ニューラルネット入門

栗田多喜夫
脳神経情報研究部門
産業技術総合研究所
E-mail: takio-kurita@aist.go.jp

1 はじめに

人間は脳の神経回路網 (ニューラルネットワーク) を使って、非常に優れた情報処理を行っています。人間や動物の脳には、非常にたくさんのニューロンがあり、それらは非常に複雑に絡み合って情報をやり取りしています。そうした複雑な神経回路網の上でのニューロン間の情報のやり取りによって、優れた情報処理が実現されています。一方、現在のコンピュータは、1 個の CPU でプログラムとして与えられた命令を逐次的に高速に処理する方式を取っており、脳の情報処理の方式とはかなり違っています。また、現在のコンピュータは、プログラムとして予め与えられた命令を忠実に実行することは得意ですが、新しい環境に適応したり、学習により新しい能力を身に付けたりすることはあまり得意ではありません。音声を聞き分けたりするパターン認識の能力や直観的な判断能力、新しい環境への適応能力や学習能力では、我々の脳は現在のコンピュータに比べてはるかに勝っています。我々の脳では、ニューロン間の結合の強さを変化さることにより学習が行われていると考えられています。(人工) ニューラルネットワークは、脳を真似て多数のニューロンを結合したネットワーク上での情報処理をさせようとするものです。この講義では、その基本となる学習の考え方あるいは学習のアルゴリズムについて理解してもらえればと思います。

2 最急降下法

多くの場合、学習の問題は、与えられた評価関数を最適とするようなパラメータを求める問題として定式化されます。従って、学習のためには、その最適化問題を解くための手法が必要になります。最適化手法には、簡単なものから高速性や安定性のために工夫した複雑手法まで、多くの手法がありますが、ここでは、最も簡単な最適化手法のひとつである最急降下法と呼ばれる最適化手法の基本的な考え方について理解し、そのプログラムを作ってみることにします。

例題として以下のような問題を考えてみることにします。

問題 1. あるパラメータ a の良さの評価尺度が以下のような 2 次の関数

$$f(a) = (a - 1.0)^2 \tag{1}$$

で与えられたとします。このとき、この評価関数が最小となるパラメータ a の (最適解) を求めなさい。

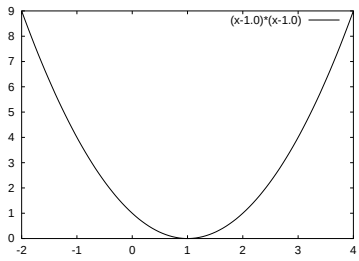
このような問題は、一般に最適化問題と呼ばれています。図 1(a) に評価関数のグラフを示します。この問題のように評価関数 $f(a)$ がパラメータ a に関して 2 次の関数の場合には、最適なパラメータはただひとつに決まり、その解も以下のような方法で簡単に求まります。

解析的な解法 式 (1) のパラメータ a に関する微分を求めると

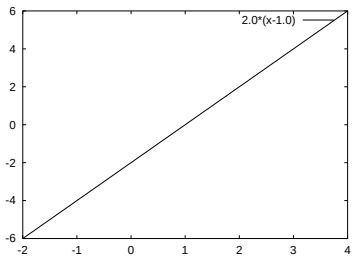
$$\frac{\partial f(a)}{\partial a} = 2(a - 1.0) \tag{2}$$

となります。ここで考えている 2 次の評価関数の場合、微分が 0 となる点が最小値となりますので、この評価関数を最小とする最適な a は、

$$\frac{\partial f(a)}{\partial a} = 2(a - 1.0) = 0 \tag{3}$$



(a) $f(a)$



(b) $f(a)$ の微分

Figure 1: 評価関数 $f(a)$ およびその微分

から、$a = 1.0$ であることがわかります。実際、式 (1) の a に 1.0 を代入してみると、$f(1.0) = (1.0 - 1.0)^2 = 0.0$ となります。評価関数 $f(a)$ は 0 以上の関数（非負の関数）であることから、$a = 1.0$ で最小値 0.0 を取ることが確かめられます。

最急降下法 最急降下法は、ある適当な初期値 (初期パラメータ) からはじめて、その値を繰り返し更新する (修正する) ことにより、最適なパラメータの値を求める方法 (繰り返し最適化手法) の最も基本的で簡単な方法です。

問題 1 のような評価関数が最小となるパラメータを求める問題では、最急降下法でのパラメータの更新は、

$$a^{(k+1)} = a^{(k)} - \alpha \frac{\partial f(a)}{\partial a}|_{a=a^{(k)}} \tag{4}$$

のようになります。ここで、$a^{(k)}$ は、k 回目の繰り返して得られたパラメータ a の推定値で、$\frac{\partial f(a)}{\partial a}|_{a=a \wedge (k)}$ は、$a = a^{(k)}$ での評価関数のパラメータ a に関する微分値です。また、α は、1 回の繰り返しでどれくらいパラメータを更新するかを制御する小さな正の定数で、学習係数と呼ばれたりします。つまり、最急降下法では、パラメータの値を微分値と逆の方向にちょっとだけ変化させて徐々に最適なパラメータに近づけて行きます。そのため、学習係数 α の値を大きくすると 1 回の繰り返しでパラメータの値を大きく更新できますが、大きすぎると最適なパラメータの近くで値が振動してしまったり、発散してしまったりします。逆に、小さくしすぎると 1 回の更新ではパラメータの値がほとんど修正されず、最適なパラメータが求まるまでの繰り返し回数が多く必要になります。そのため、最急降下法では、この値を適切に設定することが重要です。

それでは、問題 1 の最適なパラメータ a の値を最急降下法で求めるための具体的な更新式を求めてみましょう。

評価関数 $f(a)$ のパラメータ a に関する微分は、先に求めたように

$$\frac{\partial f(a)}{\partial a} = 2(a - 1.0) \tag{5}$$

のようになります。これを最急降下の更新式に代入すると、パラメータの更新式は、

$$a^{(k+1)} = a^{(k)} - 2\alpha(a^{(k)} - 1.0) \tag{6}$$

となります。図 1(b) に評価関数 $f(a)$ のパラメータ a に関する微分 $\frac{\partial f(a)}{\partial a}$ のグラフを示します。このグラフからパラメータの現在の推定値が 1.0 以上の場合には、微分は正となり、現在の a の推定値より小さな値に更新され、逆に 1.0 以下の場合には、微分は負となり、現在の推定値より大きな値に更新されます。その結果、いずれの場合でも 1.0 に近付く方向に更新されるようになります。

この更新式を使って評価関数が最小となる a の値（最適解）を求めるプログラムを作ると、以下のようになります。

```
/*
 * Program to find the optimum value
 * which minimizes the function f(a) = (a - 1.0)^2
```

```
 * using Steepest Decent Method
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


double f(double a) {
  return((a-1.0)*(a-1.0));
}

double df(double a) {
  return(2.0*(a-1.0));
}

main() {
  double a;
  int    i;
  double alpha = 0.1; /* Learning Rate */

  /* set the initial value of a by random number within [-50.0:50.0] */
  a = 100.0 * (drand48() - 0.5);
  printf("Value of a at Step 0 is %f, ", a);
  printf("Value of f(a) is %f\n", f(a));

  for (i = 1; i < 100; i++) {
    /* update theta by steepest decent method */
    a = a - alpha * df(a);
    printf("Value of a at Step %d is %f, ", i, a);
    printf("Value of f(a) is %f\n", f(a));  }

}
```

ここで、関数 f は、最適化したい評価関数（２次関数）で、その微分が df です。求めたい a の最初値をランダムな値で初期化し、先の更新式に従って、100 回更新しています。更新の際の学習係数が $alpha$ で、このプログラムでは、0.1 に設定しています。

このプログラムを適当なコンピュータでコンパイルして、走らせると以下のような結果が得られます。

```
Value of a at Step 0 is -50.000000, Value of f(a) is 2601.000000
Value of a at Step 1 is -39.800000, Value of f(a) is 1664.640000
Value of a at Step 2 is -31.640000, Value of f(a) is 1065.369600
Value of a at Step 3 is -25.112000, Value of f(a) is 681.836544
Value of a at Step 4 is -19.889600, Value of f(a) is 436.375388
Value of a at Step 5 is -15.711680, Value of f(a) is 279.280248
Value of a at Step 6 is -12.369344, Value of f(a) is 178.739359
Value of a at Step 7 is -9.695475, Value of f(a) is 114.393190
Value of a at Step 8 is -7.556380, Value of f(a) is 73.211641
Value of a at Step 9 is -5.845104, Value of f(a) is 46.855451
Value of a at Step 10 is -4.476083, Value of f(a) is 29.987488
Value of a at Step 11 is -3.380867, Value of f(a) is 19.191993
Value of a at Step 12 is -2.504693, Value of f(a) is 12.282875
Value of a at Step 13 is -1.803755, Value of f(a) is 7.861040
Value of a at Step 14 is -1.243004, Value of f(a) is 5.031066
Value of a at Step 15 is -0.794403, Value of f(a) is 3.219882
Value of a at Step 16 is -0.435522, Value of f(a) is 2.060725
Value of a at Step 17 is -0.148418, Value of f(a) is 1.318864
```

```
Value of a at Step 18 is 0.081266, Value of f(a) is 0.844073
Value of a at Step 19 is 0.265013, Value of f(a) is 0.540207
Value of a at Step 20 is 0.412010, Value of f(a) is 0.345732
Value of a at Step 21 is 0.529608, Value of f(a) is 0.221269
Value of a at Step 22 is 0.623686, Value of f(a) is 0.141612
Value of a at Step 23 is 0.698949, Value of f(a) is 0.090632
Value of a at Step 24 is 0.759159, Value of f(a) is 0.058004
Value of a at Step 25 is 0.807327, Value of f(a) is 0.037123
Value of a at Step 26 is 0.845862, Value of f(a) is 0.023759
Value of a at Step 27 is 0.876690, Value of f(a) is 0.015205
Value of a at Step 28 is 0.901352, Value of f(a) is 0.009731
Value of a at Step 29 is 0.921081, Value of f(a) is 0.006228
Value of a at Step 30 is 0.936865, Value of f(a) is 0.003986
Value of a at Step 31 is 0.949492, Value of f(a) is 0.002551
Value of a at Step 32 is 0.959594, Value of f(a) is 0.001633
Value of a at Step 33 is 0.967675, Value of f(a) is 0.001045
Value of a at Step 34 is 0.974140, Value of f(a) is 0.000669
Value of a at Step 35 is 0.979312, Value of f(a) is 0.000428
Value of a at Step 36 is 0.983450, Value of f(a) is 0.000274
Value of a at Step 37 is 0.986760, Value of f(a) is 0.000175
Value of a at Step 38 is 0.989408, Value of f(a) is 0.000112
Value of a at Step 39 is 0.991526, Value of f(a) is 0.000072
Value of a at Step 40 is 0.993221, Value of f(a) is 0.000046
Value of a at Step 41 is 0.994577, Value of f(a) is 0.000029
Value of a at Step 42 is 0.995661, Value of f(a) is 0.000019
Value of a at Step 43 is 0.996529, Value of f(a) is 0.000012
Value of a at Step 44 is 0.997223, Value of f(a) is 0.000008
Value of a at Step 45 is 0.997779, Value of f(a) is 0.000005
Value of a at Step 46 is 0.998223, Value of f(a) is 0.000003
Value of a at Step 47 is 0.998578, Value of f(a) is 0.000002
Value of a at Step 48 is 0.998863, Value of f(a) is 0.000001
Value of a at Step 49 is 0.999090, Value of f(a) is 0.000001
Value of a at Step 50 is 0.999272, Value of f(a) is 0.000001
Value of a at Step 51 is 0.999418, Value of f(a) is 0.000000
Value of a at Step 52 is 0.999534, Value of f(a) is 0.000000
Value of a at Step 53 is 0.999627, Value of f(a) is 0.000000
Value of a at Step 54 is 0.999702, Value of f(a) is 0.000000
Value of a at Step 55 is 0.999761, Value of f(a) is 0.000000
Value of a at Step 56 is 0.999809, Value of f(a) is 0.000000
Value of a at Step 57 is 0.999847, Value of f(a) is 0.000000
Value of a at Step 58 is 0.999878, Value of f(a) is 0.000000
Value of a at Step 59 is 0.999902, Value of f(a) is 0.000000
Value of a at Step 60 is 0.999922, Value of f(a) is 0.000000
Value of a at Step 61 is 0.999937, Value of f(a) is 0.000000
Value of a at Step 62 is 0.999950, Value of f(a) is 0.000000
Value of a at Step 63 is 0.999960, Value of f(a) is 0.000000
Value of a at Step 64 is 0.999968, Value of f(a) is 0.000000
Value of a at Step 65 is 0.999974, Value of f(a) is 0.000000
Value of a at Step 66 is 0.999980, Value of f(a) is 0.000000
Value of a at Step 67 is 0.999984, Value of f(a) is 0.000000
Value of a at Step 68 is 0.999987, Value of f(a) is 0.000000
Value of a at Step 69 is 0.999990, Value of f(a) is 0.000000
Value of a at Step 70 is 0.999992, Value of f(a) is 0.000000
Value of a at Step 71 is 0.999993, Value of f(a) is 0.000000
Value of a at Step 72 is 0.999995, Value of f(a) is 0.000000
```

```
Value of a at Step 73 is 0.999996, Value of f(a) is 0.000000
Value of a at Step 74 is 0.999997, Value of f(a) is 0.000000
Value of a at Step 75 is 0.999997, Value of f(a) is 0.000000
Value of a at Step 76 is 0.999998, Value of f(a) is 0.000000
Value of a at Step 77 is 0.999998, Value of f(a) is 0.000000
Value of a at Step 78 is 0.999999, Value of f(a) is 0.000000
Value of a at Step 79 is 0.999999, Value of f(a) is 0.000000
Value of a at Step 80 is 0.999999, Value of f(a) is 0.000000
Value of a at Step 81 is 0.999999, Value of f(a) is 0.000000
Value of a at Step 82 is 0.999999, Value of f(a) is 0.000000
Value of a at Step 83 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 84 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 85 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 86 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 87 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 88 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 89 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 90 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 91 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 92 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 93 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 94 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 95 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 96 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 97 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 98 is 1.000000, Value of f(a) is 0.000000
Value of a at Step 99 is 1.000000, Value of f(a) is 0.000000
```

パラメータ a の更新が繰り返されると、次第に 1.0 に近付き、それと同時に評価関数 $f(a)$ の値が 0.0 に近付いて行く様子がわかると思います。

次に、応用問題として、以下のような問題の最適解を最急降下法で求めるためのプログラムも作成してみてみましょう。

応用問題 1. あるパラメータ a の良さの評価尺度が以下のような４次の関数

$$f(a) = (a - 1.0)^2(a + 1.0)^2 \tag{7}$$

で与えられたとします。このとき、この評価関数の意味で最も良いパラメータを求めなさい。

この場合には、解は一意に決まらず、初期値に依存します。いろいろと初期値を変えて解への収束の様子を調べてみてください。

ヒント この評価関数 $f(a)$ のパラメータ a に関する微分は、

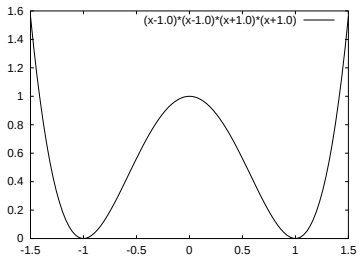
$$\frac{\partial f(a)}{\partial a} = 4.0a(a - 1.0)(a + 1.0) \tag{8}$$

のようになります。

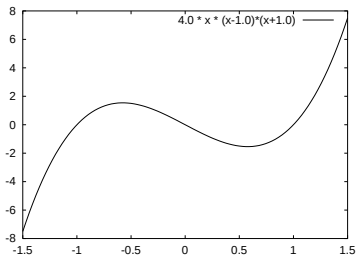
評価関数およびその微分を図 2(a) および (b) に示します。

3　最小２乗法

最小２乗法は、ある変量の組 (説明変数) とその変量に対する望みの結果 (目的変数、教師信号) が学習データとして与えられた時、説明変数から目的変数を予測するモデルを構築するための統計的データ解析手法で、最も基本的で、最も広く用いられています。ここでは、以下のような例題に対するプログラムを作ってみましょう。



(a) $f(a)$



(b) $f(a)$ の微分

Figure 2: 評価関数 $f(a)$ およびその微分

Table 1: 中学生のボール投げの記録 (t) と握力 ($x1$)、身長 ($x2$)、体重 ($x3$) のデータ

生徒番号	ボール投げ (m)	握力 (kg)	身長 (cm)	体重 (kg)
1	22	28	146	34
2	36	46	169	57
3	24	39	160	48
4	22	25	156	38
5	27	34	161	47
6	29	29	168	50
7	26	38	154	54
8	23	23	153	40
9	31	42	160	62
10	24	27	152	39
11	23	35	155	46
12	27	39	154	54
13	31	38	157	57
14	25	32	162	53
15	23	25	142	32

問題 2. 15 人の中学生のボール投げの記録 (t) と握力 ($x1$)、身長 ($x2$)、体重 ($x3$) のデータがあります。

このデータを用いて、握力 ($x1$)、身長 ($x2$)、体重 ($x3$) からボール投げの記録 (t) を予測するための線形モデル

$$y(x1, x2, x3) = a_0 + a_1 x1 + a_2 x2 + a_3 x3 \tag{9}$$

を求めなさい。

つまり、中学生のボール投げの記録 (t) と握力 ($x1$)、身長 ($x2$)、体重 ($x3$) のデータを $\{< t_l, x1_l, x2_l, x3_l > | l = 1, \ldots, 15\}$ とすると、l 番目の生徒のボール投げの記録を

$$y(x1_l, x2_l, x3_l) = a_0 + a_1 x1_l + a_2 x2_l + a_3 x3_l \tag{10}$$

で予測します。このモデルには、a_0, a_1, a_2 および a_3 の 4 個のパラメータが含まれています。これらのパラメータを学習用のデータから決めなければなりません。最小 2 乗法では、予測のための線形モデル良さの評価基準として、望みの結果とモデルが予測した結果との誤差の 2 乗の期待値（平均 2 乗誤差）を用い、2 乗誤差が最も小さくなるようなパラメータを探索します。今考えている例題では、説明変数の組 $< x1_l, x2_l, x3_l >$ に対する望みの結果が t_l で、モデルの出力が y_l ですので、その誤差 $(t_l - y_l)$ の 2 乗の平均（平均 2 乗誤差）$\varepsilon^2(a_0, a_1, a_2, a_3)$ は、

$$\varepsilon^2(a_0, a_1, a_2, a_3) \quad = \quad \frac{1}{15}\sum_{l=1}^{15} \varepsilon_l^2 = \frac{1}{15}\sum_{l=1}^{15} (t_l - y_l)^2$$

$$= \frac{1}{15}\sum_{l=1}^{15}\{t_l - (a_0 + a_1 x1_l + a_2 x2_l + a_3 x3_l)\}^2 \tag{11}$$

のようになります。

この問題では、評価関数 $\varepsilon^2(a_0, a_1, a_2, a_3)$ は、各パラメータに関して 2 次の関数ですので、以下のような方法で最適なパラメータを求めることが可能です。

最小２乗法の解法 式 (11) のパラメータ a_0 に関する微分を求めると

$$\begin{aligned}\frac{\partial \varepsilon^2}{\partial a_0} &= -2\{(\frac{1}{15}\sum_{l=1}^{15} t_l) - a_0(\frac{1}{15}\sum_{l=1}^{15} 1) - a_1(\frac{1}{15}\sum_{l=1}^{15} x1_l) - a_2(\frac{1}{15}\sum_{l=1}^{15} x2_l) - a_3(\frac{1}{15}\sum_{l=1}^{15} x3_l)\} \\ &= -2\{\bar{t} - a_0 - a_1\bar{x1} - a_2\bar{x2} - a_3\bar{x3}\} \end{aligned} \tag{12}$$

のようになります。最適な解は、この値が 0 となることが必要ですから、$\frac{\partial \varepsilon^2}{\partial a_0} = 0$ とおくと、

$$a_0 = \bar{t} - a_1\bar{x1} - a_2\bar{x2} - a_3\bar{x3} \tag{13}$$

が得られます。ここで、$\bar{t}$, $\bar{x1}$, $\bar{x2}$ および $\bar{x3}$ は、それぞれ、t, $x1$, $x2$ および $x3$ の平均値で、

$$\begin{aligned}\bar{t} &= \frac{1}{15}\sum_{l=1}^{15} t_l \\ \bar{x1} &= \frac{1}{15}\sum_{l=1}^{15} x1_l \\ \bar{x2} &= \frac{1}{15}\sum_{l=1}^{15} x2_l \\ \bar{x3} &= \frac{1}{15}\sum_{l=1}^{15} x3_l \end{aligned} \tag{14}$$

のように定義されます。今、これを、式 (10) に代入すると

$$y(x1_l, x2_l, x3_l) = \bar{t} + a_1(x1_l - \bar{x1}) + a_2(x2_l - \bar{x2}) + a_3(x3_l - \bar{x3}) \tag{15}$$

のようになります。従って、平均２乗誤差は、パラメータ a_1, a_2 および a_3 の関数として、

$$\begin{aligned}\varepsilon^2(a_1, a_2, a_3) &= \frac{1}{15}\sum_{l=1}^{15}(t_l - y_l)^2 \\ &= \frac{1}{15}\sum_{l=1}^{15}\{t_l - \bar{t} - a_1(x1_l - \bar{x1}) - a_2(x2_l - \bar{x2}) - a_3(x3_l - \bar{x3})\}^2 \end{aligned} \tag{16}$$

のように書けます。今、この平均２乗誤差をパラメータ a_1 で微分すると

$$\begin{aligned}\frac{\partial \varepsilon^2}{\partial a_1} &= -\frac{1}{15}\sum_{l=1}^{15}\{t_l - \bar{t} - a_1(x1_l - \bar{x1}) - a_2(x2_l - \bar{x2}) - a_3(x3_l - \bar{x3})\}\{x1_l - \bar{x1}\} \\ &= -\{\sigma_{t1} - a_1\sigma_{11} - a_2\sigma_{21} - a_3\sigma_{31}\} \end{aligned} \tag{17}$$

のようになります。同様に、平均２乗誤差をパラメータ a_2 および a_3 で微分すると、

$$\frac{\partial \varepsilon^2}{\partial a_2} = -\{\sigma_{t2} - a_1\sigma_{12} - a_2\sigma_{22} - a_3\sigma_{32}\} \tag{18}$$

$$\frac{\partial \varepsilon^2}{\partial a_3} = -\{\sigma_{t3} - a_1\sigma_{13} - a_2\sigma_{23} - a_3\sigma_{33}\} \tag{19}$$

のようになります。ここで、

$$
\begin{aligned}
\sigma_{11} &= \frac{1}{15}\sum_{l=1}^{15}(x1_l - \bar{x1})(x1_l - \bar{x1}),\\
\sigma_{12} &= \frac{1}{15}\sum_{l=1}^{15}(x1_l - \bar{x1})(x2_l - \bar{x2}),\\
\sigma_{13} &= \frac{1}{15}\sum_{l=1}^{15}(x1_l - \bar{x1})(x3_l - \bar{x3}),\\
\sigma_{21} &= \frac{1}{15}\sum_{l=1}^{15}(x2_l - \bar{x2})(x1_l - \bar{x1}),\\
\sigma_{22} &= \frac{1}{15}\sum_{l=1}^{15}(x2_l - \bar{x2})(x2_l - \bar{x2}),\\
\sigma_{23} &= \frac{1}{15}\sum_{l=1}^{15}(x2_l - \bar{x2})(x3_l - \bar{x3}),\\
\sigma_{31} &= \frac{1}{15}\sum_{l=1}^{15}(x3_l - \bar{x3})(x1_l - \bar{x1}),\\
\sigma_{32} &= \frac{1}{15}\sum_{l=1}^{15}(x3_l - \bar{x3})(x2_l - \bar{x2}),\\
\sigma_{33} &= \frac{1}{15}\sum_{l=1}^{15}(x3_l - \bar{x3})(x3_l - \bar{x3}),\\
\sigma_{t1} &= \frac{1}{15}\sum_{l=1}^{15}(t_l - \bar{t})(x1_l - \bar{x1}),\\
\sigma_{t2} &= \frac{1}{15}\sum_{l=1}^{15}(t_l - \bar{t})(x2_l - \bar{x2}),\\
\sigma_{t3} &= \frac{1}{15}\sum_{l=1}^{15}(t_l - \bar{t})(x3_l - \bar{x3}) \qquad (20)
\end{aligned}
$$

です。これらは、分散あるいは共分散と呼ばれています。また、これらの定義から、

$$\sigma_{12} = \sigma_{21},\ \ \sigma_{13} = \sigma_{31},\ \ \sigma_{23} = \sigma_{32} \qquad (21)$$

のような関係が成り立つことが分かります。

最適なパラメータでは、平均２乗誤差のパラメータに関する微分が 0 となるはずですので、それらを 0 とおくと、結局、

$$
\begin{aligned}
a_1\sigma_{11} + a_2\sigma_{12} + a_3\sigma_{13} &= \sigma_{t1}\\
a_1\sigma_{21} + a_2\sigma_{22} + a_3\sigma_{23} &= \sigma_{t2}\\
a_1\sigma_{31} + a_2\sigma_{32} + a_3\sigma_{33} &= \sigma_{t3} \qquad (22)
\end{aligned}
$$

のような連立方程式が得られます。従って、この連立方程式を解くためのサブルーチンがあれば、最適なパラメータが求まることになります。この連立方程式を行列とベクトルを使って表現すると、

$$\Sigma \boldsymbol{a} = \boldsymbol{\sigma} \qquad (23)$$

のようになります。ここで、行列 Σ、およびベクトル $\boldsymbol{a}, \boldsymbol{\sigma}$ は、

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix}, \quad \boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}, \quad \boldsymbol{\sigma} = \begin{pmatrix} \sigma_{t1} \\ \sigma_{t2} \\ \sigma_{t3} \end{pmatrix} \qquad (24)$$

のように定義されます。行列 Σ は、分散共分散行列と呼ばれています。もし、この行列 Σ が正則で逆行列 Σ^{-1} が存在するなら、上の連立方程式の両辺に左から逆行列 Σ^{-1} をかけて、

$$\boldsymbol{a} = \Sigma^{-1}\boldsymbol{\sigma} \tag{25}$$

のように最適なパラメータが求まります。今考えている問題では、分散共分散行列 Σ は、3×3 の行列で、その逆行列は、逆行列の公式から

$$\Sigma^{-1} = \frac{1}{|\Sigma|}\begin{pmatrix} \sigma_{22}\sigma_{33} - \sigma_{23}^2 & -\sigma_{12}\sigma_{33} + \sigma_{13}\sigma_{23} & -(-\sigma_{12}\sigma_{23} + \sigma_{13}\sigma_{22}) \\ -\sigma_{12}\sigma_{33} + \sigma_{13}\sigma_{23} & \sigma_{11}\sigma_{33} - \sigma_{13}^2 & -(\sigma_{11}\sigma_{23} - \sigma_{12}\sigma_{13}) \\ -(-\sigma_{12}\sigma_{23} + \sigma_{13}\sigma_{22}) & -(\sigma_{11}\sigma_{23} - \sigma_{12}\sigma_{13}) & \sigma_{11}\sigma_{22} - \sigma_{12}^2 \end{pmatrix} \tag{26}$$

のように求まります。ここで、$|\Sigma|$ は、行列 Σ の行列式で、$|\Sigma| = \sigma_{11}\sigma_{22}\sigma_{33} - \sigma_{11}\sigma_{23}^2 - \sigma_{12}^2\sigma_{33} - \sigma_{13}^2\sigma_{22} + 2\sigma_{12}\sigma_{13}\sigma_{23}$ です。この行列式 $|\Sigma|$ が 0 でない場合に逆行列が存在します。つまり、これが、最適なパラメータが計算できる条件になります。

この式に従って、ボール投げの記録を予測するための最適なパラメータを具体的に計算すると、

$$\begin{aligned} a_0 &= -13.21730 \\ a_1 &= 0.20138 \\ a_2 &= 0.17103 \\ a_3 &= 0.12494 \end{aligned} \tag{27}$$

のようになります。以上ののようにして学習データから最適なパラメータが求まれば、握力 ($x1 = 30$)、身長 ($x2 = 165$)、体重 ($x3 = 55$) のデータからの学生のボール投げの記録を

$$y = -13.21730 + 0.20138x30 + 0.17103x165 + 0.12494x55 = 27.91575 \tag{28}$$

のように予測することができるようになります。この問題の場合は、行列 Σ が 3×3 でしたので、手計算でも最適解が求まりましたが、一般には、連立方程式を解くサブルーチンを用いるなどして、最適解を求める必要があります。

最急降下法によるパラメータの推定 次に、最小 2 乗法の最適なパラメータを線形方程式を解かないで、最急降下法により求めるプログラムについて考えてみます。最急降下法は、適当な初期パラメータからはじめて、パラメータの値を微分値と逆の方向にちょっとだけ変化させて徐々に最適なパラメータに近づけて行く方法ですので、まずは、評価関数 (最小 2 乗法では、最小 2 乗誤差) の各パラメータでの微分を計算する必要があります。

式 (11) の最小 2 乗誤差のパラメータ a_0 に関する微分は、

$$\frac{\partial \varepsilon^2}{\partial a_0} = -2\frac{1}{15}\sum_{l=1}^{15}\{\varepsilon_l \frac{\partial \varepsilon_l}{\partial a_0}\} = -2\frac{1}{15}\sum_{l=1}^{15}\varepsilon_l = -2\frac{1}{15}\sum_{l=1}^{15}(t_l - y(x1_l, x2_l, x3_l)) \tag{29}$$

のように書けます。

同様に、最小 2 乗誤差のパラメータ a_1, a_2 および a_3 に関する微分は、

$$\begin{aligned} \frac{\partial \varepsilon^2}{\partial a_1} &= -2\frac{1}{15}\sum_{l=1}^{15}\{\varepsilon_l \frac{\partial \varepsilon_l}{\partial a_1}\} = -2\frac{1}{15}\sum_{l=1}^{15}\varepsilon_l x1_l = -2\frac{1}{15}\sum_{l=1}^{15}(t_l - y(x1_l, x2_l, x3_l))x1_l \\ \frac{\partial \varepsilon^2}{\partial a_2} &= -2\frac{1}{15}\sum_{l=1}^{15}\{\varepsilon_l \frac{\partial \varepsilon_l}{\partial a_2}\} = -2\frac{1}{15}\sum_{l=1}^{15}\varepsilon_l x2_l = -2\frac{1}{15}\sum_{l=1}^{15}(t_l - y(x1_l, x2_l, x3_l))x2_l \\ \frac{\partial \varepsilon^2}{\partial a_3} &= -2\frac{1}{15}\sum_{l=1}^{15}\{\varepsilon_l \frac{\partial \varepsilon_l}{\partial a_3}\} = -2\frac{1}{15}\sum_{l=1}^{15}\varepsilon_l x3_l = -2\frac{1}{15}\sum_{l=1}^{15}(t_l - y(x1_l, x2_l, x3_l))x3_l \end{aligned} \tag{30}$$

のようになります。

従って、最急降下法による各パラメータの更新式は、

$$
\begin{aligned}
a_0^{(k+1)} &= a_0^{(k)} - \alpha \frac{\partial \varepsilon^2}{\partial a_0}|_{a_0=a_0^{(k)}} = a_0^{(k)} + 2\alpha \frac{1}{15}\sum_{l=1}^{15}(t_l - y(x1_l, x2_l, x3_l)) \\
a_1^{(k+1)} &= a_1^{(k)} - \alpha \frac{\partial \varepsilon^2}{\partial a_1}|_{a_1=a_1^{(k)}} = a_1^{(k)} + 2\alpha \frac{1}{15}\sum_{l=1}^{15}(t_l - y(x1_l, x2_l, x3_l))x1_l \\
a_2^{(k+1)} &= a_2^{(k)} - \alpha \frac{\partial \varepsilon^2}{\partial a_2}|_{a_2=a_2^{(k)}} = a_2^{(k)} + 2\alpha \frac{1}{15}\sum_{l=1}^{15}(t_l - y(x1_l, x2_l, x3_l))x2_l \\
a_3^{(k+1)} &= a_3^{(k)} - \alpha \frac{\partial \varepsilon^2}{\partial a_3}|_{a_3=a_3^{(k)}} = a_3^{(k)} + 2\alpha \frac{1}{15}\sum_{l=1}^{15}(t_l - y(x1_l, x2_l, x3_l))x3_l
\end{aligned} \tag{31}
$$

のようになります。

では、この式を用いて、握力 ($x1$)、身長 ($x2$)、体重 ($x3$) のデータからボール投げの記録 (t) を予測するための線形モデルのパラメータを最急降下法で求めるプログラムを作成してみます。ただし、学習を安定化させるため各変数 ($x1$,$x2$,$x3$) の値を 100 で割って、

$$
x1' = \frac{x1}{100},\ \ x2' = \frac{x2}{100},\ \ x3' = \frac{x3}{100} \tag{32}
$$

のように正規化してから利用します。これにより、学習係数をパラメータ毎に変える必要がなくなると思います。

以下がそのプログラムの例です。

```
#include <stdio.h>

#define NSAMPLE 15
#define XDIM 3

main() {
  FILE *fp;
  double t[NSAMPLE];
  double x[NSAMPLE][XDIM];
  double a[XDIM+1];
  int   i, j, l;
  double y, err, mse;
  double derivatives[XDIM+1];
  double alpha = 0.2; /* Learning Rate */

  /* Open Data File */
  if ((fp = fopen("ball.dat","r")) == NULL) {
    fprintf(stderr,"File Open Fail\n");
    exit(1);
  }

  /* Read Data */
  for (l = 0; l < NSAMPLE; l++) {
    /* Teacher Signal (Ball) */
    fscanf(fp,"%lf", &(t[l]));
    /* Input input vectors */
    for (j = 0; j < XDIM; j++) {
      fscanf(fp,"%lf",&(x[l][j]));
    }
  }
```

```
  /* Close Data File */
  fclose(fp);

  /* Print the data */
  for (l = 0; l < NSAMPLE; l++) {
    printf("%3d : %8.2f ", l, t[l]);
    for (j = 0; j < XDIM; j++) {
      printf("%8.2f ", x[l][j]);
    }
    printf("\n");
  }

  /* scaling the data */
  for (l = 0; l < NSAMPLE; l++) {
    /*    t[l] = t[l] / tmean;*/
    for (j = 0; j < XDIM; j++) {
      x[l][j] = x[l][j] / 100.0;
    }
  }

  /* Initialize the parameters by random number */
  for (j = 0; j < XDIM+1; j++) {
    a[j] = (drand48() - 0.5);
  }

  /* Open output file */
  fp = fopen("mse.out","w");

  /* Learning the parameters */
  for (i = 1; i < 20000; i++) { /* Learning Loop */

    /* Compute derivatives */
    /* Initialize derivatives */
    for (j = 0; j < XDIM+1; j++) {
      derivatives[j] = 0.0;
    }

    /* update derivatives */
    for (l = 0; l < NSAMPLE; l++) {

      /* prediction */
      y = a[0];
      for (j = 1; j < XDIM+1; j++) {
y += a[j] * x[l][j-1];
      }

      /* error */
      err = t[l] - y;
      /*      printf("err[%d] = %f\n", l, err);*/

      /* update derivatives */
      derivatives[0] += err;
      for (j = 1; j < XDIM+1; j++) {
derivatives[j] += err * x[l][j-1];
      }
```

```
    }
    for (j = 0; j < XDIM+1; j++) {
      derivatives[j] = -2.0 * derivatives[j] / (double)NSAMPLE;
    }

    /* update parameters */
    for (j = 0; j < XDIM+1; j++) {
      a[j] = a[j] - alpha * derivatives[j];
    }

    /* Compute Mean Squared Error */
    mse = 0.0;
    for (l = 0; l < NSAMPLE; l++) {
      /* prediction */
      y = a[0];
      for (j = 1; j < XDIM+1; j++) {
y += a[j] * x[l][j-1];
      }

      /* error */
      err = t[l] - y;

      mse += err * err;
    }
    mse = mse / (double)NSAMPLE;

    printf("%d : Mean Squared Error is %f\n", i, mse);
    fprintf(fp, "%f\n", mse);

  }

  fclose(fp);

  /* Print Estmated Parameters */
  for (j = 0; j < XDIM+1; j++) {
    printf("a[%d]=%f, ",j, a[j]);
  }
  printf("\n");

  /* Prediction and Errors */
  for (l = 0; l < NSAMPLE; l++) {
    /* prediction */
    y = a[0];
    for (j = 1; j < XDIM+1; j++) {
      y += a[j] * x[l][j-1];
    }

    /* error */
    err = t[l] - y;

    printf("%3d : t = %f, y = %f (err = %f)\n", l, t[l], y, err);
  }

}
```

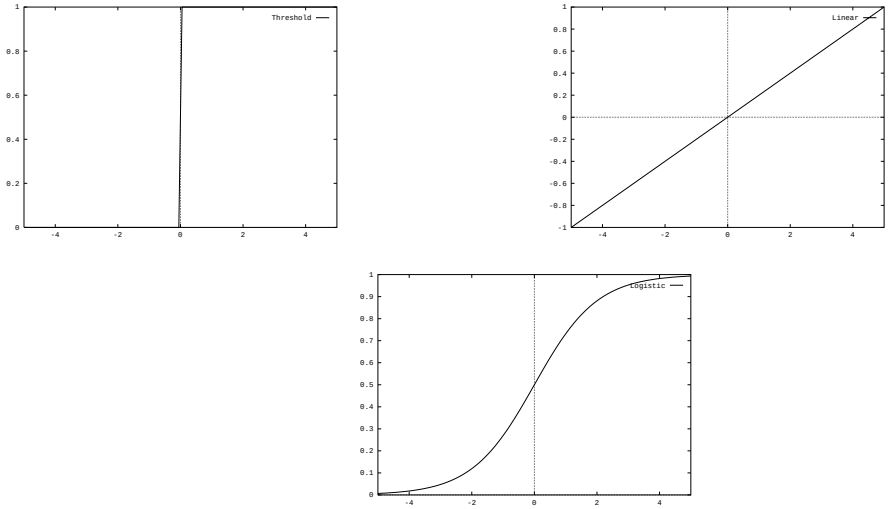
このプログラムを実行させると、

```
a[0]=-13.156891, a[1]=20.077345, a[2]=17.056200, a[3]=12.562173,
  0 : t = 22.000000, y = 21.637957 (err = 0.362043)
  1 : t = 36.000000, y = 32.064105 (err = 3.935895)
  2 : t = 24.000000, y = 27.993037 (err = -3.993037)
  3 : t = 22.000000, y = 23.243744 (err = -1.243744)
  4 : t = 27.000000, y = 27.034110 (err = -0.034110)
  5 : t = 29.000000, y = 27.601042 (err = 1.398958)
  6 : t = 26.000000, y = 27.522622 (err = -1.522622)
  7 : t = 23.000000, y = 22.581754 (err = 0.418246)
  8 : t = 31.000000, y = 30.354062 (err = 0.645938)
  9 : t = 24.000000, y = 23.088664 (err = 0.911336)
 10 : t = 23.000000, y = 26.085890 (err = -3.085890)
 11 : t = 27.000000, y = 27.723396 (err = -0.723396)
 12 : t = 31.000000, y = 28.411174 (err = 2.588826)
 13 : t = 25.000000, y = 27.556856 (err = -2.556856)
 14 : t = 23.000000, y = 20.102145 (err = 2.897855)
```

のようなパラメータが求まり、学習に用いたデータに対して、求まったパラメータを用いたモデルで予測した結果が表示されます。パラメータが先に手計算で求めた最適なパラメータと近い値に収束し、ボール投げの記録がだいたい予測できるようになっていることがわかります。

4 パーセプトロン

ここまでに、最急降下法の考え方を理解し、最小２乗法を用いてボール投げの記録を予測する線形モデルを学習するプログラムを作成しました。ここでは、この解説の本題であるニューラルネットの話題に入ります。その前に、ちょっとだけニューラルネットに関する研究の歴史的な流れをみておきます。



(a) 閾値関数　(b) 線形関数　(c) ロジスティック関数

Figure 3: 出力関数

1943 年に、McCulloch と Pitts は、M 個の２値 (± 1) の入力の組 $< x_1, x_2, \ldots, x_M >$ から２値の出力 y

を計算する神経細胞 (ニューロン) を閾値論理素子

$$y = U(\sum_{i=1}^{M} a_i x_i + a_0) \tag{33}$$

でモデル化しました。ここで、出力関数として用いられている $U(\eta)$ は、

$$U(\eta) = \begin{cases} 1, & \text{if } \eta > 0 \\ -1, & \text{if } \eta \leq 0 \end{cases} \tag{34}$$

のような閾値関数とします。図 3(a) に閾値関数を示します。McCulloch と Pitts は、このようなニューロンをたくさん相互に結合したネットワークによって任意の論理関数が表現できることを示しました。また、1949 年には、Hebb が実際の神経回路を調べて、ニューロン間の結合の強さ (結合強度) はニューロンの入力と出力が共にアクティブな場合に強化されるという Hebb の学習則を提案しました。

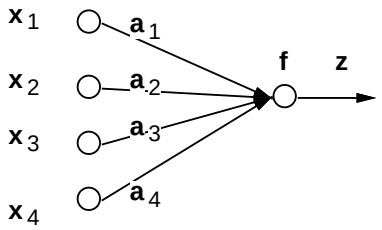


Figure 4: パーセプトロン

このような研究を背景として、1957 年に心理学者の Rosenblatt は、世界初のパターン認識のための学習機械のモデルを提案しました。そのモデルは、パーセプトロンとよばれ、その後の学習法の規範となっています。図 4 にパーセプトロンの概念図を示します。パーセプトロンは、閾値論理素子をニューロンのモデルとしていて、網膜に相当する入力層、そこからランダムに結線された連合層、そして連合層の出力を線形加重和として集めて出力を出す反応層の 3 層からなるニューラルネットワークモデルです。このネットワークモデルでは、入力層への入力とその入力に対する望みの答え (教師の答え) が与えられると、連合層からの反応層への結合重みが逐次修正されます。Rosenblatt の学習法は、まず、ネットワークに入力パターンを分類させてみて、その結果が教師の答えと違っていたら結合重みを修正するものでした。入力を完全に識別できるような課題に対しては、この学習法を繰り返すことで入力パターンを識別できるようになります。しかし、完全には識別できないような課題に対しては、いくら学習を繰り返しても解に到達できない可能性があります。

5 ADALINE

最小 2 乗法の考え方を用いるとパーセプトロンの欠点をある程度解決した学習法を導くことができます。これは、1960 年に Widrow と Hoff が提案した ADALINE(Adaptive Linear Neuron) というモデルで、閾値論理素子の線形の部分

$$y = \sum_{i=1}^{M} a_i x_i + a_0 \tag{35}$$

のみを取り出して利用するものです。このモデルでの学習は、教師の答えとネットワークの出力との平均 2 乗誤差を最小とするような結合重み $(a_0, a_1, \ldots, a_M)$ を最急降下法によって求めるものです。従って、このモデルの出力関数は、McCulloch と Pitts の閾値論理素子や Rosenblatt のパーセプトロンのように閾値関数ではなく、図 3(b) のような線形関数であるとみなすことができます。

まずは、前回のボール投げの記録を予測する最小 2 乗法のプログラムを参考に 2 種類のアヤメのデータを識別する ADALINE のプログラムを作ってみましょう。

アヤメ科のイリス・ベルシコロールとイリス・ベルジニカという 2 種類の花から、がくの長さ $(x1)$、がくの幅 $(x2)$、花弁の長さ $(x3)$、花弁の幅 $(x4)$ の 4 種類の特徴を計測したデータがあります。データ数は、

各花とも 50 個づつあります。これは、Fisher という有名な統計学者が 1936 年に線形判別関数を適用した有名なデータで、それ以来パターン認識の手法を確認する例として頻繁に用いられています。

ここでは、教師の答えとしてイリス・ベルシコロールには $(t = 1)$ を与え、イリス・ベルジニカには $(t = 0)$ を与えるものとします。がくの長さ $(x1)$、がくの幅 $(x2)$、花弁の長さ $(x3)$、花弁の幅 $(x4)$ のデータから教師の答えを予測するための ADALINE モデルは、

$$y(x1, x2, x3, x4) = a_0 + a_1x1 + a_2x2 + a_3x3 + a_4x4 \tag{36}$$

となります。最小 2 乗法の場合と全く同じように、ADALINE でも予測値と教師の答えとの平均 2 乗誤差を最小にするようなパラメータ $(a_0, a_1, a_2, a_3, a_4)$ を最急降下法で求めます。平均 2 乗誤差の各パラメータでの微分を計算して、パラメータの更新式を具体的に求めると

$$\begin{aligned}
a_0^{(k+1)} &= a_0^{(k)} + 2\alpha\frac{1}{100}\sum_{l=1}^{100}(t_l - y_l) \\
a_1^{(k+1)} &= a_1^{(k)} + 2\alpha\frac{1}{100}\sum_{l=1}^{100}(t_l - y_l)x1_l \\
a_2^{(k+1)} &= a_2^{(k)} + 2\alpha\frac{1}{100}\sum_{l=1}^{100}(t_l - y_l)x2_l \\
a_3^{(k+1)} &= a_3^{(k)} + 2\alpha\frac{1}{100}\sum_{l=1}^{100}(t_l - y_l)x3_l \\
a_4^{(k+1)} &= a_4^{(k)} + 2\alpha\frac{1}{100}\sum_{l=1}^{100}(t_l - y_l)x4_l
\end{aligned} \tag{37}$$

のようになります。ここ t_l および y_l は、それぞれ、l 番目の計測データに対する教師の答えおよび ADALINE モデルでの予測値です。また、$x1_l$、$x2_l$、$x3_l$ および $x4_l$ は、l 番目の花を計測した特徴量の計測値です。

学習した ADALINE モデルを用いてアヤメの花を識別するには、学習した ADALINE モデルに計測した特徴量を代入し、教師の答えの予測値を求め、それが 1 に近ければイリス・ベルシコロールと判定し、0 に近ければイリス・ベルジニカと判定すれば良いことになります。

具体的なプログラムは、以下のようになります。

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

#define frand()   rand()/((double)RAND_MAX)

#define NSAMPLE 100
#define XDIM 4

main() {
  FILE *fp;
  double t[NSAMPLE];
  double x[NSAMPLE][XDIM];
  double a[XDIM+1];
  int   i, j, l;
  double y, err, mse;
  double derivatives[XDIM+1];
  double alpha = 0.1; /* Learning Rate */

  /* Open Data File */
  if ((fp = fopen("niris.dat","r")) == NULL) {
    fprintf(stderr,"File Open Fail\n");
    exit(1);
  }
```

```
  /* Read Data */
  for (l = 0; l < NSAMPLE; l++) {
    /* Input input vectors */
    for (j = 0; j < XDIM; j++) {
      fscanf(fp,"%lf",&(x[l][j]));
    }
    /* Set teacher signal */
    if (l < 50) t[l] = 1.0; else t[l] = 0.0;
  }
  /* Close Data File */
  fclose(fp);

  /* Print the data */
  for (l = 0; l < NSAMPLE; l++) {
    printf("%3d : %8.2f ", l, t[l]);
    for (j = 0; j < XDIM; j++) {
      printf("%8.2f ", x[l][j]);
    }
    printf("\n");
  }

  /* Initialize the parameters by random number */
  for (j = 0; j < XDIM+1; j++) {
    a[j] = (frand() - 0.5);
  }

  /* Open output file */
  fp = fopen("mse.out","w");

  /* Learning the parameters */
  for (i = 1; i < 1000; i++) { /* Learning Loop */

    /* Compute derivatives */
    /* Initialize derivatives */
    for (j = 0; j < XDIM+1; j++) {
      derivatives[j] = 0.0;
    }
    /* update derivatives */
    for (l = 0; l < NSAMPLE; l++) {
      /* prediction */
      y = a[0];
      for (j = 1; j < XDIM+1; j++) {
y += a[j] * x[l][j-1];
      }

      /* error */
      err = t[l] - y;
      /*      printf("err[%d] = %f\n", l, err);*/

      /* update derivatives */
      derivatives[0] += err;
      for (j = 1; j < XDIM+1; j++) {
derivatives[j] += err * x[l][j-1];
      }
```

```
    }
    for (j = 0; j < XDIM+1; j++) {
      derivatives[j] = -2.0 * derivatives[j] / (double)NSAMPLE;
    }

    /* update parameters */
    for (j = 0; j < XDIM+1; j++) {
      a[j] = a[j] - alpha * derivatives[j];
    }

    /* Compute Mean Squared Error */
    mse = 0.0;
    for (l = 0; l < NSAMPLE; l++) {
      /* prediction */
      y = a[0];
      for (j = 1; j < XDIM+1; j++) {
y += a[j] * x[l][j-1];
      }

      /* error */
      err = t[l] - y;

      mse += err * err;
    }
    mse = mse / (double)NSAMPLE;

    printf("%d : Mean Squared Error is %f\n", i, mse);
    fprintf(fp, "%f\n", mse);

  }

  fclose(fp);

  /* Print Estmated Parameters */
  printf("\nEstimated Parameters\n");
  for (j = 0; j < XDIM+1; j++) {
    printf("a[%d]=%f, ",j, a[j]);
  }
  printf("\n\n");

  /* Prediction and Errors */
  for (l = 0; l < NSAMPLE; l++) {
    /* prediction */
    y = a[0];
    for (j = 1; j < XDIM+1; j++) {
      y += a[j] * x[l][j-1];
    }

    /* error */
    err = t[l] - y;

    if ((1.0 - y)*(1.0 - y) <= (0.0 - y)*(0.0 - y)) {
      if (l < 50) {
printf("%3d [Class1 : correct] : t = %f, y = %f (err = %f)\n", l, t[l], y, err);
```

```
      } else {
printf("%3d [Class1 : not correct] : t = %f, y = %f (err = %f)\n", l, t[l], y, err);
      }
    } else {
      if (l >= 50) {
printf("%3d [Class2 : correct] : t = %f, y = %f (err = %f)\n", l, t[l], y, err);
      } else {
printf("%3d [Class2 : not correct] : t = %f, y = %f (err = %f)\n", l, t[l], y, err);
      }
    }
  }

}
```

このプログラムは、先の最小 2 乗法を最急降下法で解くプログラムとほとんど同じです。アヤメのデータファイル niris.dat を読み込んで、そのデータに対して最急降下法でパラメータを求めています。教師信号が実数値ではなく、0 と 1 の 2 値で与えられるとことだけが最小 2 乗法との違いです。プログラムの最後の部分では、得られたニューラルネット（識別器）の良さを確認するために学習に用いたアヤメのデータを識別させています。ニューラルネットの出力が 0 と 1 のどちらに近いかで、どちらのアヤメかを決定しています。プログラムの実行結果は、以下のようになります。

```
Estimated Parameters
a[0]=1.239302, a[1]=0.145552, a[2]=0.139415, a[3]=-0.638937, a[4]=-0.537277,

  0 [Class1 : correct] : t = 1.000000, y = 1.003690 (err = -0.003690)
  1 [Class1 : correct] : t = 1.000000, y = 0.899888 (err = 0.100112)
  2 [Class1 : correct] : t = 1.000000, y = 0.810625 (err = 0.189375)
  3 [Class1 : correct] : t = 1.000000, y = 0.775510 (err = 0.224490)
  4 [Class1 : correct] : t = 1.000000, y = 0.752820 (err = 0.247180)
  5 [Class1 : correct] : t = 1.000000, y = 0.789633 (err = 0.210367)
  6 [Class1 : correct] : t = 1.000000, y = 0.771009 (err = 0.228991)
  7 [Class1 : correct] : t = 1.000000, y = 1.168271 (err = -0.168271)
  8 [Class1 : correct] : t = 1.000000, y = 0.943976 (err = 0.056024)
  9 [Class1 : correct] : t = 1.000000, y = 0.816619 (err = 0.183381)
 10 [Class1 : correct] : t = 1.000000, y = 0.984887 (err = 0.015113)
 11 [Class1 : correct] : t = 1.000000, y = 0.856557 (err = 0.143443)
 12 [Class1 : correct] : t = 1.000000, y = 1.043678 (err = -0.043678)
 13 [Class1 : correct] : t = 1.000000, y = 0.748846 (err = 0.251154)
 14 [Class1 : correct] : t = 1.000000, y = 1.130948 (err = -0.130948)
 15 [Class1 : correct] : t = 1.000000, y = 1.027689 (err = -0.027689)
 16 [Class1 : correct] : t = 1.000000, y = 0.694755 (err = 0.305245)
 17 [Class1 : correct] : t = 1.000000, y = 1.132590 (err = -0.132590)
 18 [Class1 : correct] : t = 1.000000, y = 0.543723 (err = 0.456277)
 19 [Class1 : correct] : t = 1.000000, y = 1.035076 (err = -0.035076)
 20 [Class2 : not correct] : t = 1.000000, y = 0.490680 (err = 0.509320)
 21 [Class1 : correct] : t = 1.000000, y = 1.041685 (err = -0.041685)
 22 [Class1 : correct] : t = 1.000000, y = 0.512358 (err = 0.487642)
 23 [Class1 : correct] : t = 1.000000, y = 0.858199 (err = 0.141801)
 24 [Class1 : correct] : t = 1.000000, y = 1.017686 (err = -0.017686)
 25 [Class1 : correct] : t = 1.000000, y = 0.977978 (err = 0.022022)
 26 [Class1 : correct] : t = 1.000000, y = 0.803766 (err = 0.196234)
 27 [Class1 : correct] : t = 1.000000, y = 0.565534 (err = 0.434466)
 28 [Class1 : correct] : t = 1.000000, y = 0.733136 (err = 0.266864)
 29 [Class1 : correct] : t = 1.000000, y = 1.300773 (err = -0.300773)
 30 [Class1 : correct] : t = 1.000000, y = 1.021680 (err = -0.021680)
```

```
31 [Class1 : correct] : t = 1.000000, y = 1.128719 (err = -0.128719)
32 [Class1 : correct] : t = 1.000000, y = 1.063775 (err = -0.063775)
33 [Class2 : not correct] : t = 1.000000, y = 0.380334 (err = 0.619666)
34 [Class1 : correct] : t = 1.000000, y = 0.659518 (err = 0.340482)
35 [Class1 : correct] : t = 1.000000, y = 0.822877 (err = 0.177123)
36 [Class1 : correct] : t = 1.000000, y = 0.848019 (err = 0.151981)
37 [Class1 : correct] : t = 1.000000, y = 0.771196 (err = 0.228804)
38 [Class1 : correct] : t = 1.000000, y = 0.981463 (err = 0.018537)
39 [Class1 : correct] : t = 1.000000, y = 0.839696 (err = 0.160304)
40 [Class1 : correct] : t = 1.000000, y = 0.797250 (err = 0.202750)
41 [Class1 : correct] : t = 1.000000, y = 0.817255 (err = 0.182745)
42 [Class1 : correct] : t = 1.000000, y = 0.995367 (err = 0.004633)
43 [Class1 : correct] : t = 1.000000, y = 1.153796 (err = -0.153796)
44 [Class1 : correct] : t = 1.000000, y = 0.848869 (err = 0.151131)
45 [Class1 : correct] : t = 1.000000, y = 1.033489 (err = -0.033489)
46 [Class1 : correct] : t = 1.000000, y = 0.930673 (err = 0.069327)
47 [Class1 : correct] : t = 1.000000, y = 0.982449 (err = 0.017551)
48 [Class1 : correct] : t = 1.000000, y = 1.273824 (err = -0.273824)
49 [Class1 : correct] : t = 1.000000, y = 0.934896 (err = 0.065104)
50 [Class2 : correct] : t = 0.000000, y = -0.337600 (err = 0.337600)
51 [Class2 : correct] : t = 0.000000, y = 0.132929 (err = -0.132929)
52 [Class2 : correct] : t = 0.000000, y = 0.026276 (err = -0.026276)
53 [Class2 : correct] : t = 0.000000, y = 0.174352 (err = -0.174352)
54 [Class2 : correct] : t = 0.000000, y = -0.113841 (err = 0.113841)
55 [Class2 : correct] : t = 0.000000, y = -0.139841 (err = 0.139841)
56 [Class2 : correct] : t = 0.000000, y = 0.269516 (err = -0.269516)
57 [Class2 : correct] : t = 0.000000, y = 0.096326 (err = -0.096326)
58 [Class2 : correct] : t = 0.000000, y = 0.043823 (err = -0.043823)
59 [Class2 : correct] : t = 0.000000, y = -0.119072 (err = 0.119072)
60 [Class2 : correct] : t = 0.000000, y = 0.345999 (err = -0.345999)
61 [Class2 : correct] : t = 0.000000, y = 0.166008 (err = -0.166008)
62 [Class2 : correct] : t = 0.000000, y = 0.118683 (err = -0.118683)
63 [Class2 : correct] : t = 0.000000, y = 0.016717 (err = -0.016717)
64 [Class2 : correct] : t = 0.000000, y = -0.188593 (err = 0.188593)
65 [Class2 : correct] : t = 0.000000, y = 0.043580 (err = -0.043580)
66 [Class2 : correct] : t = 0.000000, y = 0.277996 (err = -0.277996)
67 [Class2 : correct] : t = 0.000000, y = 0.027482 (err = -0.027482)
68 [Class2 : correct] : t = 0.000000, y = -0.500986 (err = 0.500986)
69 [Class2 : correct] : t = 0.000000, y = 0.326909 (err = -0.326909)
70 [Class2 : correct] : t = 0.000000, y = -0.013590 (err = 0.013590)
71 [Class2 : correct] : t = 0.000000, y = 0.290259 (err = -0.290259)
72 [Class2 : correct] : t = 0.000000, y = -0.152001 (err = 0.152001)
73 [Class2 : correct] : t = 0.000000, y = 0.364375 (err = -0.364375)
74 [Class2 : correct] : t = 0.000000, y = 0.124712 (err = -0.124712)
75 [Class2 : correct] : t = 0.000000, y = 0.283933 (err = -0.283933)
76 [Class2 : correct] : t = 0.000000, y = 0.415164 (err = -0.415164)
77 [Class2 : correct] : t = 0.000000, y = 0.425416 (err = -0.425416)
78 [Class2 : correct] : t = 0.000000, y = -0.052291 (err = 0.052291)
79 [Class2 : correct] : t = 0.000000, y = 0.433825 (err = -0.433825)
80 [Class2 : correct] : t = 0.000000, y = 0.083760 (err = -0.083760)
81 [Class2 : correct] : t = 0.000000, y = 0.313110 (err = -0.313110)
82 [Class2 : correct] : t = 0.000000, y = -0.123014 (err = 0.123014)
83 [Class1 : not correct] : t = 0.000000, y = 0.536005 (err = -0.536005)
84 [Class2 : correct] : t = 0.000000, y = 0.325728 (err = -0.325728)
85 [Class2 : correct] : t = 0.000000, y = -0.082091 (err = 0.082091)
```

```
 86 [Class2 : correct] : t = 0.000000, y = -0.089522 (err = 0.089522)
 87 [Class2 : correct] : t = 0.000000, y = 0.292471 (err = -0.292471)
 88 [Class2 : correct] : t = 0.000000, y = 0.444113 (err = -0.444113)
 89 [Class2 : correct] : t = 0.000000, y = 0.204709 (err = -0.204709)
 90 [Class2 : correct] : t = 0.000000, y = -0.115327 (err = 0.115327)
 91 [Class2 : correct] : t = 0.000000, y = 0.172210 (err = -0.172210)
 92 [Class2 : correct] : t = 0.000000, y = 0.132929 (err = -0.132929)
 93 [Class2 : correct] : t = 0.000000, y = -0.103840 (err = 0.103840)
 94 [Class2 : correct] : t = 0.000000, y = -0.158180 (err = 0.158180)
 95 [Class2 : correct] : t = 0.000000, y = 0.068565 (err = -0.068565)
 96 [Class2 : correct] : t = 0.000000, y = 0.193150 (err = -0.193150)
 97 [Class2 : correct] : t = 0.000000, y = 0.245497 (err = -0.245497)
 98 [Class2 : correct] : t = 0.000000, y = 0.036213 (err = -0.036213)
 99 [Class2 : correct] : t = 0.000000, y = 0.317548 (err = -0.317548)
```

この例では、3 個の間違いましたが、4 個の特徴からほぼアヤメの種類を識別できていることがわかります。

6 ロジスティック回帰モデル

パーセプトロンで用いた閾値関数は、入力が正の場合と負の場合で出力が急に変化するような不連続の関数ですので解析的な取り扱いが簡単ではありません。ADALINE モデルでは、パーセプトロンの閾値論理素子の線形の部分のみの線形モデルを用いました。しかし、線形のモデル化では、複数のニューロンを結合しても全体のモデルが線形となってしまい、非線形の関係を表現することができません。また、実際のニューロンに近いモデルを作ると言う観点でも好ましくありません。そこで、最近のニューラルネットワークでは、閾値関数の変わりに入力が負から正へ変化する時、その出力も滑らかに変化する出力関数

$$S(\eta) = \frac{\exp(\eta)}{1+\exp(\eta)} \tag{38}$$

が用いられるようになっています。この関数はロジスティック関数と呼ばれています。図 3(c) にロジスティック関数の例を示します。このロジスティック関数を出力関数として用いた最も簡単なモデル

$$y = S(\sum_{i=1}^{M} a_i x_i + a_0) \tag{39}$$

は、ロジスティック回帰モデルと呼ばれています。

次にこのロジスティック回帰モデルのパラメータの推定法について説明します。ADALINE モデルでは、平均 2 乗誤差が最小となるようなパラメータを推定しましたが、ここでは最尤法と呼ばれている方法でパラメータを推定してみます。例題としては、先程のアヤメの識別問題を考えることにします。

6.1 最尤法

式 (39) のロジスティック回帰モデルでは、出力は 0 から 1 の間の値で、イリス・ベルシコロールの場合には 1 に近い値を出力し、そうでない場合 (イリス・ベルジニカの場合) には 0 に近い値を出力することが期待されます。そこで、ロジスティック回帰モデルの出力 y をイリス・ベルシコロールである確率と解釈します。また、今考えている問題ではアヤメの種類は 2 種類のみですので、イリス・ベルジニカである確率は $1-y$ と解釈できます。従って、100 個のアヤメの計測データが得られる尤もらしさ (尤度) は、

$$L = \prod_{l=1}^{100} (y_l)^{t_l} (1-y_l)^{1-t_l} \tag{40}$$

のようにそれぞれの確率の積で定義できます。この対数をとると

$$\log(L) \quad = \quad \sum_{l=1}^{100} \{t_l \log y_l + (1-t_l)\log(1-y_l)\}$$

$$
\begin{aligned}
&= \sum_{l=1}^{100}\{t_l \log\{\frac{\exp(\eta_l)}{1+\exp(\eta_l)} + (1-t_l)\log\{\frac{1}{1+\exp(\eta_l)}\}\} \\
&= \sum_{l=1}^{100}\{t_l\eta_l - \log\{1+\exp(\eta_l)\}\} \qquad (41)
\end{aligned}
$$

となります。この尤度の対数は、一般には対数尤度と呼ばれています。尤度を最大とすることは対数尤度を最大とすることと同じですし、対数尤度を用いる方が計算が簡単になることが多いので、一般には対数尤度を最大とするパラメータが求められます。

これまでと同じように、最急降下法を適用するためには、評価関数（対数尤度）の各パラメータでの微分が必要となります。対数尤度をパラメータ a_0 で微分すると

$$
\frac{\partial \log(L)}{\partial a_0} = \sum_{l=1}^{100}\{t_l - \frac{\exp(\eta_l)}{1+\exp(\eta_l)}\} = \sum_{l=1}^{100}\{t_l - y_l\} \qquad (42)
$$

のようになります。同様に、対数尤度をパラメータ a_1、a_2、a_3 および a_4 で微分すると

$$
\begin{aligned}
\frac{\partial \log(L)}{\partial a_1} &= \sum_{l=1}^{100}\{t_l x1_l - \frac{\exp(\eta_l)}{1+\exp(\eta_l)}x1_l\} = \sum_{l=1}^{100}\{(t_l - y_l)x1_l\} \\
\frac{\partial \log(L)}{\partial a_2} &= \sum_{l=1}^{100}\{t_l x2_l - \frac{\exp(\eta_l)}{1+\exp(\eta_l)}x2_l\} = \sum_{l=1}^{100}\{(t_l - y_l)x2_l\} \\
\frac{\partial \log(L)}{\partial a_3} &= \sum_{l=1}^{100}\{t_l x3_l - \frac{\exp(\eta_l)}{1+\exp(\eta_l)}x3_l\} = \sum_{l=1}^{100}\{(t_l - y_l)x3_l\} \\
\frac{\partial \log(L)}{\partial a_4} &= \sum_{l=1}^{100}\{t_l x4_l - \frac{\exp(\eta_l)}{1+\exp(\eta_l)}x4_l\} = \sum_{l=1}^{100}\{(t_l - y_l)x4_l\} \qquad (43)
\end{aligned}
$$

となります。対数尤度を最大とするパラメータを求めるためには、微分と同じ方向にパラメータを更新すればよいので、パラメータの更新式は、

$$
\begin{aligned}
a_0^{(k+1)} &= a_0^{(k)} + \alpha\sum_{l=1}^{100}(t_l - y_l) \\
a_1^{(k+1)} &= a_1^{(k)} + \alpha\sum_{l=1}^{100}(t_l - y_l)x1_l \\
a_2^{(k+1)} &= a_2^{(k)} + \alpha\sum_{l=1}^{100}(t_l - y_l)x2_l \\
a_3^{(k+1)} &= a_3^{(k)} + \alpha\sum_{l=1}^{100}(t_l - y_l)x3_l \\
a_4^{(k+1)} &= a_4^{(k)} + \alpha\sum_{l=1}^{100}(t_l - y_l)x4_l \qquad (44)
\end{aligned}
$$

のようになります。この更新式は、先の ADALINE の更新式とほとんど同じであることがわかります。

ロジスティック回帰の場合には、出力値は 0 から 1 の間の値を取り、イリス・ベルシコロールである確率の推定値であると解釈できますので、アヤメの花を識別するには、出力値が 0.5 以上ならイリス・ベルシコロールであり、0.5 以下ならイリス・ベルジニカであると判断すれば良いことになります。

先の ADALINE のプログラムを修正して、ロジスティック回帰モデルを用いてアヤメの識別のためのパラメータを学習するプログラムを作ってみると、以下のようになります。

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

```
#include <math.h>

#define frand()   rand()/((double)RAND_MAX)

#define NSAMPLE 100
#define XDIM 4

double logit(double eta)
{

  return(exp(eta)/(1.0+exp(eta)));
}

main() {
  FILE *fp;
  double t[NSAMPLE];
  double x[NSAMPLE][XDIM];
  double a[XDIM+1];
  int   i, j, l;
  double eta;
  double y, err, likelihood;
  double derivatives[XDIM+1];
  double alpha = 0.1; /* Learning Rate */

  /* Open Data File */
  if ((fp = fopen("niris.dat","r")) == NULL) {
    fprintf(stderr,"File Open Fail\n");
    exit(1);
  }

  /* Read Data */
  for (l = 0; l < NSAMPLE; l++) {
    /* Input input vectors */
    for (j = 0; j < XDIM; j++) {
      fscanf(fp,"%lf",&(x[l][j]));
    }
    /* Set teacher signal */
    if (l < 50) t[l] = 1.0; else t[l] = 0.0;
  }
  /* Close Data File */
  fclose(fp);

  /* Print the data */
  for (l = 0; l < NSAMPLE; l++) {
    printf("%3d : %8.2f ", l, t[l]);
    for (j = 0; j < XDIM; j++) {
      printf("%8.2f ", x[l][j]);
    }
    printf("\n");
  }

  /* Initialize the parameters by random number */
  for (j = 0; j < XDIM+1; j++) {
    a[j] = (frand() - 0.5);
  }
```

```
  /* Open output file */
  fp = fopen("likelihood.out","w");

  /* Learning the parameters */
  for (i = 1; i < 100; i++) { /* Learning Loop */

    /* Compute derivatives */
    /* Initialize derivatives */
    for (j = 0; j < XDIM+1; j++) {
      derivatives[j] = 0.0;
    }
    /* update derivatives */
    for (l = 0; l < NSAMPLE; l++) {
      /* prediction */
      eta = a[0];
      for (j = 1; j < XDIM+1; j++) {
eta += a[j] * x[l][j-1];
      }
      y = logit(eta);

      /* error */
      err = t[l] - y;

      /* update derivatives */
      derivatives[0] += err;
      for (j = 1; j < XDIM+1; j++) {
derivatives[j] += err * x[l][j-1];
      }

    }

    /* update parameters */
    for (j = 0; j < XDIM+1; j++) {
      a[j] = a[j] + alpha * derivatives[j];
    }

    /* Compute Log Likelihood */
    likelihood = 0.0;
    for (l = 0; l < NSAMPLE; l++) {
      /* prediction */
      eta = a[0];
      for (j = 1; j < XDIM+1; j++) {
eta += a[j] * x[l][j-1];
      }
      y = logit(eta);

      likelihood += t[l] * log(y) + (1.0 - t[l]) * log(1.0 - y);

    }

    printf("%d : Log Likeihood is %f\n", i, likelihood);
    fprintf(fp, "%f\n", likelihood);

  }
```

```
  fclose(fp);

  /* Print Estmated Parameters */
  printf("\nEstimated Parameters\n");
  for (j = 0; j < XDIM+1; j++) {
    printf("a[%d]=%f, ",j, a[j]);
  }
  printf("\n\n");

  /* Prediction and Log Likelihood */
  for (l = 0; l < NSAMPLE; l++) {
    /* prediction */
    eta = a[0];
    for (j = 1; j < XDIM+1; j++) {
      eta += a[j] * x[l][j-1];
    }
    y = logit(eta);

    if ( y > 0.5) {
      if (l < 50) {
printf("%3d [Class1 : correct] : t = %f, y = %f\n", l, t[l], y);
      } else {
printf("%3d [Class1 : not correct] : t = %f, y = %f\n", l, t[l], y);
      }
    } else {
      if (l >= 50) {
printf("%3d [Class2 : correct] : t = %f, y = %f\n", l, t[l], y);
      } else {
printf("%3d [Class2 : not correct] : t = %f, y = %f\n", l, t[l], y);
      }
    }
  }

}
```

このプログラムの出力結果は、以下のようになります。

```
Estimated Parameters
a[0]=8.946368, a[1]=0.882509, a[2]=1.338263, a[3]=-6.766164, a[4]=-7.298297,

  0 [Class1 : correct] : t = 1.000000, y = 0.993719
  1 [Class1 : correct] : t = 1.000000, y = 0.985676
  2 [Class1 : correct] : t = 1.000000, y = 0.948786
  3 [Class1 : correct] : t = 1.000000, y = 0.987152
  4 [Class1 : correct] : t = 1.000000, y = 0.938278
  5 [Class1 : correct] : t = 1.000000, y = 0.984824
  6 [Class1 : correct] : t = 1.000000, y = 0.937193
  7 [Class1 : correct] : t = 1.000000, y = 0.999931
  8 [Class1 : correct] : t = 1.000000, y = 0.993680
  9 [Class1 : correct] : t = 1.000000, y = 0.990782
 10 [Class1 : correct] : t = 1.000000, y = 0.999542
 11 [Class1 : correct] : t = 1.000000, y = 0.985725
 12 [Class1 : correct] : t = 1.000000, y = 0.999419
 13 [Class1 : correct] : t = 1.000000, y = 0.960009
```

```
14 [Class1 : correct] : t = 1.000000, y = 0.999605
15 [Class1 : correct] : t = 1.000000, y = 0.996275
16 [Class1 : correct] : t = 1.000000, y = 0.940514
17 [Class1 : correct] : t = 1.000000, y = 0.999773
18 [Class1 : correct] : t = 1.000000, y = 0.718517
19 [Class1 : correct] : t = 1.000000, y = 0.999371
20 [Class2 : not correct] : t = 1.000000, y = 0.416174
21 [Class1 : correct] : t = 1.000000, y = 0.998533
22 [Class1 : correct] : t = 1.000000, y = 0.605839
23 [Class1 : correct] : t = 1.000000, y = 0.991769
24 [Class1 : correct] : t = 1.000000, y = 0.997522
25 [Class1 : correct] : t = 1.000000, y = 0.994371
26 [Class1 : correct] : t = 1.000000, y = 0.962073
27 [Class1 : correct] : t = 1.000000, y = 0.522861
28 [Class1 : correct] : t = 1.000000, y = 0.946845
29 [Class1 : correct] : t = 1.000000, y = 0.999966
30 [Class1 : correct] : t = 1.000000, y = 0.999352
31 [Class1 : correct] : t = 1.000000, y = 0.999831
32 [Class1 : correct] : t = 1.000000, y = 0.999283
33 [Class2 : not correct] : t = 1.000000, y = 0.268092
34 [Class1 : correct] : t = 1.000000, y = 0.927374
35 [Class1 : correct] : t = 1.000000, y = 0.969515
36 [Class1 : correct] : t = 1.000000, y = 0.969959
37 [Class1 : correct] : t = 1.000000, y = 0.974863
38 [Class1 : correct] : t = 1.000000, y = 0.998015
39 [Class1 : correct] : t = 1.000000, y = 0.993021
40 [Class1 : correct] : t = 1.000000, y = 0.990881
41 [Class1 : correct] : t = 1.000000, y = 0.979586
42 [Class1 : correct] : t = 1.000000, y = 0.998568
43 [Class1 : correct] : t = 1.000000, y = 0.999916
44 [Class1 : correct] : t = 1.000000, y = 0.992693
45 [Class1 : correct] : t = 1.000000, y = 0.998997
46 [Class1 : correct] : t = 1.000000, y = 0.996440
47 [Class1 : correct] : t = 1.000000, y = 0.996933
48 [Class1 : correct] : t = 1.000000, y = 0.999966
49 [Class1 : correct] : t = 1.000000, y = 0.996702
50 [Class2 : correct] : t = 0.000000, y = 0.000018
51 [Class2 : correct] : t = 0.000000, y = 0.016302
52 [Class2 : correct] : t = 0.000000, y = 0.001129
53 [Class2 : correct] : t = 0.000000, y = 0.019609
54 [Class2 : correct] : t = 0.000000, y = 0.000335
55 [Class2 : correct] : t = 0.000000, y = 0.000131
56 [Class2 : correct] : t = 0.000000, y = 0.190189
57 [Class2 : correct] : t = 0.000000, y = 0.003928
58 [Class2 : correct] : t = 0.000000, y = 0.004127
59 [Class2 : correct] : t = 0.000000, y = 0.000079
60 [Class2 : correct] : t = 0.000000, y = 0.058820
61 [Class2 : correct] : t = 0.000000, y = 0.014368
62 [Class2 : correct] : t = 0.000000, y = 0.003806
63 [Class2 : correct] : t = 0.000000, y = 0.004500
64 [Class2 : correct] : t = 0.000000, y = 0.000185
65 [Class2 : correct] : t = 0.000000, y = 0.001456
66 [Class2 : correct] : t = 0.000000, y = 0.047170
67 [Class2 : correct] : t = 0.000000, y = 0.000445
68 [Class2 : correct] : t = 0.000000, y = 0.000002
```

```
 69 [Class2 : correct] : t = 0.000000, y = 0.231585
 70 [Class2 : correct] : t = 0.000000, y = 0.000534
 71 [Class2 : correct] : t = 0.000000, y = 0.037842
 72 [Class2 : correct] : t = 0.000000, y = 0.000140
 73 [Class2 : correct] : t = 0.000000, y = 0.137514
 74 [Class2 : correct] : t = 0.000000, y = 0.003994
 75 [Class2 : correct] : t = 0.000000, y = 0.027528
 76 [Class2 : correct] : t = 0.000000, y = 0.222651
 77 [Class2 : correct] : t = 0.000000, y = 0.244983
 78 [Class2 : correct] : t = 0.000000, y = 0.000915
 79 [Class2 : correct] : t = 0.000000, y = 0.183885
 80 [Class2 : correct] : t = 0.000000, y = 0.002655
 81 [Class2 : correct] : t = 0.000000, y = 0.011796
 82 [Class2 : correct] : t = 0.000000, y = 0.000350
 83 [Class1 : not correct] : t = 0.000000, y = 0.642195
 84 [Class2 : correct] : t = 0.000000, y = 0.230225
 85 [Class2 : correct] : t = 0.000000, y = 0.000146
 86 [Class2 : correct] : t = 0.000000, y = 0.000293
 87 [Class2 : correct] : t = 0.000000, y = 0.057084
 88 [Class2 : correct] : t = 0.000000, y = 0.299894
 89 [Class2 : correct] : t = 0.000000, y = 0.008427
 90 [Class2 : correct] : t = 0.000000, y = 0.000178
 91 [Class2 : correct] : t = 0.000000, y = 0.003929
 92 [Class2 : correct] : t = 0.000000, y = 0.016302
 93 [Class2 : correct] : t = 0.000000, y = 0.000222
 94 [Class2 : correct] : t = 0.000000, y = 0.000086
 95 [Class2 : correct] : t = 0.000000, y = 0.001591
 96 [Class2 : correct] : t = 0.000000, y = 0.021935
 97 [Class2 : correct] : t = 0.000000, y = 0.022459
 98 [Class2 : correct] : t = 0.000000, y = 0.001482
 99 [Class2 : correct] : t = 0.000000, y = 0.108289
```

今回も、やはり 3 個の識別に失敗していますが、4 個の特徴からほぼアヤメの種類を識別できるようになっていることがわかります。

7 多層パーセプトロン

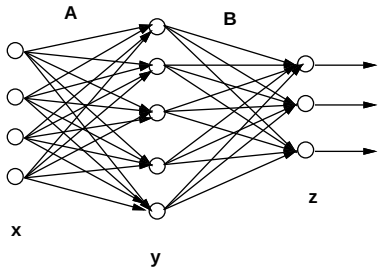


Figure 5: 多層パーセプトロン

多層パーセプトロンは、パーセプトロンを層状に繋ぎ合わせたネットワークです。1980 年代に誤差逆伝搬法と呼ばれる学習アルゴリズムが提案されたことにより注目されるようになりました。それ以来、パターン認識だけでなくさまざまな課題に適用され、その有効性が確かめられています。エアコンなどの家電製品にも「ニューロ」とか「ニューロ・ファジー」とかという宣伝文句が使われたのを記憶している人もいると思います。

例えば、I 個の入力信号の組 $\boldsymbol{x} = (x_1, x_2, \ldots, x_I)^T$ に対して、K 個の出力信号の組 $\boldsymbol{z} = (z_1, \ldots, z_K)^T$ を出力する中間層が 1 層の多層パーセプトロンは、

$$\begin{aligned} \zeta_j &= \sum_{i=1}^{I} a_{ij} x_i + a_{0j} \\ y_j &= S(\zeta_j) \\ z_k &= \sum_{j=1}^{J} b_{jk} y_j + b_{0k} \end{aligned} \tag{45}$$

のような式で表すことができます。ここで、y_j は、j 番目の中間層のニューロンの出力です。また、a_{ij} は、i 番目の入力から中間層の j 番目のニューロンへの結合荷重で、b_{jk} は、中間層の j 番目のニューロンから出力層の k 番目のニューロンへの結合荷重です。図 5 にその概念図を示します。

このような多層パーセプトロンの能力、つまり、どのような関数が表現可能かに関して非常に強力な結果が得られています。それは、中間層が 1 層の多層パーセプトロンによって、任意の連続関数が近似可能であるというものです。もちろん、任意の連続関数を近似するためには中間層のユニットの数を非常に多くする必要があるかもしれません。この結果は、多層パーセプトロンを入出力関係を学習するために使うには、理論的には、中間層が 1 層のみのネットワークで十分であることを示しています。

7.0.1 誤差逆伝搬学習法

多層パーセプトロンは任意の連続関数を近似するのに十分な表現能力をもっているのですが、そうしたネットワークに望みの情報処理をさせるためにはニューロン間の結合荷重を適切なものに設定しなければなりません。ニューロンの数が増えると結合荷重の数も増え、それらをいちいち人手で設定することは非常に難しいので、一般には、利用可能なデータからの学習によって適切な結合加重を求めます。そのためのアルゴリズムとして最も有名なものが誤差逆伝搬学習法です。このアルゴリズムは、これまでにこの講義で話した方法と同様に、最急降下法を用いて最適なパラメータを求めます。

今、N 個の学習用のデータを $\{\boldsymbol{x}_p, \boldsymbol{t}_p | p = 1, \ldots, N\}$ と表すことにします。学習のための評価基準としては、平均 2 乗誤差を最小とする基準

$$\varepsilon^2 = \frac{1}{N} \sum_{p=1}^{N} ||\boldsymbol{t}_p - \boldsymbol{z}_p||^2 = \frac{1}{N} \sum_{p=1}^{N} \varepsilon^2(p) \tag{46}$$

を用いるものとします。最急降下法を適用するために、これまでと同様に平均 2 乗誤差 ε^2 の結合荷重に関する偏微分を計算すると、

$$\begin{aligned} \frac{\partial \varepsilon^2}{\partial a_{ij}} &= \frac{1}{N} \sum_{p=1}^{N} \frac{\partial \varepsilon^2}{\partial a_{ij}} = \frac{1}{N} \sum_{p=1}^{N} -2\gamma_{pj} \nu_{pj} x_{pi} \\ \frac{\partial \varepsilon^2}{\partial b_{jk}} &= \frac{1}{N} \sum_{p=1}^{N} \frac{\partial \varepsilon^2(p)}{\partial b_{jk}} = \frac{1}{N} \sum_{p=1}^{N} -2\delta_{pk} y_{pj} \end{aligned} \tag{47}$$

のようななります。ただし、

$$\begin{aligned} \nu_{pj} &= y_{pj}(1 - y_{pj}) \\ \gamma_{pj} &= \sum_{k=1}^{K} \delta_{pk} b_{jk} \\ \delta_{pk} &= t_{pk} - z_{pk} \end{aligned} \tag{48}$$

です。ここで、a_{0j} および b_{0k} の計算も統一的に表すために、$x_{p0} = 1$ および $y_{p0} = 1$ としています。従って、最急降下法による結合荷重の更新式は

$$\begin{aligned} a_{ij} &\Leftarrow a_{ij} - \alpha \frac{\partial \varepsilon^2}{\partial a_{ij}} \\ b_{jk} &\Leftarrow b_{jk} - \alpha \frac{\partial \varepsilon^2}{\partial b_{jk}} \end{aligned} \tag{49}$$

のようになります。ここで、α は学習率と呼ばれる正のパラメータです。

この学習アルゴリズムは、教師信号とネットワークの出力との誤差 δ を結合荷重 b_{jk} を通して逆向きに伝搬して γ を計算していると解釈できるので誤差逆伝搬法という名前が付けれらています。

すでに何度か書きましたが、最急降下法を用いた学習法では、学習率をどのように決めるかによってアルゴリズムの収束の速さが影響を受けます。そのため学習率を適切な値に設定することが重要となりますが、ニューラルネットワークの研究では、それを自動的に設定するための方法もいくつか提案されています。また、学習の高速化に関しても多くの方法が提案されています。例えば、Quick Prop という方法では、多くのヒューリスティックを組み合わせて学習を高速化しています。その他、ニュートン法的な方法を用いて高速化するアルゴリズムもいくつか提案されています。