1;

    puriko = &uriko;
    stu[0] = *puriko;

    printf("学籍番号 %s の学生 %s の学年は %d です. ", stu[0].bangou, stu[0].namae,
stu[0].gakunen);
    return 0;
```

```
}
```

配列を引数とする変数も作れる．データ解析などで，データをまとめて処理する関数などを作るのに便利である．

```c
// struct4.c
#include <stdio.h>
#include <string.h>

struct student{
    char namae[50];
    char bangou[7];
    int gakunen;
};

// 構造体の配列 stu を受け取り，それぞれの要素について namae を表示.
void func(struct student stu[], int n){
    int i;
    for(i=0;i<n;i++){
        printf("%s\n", stu[i].namae);
    }
}


int main(void){
    struct student stu[2];

    strcpy(stu[0].namae,"uriko");
    strcpy(stu[0].bangou,"123456t");
    stu[0].gakunen=1;

    strcpy(stu[1].namae,"uribo");
    strcpy(stu[1].bangou,"234567t");
    stu[1].gakunen=2;

    func(stu, 2);

    return 0;
}
```

練習問題

テキストファイルに書かれているデータを読み込み，その平均値を計算するプログラムを作りたい．例えば，ファイル data.txt には，2021年7月から2022年7月までの神戸市の気温，降水量（単位はmm），日照時間（単位は時間）が保存されている．このファイルを読み込み，平均気温を計算するプログラムとなるように，以下のプログラムを完成させよ．

※ ファイルを読み込む部分は作成済みなので，平均を計算するプログラムだけを完成させればよい．

```
//struct5.c
#include <stdio.h>

typedef struct{//ある時刻，ある場所における気温を保存する構造体
    double kion;//気温
    double ame;//降水量
    double hare;//日照時間
} data;

double mean(d[], n){//気温の平均値を求めたい．d[] を構造体の配列，n をその大きさとする．
    int i;
    double mean;
    mean = 0.0;
    for(i=0;i<n;i++){
        mean = mean +    ;
    }
    mean=mean/i;//日照時間が8時間より長い日のみにする場合は，そのような日が何日あったかを数える必要がある．

    return mean;
}

int main(void){
    int i;
    int n=366;
    data dat[366];
    double d1, d2, d3;
    for(i=0;i<n;i++){
        scanf("%lf,%lf,%lf", &d1, &d2, &d3);//入力は標準入力からとしておき，  < を使う．
        dat[i].kion = d1;
        dat[i].ame = d2;
        dat[i].hare = d3;
    }

    printf("%lf \n", mean(dat, n));
    return 0;
}
```

書き換えられたら，データが data.txt に入っているので，以下のようにして実行してみよ．

```
gcc struct5.c
./a.out < data.txt
```

このように実行すると，

- dat[i].kion に i 日目の気温が
- dat[i].ame に i 日目の降水量が
- dat[i].hare に i 日目の日照時間

がそれぞれ保存される.

練習問題2

練習問題1のプログラムをさらに書き換え，日照時間が8時間を超える日のみについての平均気温を計算してみよ.

ヒント

- 平均を計算する部分に「日照時間が8時間を超えたかどうか」を判定する if 文を追加する.
- また，そのような日が何日あったのかを数え，平均を計算するときに，その数字で割るようにする.

```
double mean(d[], n){
    int i;
    double mean;
    int m=0;//日照時間が長い日が何日あったかを記録.
    mean = 0.0;
    for(i=0;i<n;i++){
        もしも，日照時間が長ければ，以下を実行 {
            mean = mean +   ;
            m++;
        }
    }
    mean=mean/ ;//割り算も修正が必要

    return mean;
}
```

アロー演算子

構造体のポインタを利用して，そのポインタが表す変数のメンバを使おうとすると，例えば，下のようにしたくなる.

```
struct student uriko;
struct student *puriko;

strcpy(uriko.namae,"uriko");
strcpy(uriko.bangou,"123456t");
uriko.gakunen=1;

puriko = &uriko;

printf("%d\n", *puriko.gakunen); //ポインタの表す変数のメンバにアクセスしたい.
```

しかし，C言語では，演算の優先順位が決まっていたが，実は，「* 」よりも「 . 」のほうが優先順位が高いので，*puriko.gakunen の部分は

```
*(puriko.gakunen)
```

のように処理されてしまう．すると，puriko はポインタであって，構造体の変数ではないので，エラーとなってしまう．従って，

```
(*puriko).gakunen
```

のように書かなくてはいけないが，毎回，このように書くのは面倒である．そこで，アロー演算子という

- 構造体を表すポインタについて，
- そのポインタに対応する変数のメンバにアクセスする

ことが出来る演算子が定義されている．

文法

```
構造体を示すポインタ->構造体のメンバ変数
```

上の例であれば，

```
puriko->gakunen
```

と書けばよい．これを使うと，構造体に含まれる変数を書き換えるプログラムが簡単にかけるようになる．

```
//struct6.c
#include <stdio.h>
```

```c
#include <string.h>

struct student{
    char namae[50];
    char bangou[7];
    int gakunen;
};

void change_grade(int g, struct student *pstu){
    pstu->gakunen = g;//(*pstu).gakunen と書く代わりに，このように書いてよい.
    return;
}

int main(void){
    struct student uriko;
    struct student *puriko;
    struct student stu[2];

    strcpy(uriko.namae,"uriko");
    strcpy(uriko.bangou,"123456t");
    uriko.gakunen=1;

    change_grade(2, &uriko);
    printf("%d\n",uriko.gakunen);//2が表示される.

    return 0;
}
```

練習問題

神戸市の気温，降水量（単位はmm），日照時間（単位は時間）のデータを読み取り，平均気温を計算するプログラムを書き換え，気温，降水量，日照時間の各変数の平均を計算するプログラムを作成したい．そのようになるように，以下のプログラムを完成させよ．

```c
//struct7.c
#include <stdio.h>

typedef struct{//ある時刻，ある場所における気温を保存する構造体
    double kion;//気温
    double ame;//降水量
    double hare;//日照時間
} data;

void mean(d[], n, *result){//平均値を求め，resultに記録する.
    int i;
    double mkion=0.0, mame=0.0, mhare=0.0;

    for(i=0;i<n;i++){
```

```
        mkion = mkion +   ;
        mame = ;
        mhare = ;
    }
    mkion=mkion/i;
    mame = ;
    mhare = ;

    resultの表す変数の kion に mkionを代入;
    resultの表す変数の ame に mameを代入;
    resultの表す変数の hare に mhareを代入;

    return;
}

int main(void){
    int i;
    int n=366;
    data dat[366];
    data result;
    double d1, d2, d3;
    for(i=0;i<n;i++){
        scanf("%lf,%lf,%lf", &d1, &d2, &d3);//入力は標準入力からとしておき, < を使う.
        dat[i].kion = d1;
        dat[i].ame = d2;
        dat[i].hare = d3;
    }
    mean(dat, n, &result);
    printf("平均気温: %lf 平均降水量: %lf 平均日照時間: %lf \n", result.kion, result.ame,
result.hare);
    return 0;
}
```

宿題：模擬テスト

今回の宿題は，試験に備えて，これまでの内容に関する模擬テストとします．10回目まで，11回目以降の2つの試験がBEEFに掲載されていますので，解いてみてください．なお，今回については，2つの試験のそれぞれについて，提出すれば満点，提出がなければ零点として採点します．分からないところもあるかもしれませんが，自分なりに解答してみてください．

締切：7月25日（月）23：59