1　ニューラルネットワーク

ニューラルネットワーク
ニューラルネットワーク (Neural network　神経回路網) は、脳機能に見られるいくつかの特性を計算機上のシミュレーションによって表現することを目指した数学モデルである。研究の源流は生体の脳のモデル化であるが、神経科学の知見の改定などにより次第に脳モデルとは乖離が著しくなり、生物学や神経科学との区別のため、人工ニューラルネットワーク (Artificial Neural Network: ANN　人工神経回路網) とも呼ばれる。

シナプスの結合によりネットワークを形成した人工ニューロン（ノード）が、学習によってシナプスの結合強度を変化させ、問題解決能力を持つようなモデル全般を指す。狭義には誤差逆伝播法を用いた多層パーセプトロンを指す場合もあるが、これは誤った用法である。

ニューラルネットワークは、教師信号（正解）の入力によって問題に最適化されていく教師あり学習と、教師信号を必要としない教師なし学習に分けられる。明確な解答が用意される場合には教師あり学習が、データ・クラスタリングには教師なし学習が用いられる。結果としていずれも次元削減されるため、画像や統計など多次元量のデータでかつ線形分離不可能な問題に対して、比較的小さい計算量で良好な解を得られることが多い。このことから、パターン認識やデータマイニングをはじめ、さまざまな分野において応用されている。

2　ニューラルネットワークの歴史

歴史
1943 年、ウォーレン・マカロックとウォルター・ピッツが形式ニューロンを発表した。
1958 年、フランク・ローゼンブラットがパーセプトロンを発表した。
1969 年、マービン・ミンスキーとシーモア・パパートが著書『パーセプトロン』の中で、単純パーセプトロンは線形分離不可能なパターンを識別できない事を示した。この後、ニューラルネットワーク研究は一時衰退した。
1982 年、ジョン・ホップフィールドによってホップフィールドモデルが提案された。
1986 年、デビッド・ラメルハートらにより誤差逆伝播法（バックプロパゲーション）が提案された。これにより、ニューラルネットワークの研究が再び広く行われるようになる。
(ウィキペディア より)

3 課題

今回のプログラムも、バグが潜んでいます。
このプログラムを動作させると下記の様な出力が得られます。

```
-2.000000 3.000000 -1.000000 -2.000000 1.000000 1.000000
-0.564737 -0.865945 38367246860006545960628422763846499498542930000000000000000000000
学習データの個数:7
nan
-2.000000 3.000000 -1.000000 -2.000000 1.000000 1.000000
nan nan 38367246860006545960628422763846499498542930000000000000000000000000000000000
0 0.000000 0.000000 -5.250031 nan
1 nan 0.000000 0.000000 nan
2 0.000000 0.000000 0.000000 nan
3 0.000000 0.000000 0.000000 nan
4 0.000000 1.000000 0.000000 nan
5 1.000000 0.000000 0.000000 nan
6 1.000000 1.000000 1.000000 nan
```

本来なら 2 行目が

```
-0.564737 -0.865945 -0.513423
```

と出力されるべきです。

4 パーセプトロン型のニューラルネット学習 p.c

パーセプトロン型のニューラルネット学習プログラムを下記に示します。

```
/*
  p.c
  パーセプトロン型のニューラルネット学習
  使い方
  $./p <（学習データセットのファイル名）
  誤差の推移や、学習結果となる結合係数などを出力します
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <limits.h>

#define INPUTNO 2
#define HIDDENNO 2
#define ALPHA 20
#define SEED 65535
#define MAXINPUTNO 100
```

```
#define BIGNUM 100
#define LIMIT 0.001

double s(double u);
void   initwh(double wh[HIDDENNO][INPUTNO + 1]);
void   initwo(double wo[HIDDENNO + 1]);
double drnd(void);
void   print(double wh[HIDDENNO][INPUTNO + 1]
             , double wo[HIDDENNO + 1]);
double forward(double wh[HIDDENNO][INPUTNO + 1]
             , double wo[HIDDENNO + 1], double hi[]
             , double e[INPUTNO + 1]);
void   olearn(double wo[HIDDENNO + 1], double hi[]
             , double e[INPUTNO + 1], double o);
int    getdata(double e[][INPUTNO + 1]);

int main()
{
  double wh[HIDDENNO][INPUTNO + 1];
  double wo[HIDDENNO + 1];
  double e[MAXINPUTNO][INPUTNO + 1];
  double hi[HIDDENNO + 1];
  double o;
  double err = BIGNUM;
  int    i,j;
  int    n_of_e;

  srand(SEED);

  initwh(wh);
  initwo(wo);
  print(wh,wo);

  n_of_e = getdata(e);
  printf("学習データの個数:%d\n", n_of_e);

  while(err > LIMIT){
    err = 0.0;
    for(j = 0; j < n_of_e; ++j){
      o = forward(wh, wo, hi, e[j]);
      olearn(wo, hi, e[j], o);
      err += (o - e[j][INPUTNO]) * (o - e[j][INPUTNO]);
    }

    printf("%lf\n", err);
  }

  print(wh, wo);
```

```
  for(i = 0; i < n_of_e; ++i){
    printf("%d ", i);
    for(j = 0; j < INPUTNO + 1; ++j){
      printf("%lf ", e[i][j]);
    }
    o = forward(wh, wo, hi, e[i]);
    printf("%lf\n", o);
  }

  return 0;
}

int getdata(double e[][INPUTNO + 1])
{
  int n_of_e;
  int j = 0;

  while(scanf("%lf", &e[n_of_e][j]) != EOF){
    ++j;
    if(j > INPUTNO){
      j = 0;
      ++n_of_e;
    }
  }
  return n_of_e;
}

void olearn(double wo[HIDDENNO + 1], double hi[]
          , double e[INPUTNO + 1], double o)
{
  int    i;
  double d;

  d = (e[INPUTNO] - o) * o * (1 - o);
  for(i = 0; i < HIDDENNO; ++i){
    wo[i] += ALPHA * (-1.0) * d;
  }
}

double forward(double wh[HIDDENNO][INPUTNO + 1]
             , double wo[HIDDENNO + 1], double hi[]
             , double e[INPUTNO + 1])
{
  int    i,j;
  double u;
  double o;

  for(i = 0; i < HIDDENNO; ++i){
    u = 0;
```

```
    for(j = 0; j < INPUTNO; ++j){
      u += e[j] * wh[i][j];
    }
    u -= wh[i][j];
    hi[i] = s(u);
  }

  o = 0;
  for(i = 0; i < HIDDENNO; ++i){
    o += hi[i] * wo[i];
  }
  o -= wo[i];

  return s(o);
}

void   print(double wh[HIDDENNO][INPUTNO + 1]
             , double wo[HIDDENNO + 1])
{
  int i,j;
  for(i = 0; i < HIDDENNO; ++i){
    for(j = 0; j < INPUTNO + 1; ++j){
      printf("%lf ", wh[i][j]);
    }
  }
  printf("\n");

  for(i = 0; i < HIDDENNO + 1; ++i){
    printf("%lf ", wo[i]);
  }
  printf("\n");
}

void   initwh(double wh[HIDDENNO][INPUTNO + 1])
{
  int i,j;
/*
  for(i = 0; i < HIDDENNO; ++i){
    for(j = 0; j < INPUTNO + 1; ++j){
      wh[i][j] = drnd();
    }
  }
*/

  wh[0][0] = -2;
  wh[0][1] = 3;
  wh[0][2] = -1;
  wh[1][0] = -2;
  wh[1][1] = 1;
```

```
  wh[1][2] = 1;

}

void   initwo(double wo[HIDDENNO + 1])
{
  int i;

  for(i = 0; i < HIDDENNO; ++i){
    wo[i] = drnd();
  }
}

double drnd(void)
{
  double rndno;

  while((rndno = (double)rand()/RAND_MAX) == 1.0);
  rndno = rndno * 2 - 1;

  return rndno;
}

double s(double u)
{
  return 1.0 / (1.0 + exp(-u));
}
```

5 and.txt

テキストファイル and.txt には、論理積 (AND) を意味する学習データが格納されている。

```
0 0 0
0 1 0
1 0 0
1 1 1
```

6 「p」の操作方法

「p」の操作方法の例

```
./p < and.txt
```

上記の「./」は、Linux でのプログラムを実行する時の例です。
ウィンドウズでは、p < and.txt「Enter」と入力してください。

7　プログラムの内部構造

デバッグをするために、プログラムの内部構造を理解していると有効です。下記に、このプログラムの内部構造の説明を「はじめての機会学習」204 頁から転記します。

パーセプトロンの学習プログラムである「p.c」について、内部の構造を簡単に説明します。
まず main() 関数です。main() 関数の冒頭では、乱数の初期化に続いて、中間層と出力層の結合荷重の初期値を初期化します。この操作には、それぞれ initwh() 関数と initwo() 関数を下請け関数として用ます。初期化終了後、print() 関数を用いて結合荷重の初期化を標準出力に出力します。

続いて、getdata() 関数を用いて学習データセットを読み込みます。読み込んだ結果は、e[][] 配列に格納します。また、getdata() 関数は学習データセットの個数を戻り値として返すので、その値を変数 n_of_e に保存しておきます。以上の初期化作業に続いて、パーセプトロンの学習を行います。この学習は、main() 関数の while 文にによる繰り返しにより実行されます。学習の終了条件は、学習データセットに対するパーセプトロンの出力の誤差の 2 乗の積算値 err が、記号定数 LIMIT 未満となることです。

while 文の内部では、学習データのそれぞれの値について学習を行うために、さらに for 文による繰り返し処理を行います。この for 文の内部では、まず forward() 関数によってパーセプトロンの出力 o を計算します。次に出力層の結合係数 wo[] の値を学習します。このために、olearn() 関数を呼び出します。olearn() 関数では、o の値や学習データセット内の正解出力の値である e[j]、さらに中間層の出力 hi[] を用いて、結合荷重を学習します。また、代入文では、出力の誤差の値を積算します。以上の処理の後、while 文の最後の部分では、誤差の積算値 err を出力します。

以上の学習処理の終了後、学習結果である結合荷重の値を print() 関数を用いて出力しています。また学習データセットの値をパーセプトロンに与えた場合の出力を計算し、その値を出力しています。以上が main() 関数内部での処理の概要です。

以下、下請け関数について概要を説明します。最初に getdata() 関数では、標準入力に与えられた学習データを読み込み、配列 e[][] に順次格納します。出力層の結合荷重を学習する olearn() 関数では、計算手順に従って、結合荷重 wo[] の値を更新します。forward() 関数は、パーセプトロンの出力 o を計算

します。forward() 関数は出力 o の値を戻り値とするとともに、計算の途中で求める中間層の出力値 hi[] も配列に格納して呼び出し側に戻します。print() 関数は、パーセプトロンの結合荷重の値を出力します。

中間層の結合荷重を初期化する関数 intwo() は、2 通りの使い方ができます。1 つは乱数により結合荷重をあたえる方法であり、もう 1 つは中間層の結合荷重をあらかじめ定数として与える方法です。

intiwo() において、定数で結合荷重の初期値を変更することができます。

ソース最後には、シグモイド関数の値を計算する s() 関数が置かれます。

8 例題 52 「C 言語」238 頁

```
/*
  list.c
*/

#include <stdio.h>
#include <stdlib.h> //<process.h> ウィンドゥズでコンパイルするとき

int main(int argc, char *argv[])  /* C>list [filename] */
{
  int  count = 0;
  char buf[256];
  FILE *fp;

  if((fp = fopen(argv[1],"r")) == NULL){
    printf("Can't open file\n");
    exit(1);
  }

  while(fgets(buf, 256, fp) != NULL){
    printf("%04d:\t%s", ++count, buf);
  }

  fclose(fp);

  return 0;
}
```