認識行動システムの基礎　演習問題１

さいたまおとこ

2005 年 1 月 26 日提出分

レポートの出てしまった経緯

ご存知の通り、今年は応用物理部門では教員の配置替えがありました。前半の並木先生、後半の篠田先生ともにこの科目を教えるのは初めてです。並木先生にいたっては教壇に立つのも今年が初めてです。さらに領域横断的な授業なので他の物理や数学などと比べ過去問がない場合の危険がとても高いだろうと考え、前半の並木先生には提出のない演習を要求しうまくゆきました。後半篠田先生にも「理解の目安が立たないので、解答をあとから配る形の演習をぜひ出してください！」とシケ対の必要から頼んだところ見事に裏目に出ました。

提出は任意で単位に不可欠なものではなく、期末試験に自信があれば出さなくてもいいとのことです。問題は 6 番以外はすぐに終えられます。しかし英 I の前日の水曜、皆非常に忙しいころが締め切りとなっておりその時期に負担を作ってしまったのは申し訳ないです。メモ程度のレポプリを作らせてもらいますので、出すつもりの方は早めに終わらせてしまいましょう。急いだので間違いが多々あると思いますがご了承ください。

解答にあたり 1 つめの資料の中のコマの番号を示します。資料は篠田先生のページにあります。

(http://www.alab.t.u-tokyo.ac.jp/~shino/ninshiki/index.html)

1. パターン認識とその難しさ
   パターン認識 $\cdots$　3〜4 コマ目のような n 次元ベクトルのクラス (カテゴリ) 分け
   難しさ $\ldots$　5 コマ目の 8 桁の数字が訴えている事を汲み取りましょう

2. 線形識別 (分離) 可能とは
   7 コマ目、データ空間 $\boldsymbol{R}^n$ を $\boldsymbol{R}^{n-1}$ で分割

3. 汎化とは
   学習に用いない（未知の（新しい））パターンへの適応能力。
   20 コマ目のように領域の指定をきめ細かくしていくことで実現

4. $A(0,2),(1,1)$ と $B(3,0),(3,3)$ の識別面を (I) ユークリッド距離で (II) マージン最大条件で決めよ。
   (I) クラス内で平均を取って、$(\frac{1}{2},\frac{3}{2})$ と $(3,\frac{3}{2})$ からの等距離面は (x,y) でみると $x=\frac{7}{4}$
   (II) 識別面が $\boldsymbol{w}\cdot\boldsymbol{x}+b=0$ のとき、マージンは $\gamma_i = y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)$ で識別面から $\boldsymbol{x}_i$ までの距離をあらわすが、特に $\min_i \gamma_i$ のことを γ と書いて面に一番近いデータまでの距離のことをいう。
   クラス A の要素とクラス B の要素の最小距離は $\boldsymbol{a}_2$ と $\boldsymbol{b}_1$ の間で、この 2 つを結ぶ線分の垂直 2 等分は $(2,\frac{1}{2})$ を通り $y=2x-\frac{7}{2}$　(正しく判別できていない。)

5. 最急降下法とは
   22 コマ目にしたがって誤差 E を決め、21 コマ目にしたがってこれを減らす方向へ重み w_n を変化させ

る。資料の「p49」下段参照
BP 法に拡張すると慣性項がつく。

6. 階層型ニューラルネットワークの計算
解答案
(a) 空欄で提出 (推奨)
２１コマ目のシグモイド関数 $y_i = \frac{1}{1+\exp(-x_i+\theta_i)}$ を使うと $\frac{1}{1+\exp(\frac{1}{1+e})}$ などを手計算で扱う事に。
別にいいのですがその時間を英 I 等に回したいかもしれません。
飛ばしても出すだけで一応の出席点にはなると思います。
(b) 入出力を線形とする
入力値 x をそのまま出力 y にします。少し楽になりそうです。説明のみ書きます。
(c) 数値計算する
時間の都合から扱うのはこの場合だけにします。いずれの場合も η が与えられていないので一応 E の偏微分までででよいはずです。

解答例
(a)

(b) 左から I 層、J 層、H 層として、各々の層の素子への入力値を x_h, x_i, x_j, 出力値を y_h, y_i, y_j、層間を結ぶ重みを w_{ih}, w_{ji} とかく。
各素子の出力値と次の層の素子への入力値の関係は、重みつき和

$$x_i = \sum_h y_h w_{ih}, \quad x_j = \sum_i y_i w_{ji}$$

とし、簡単のため入出力関係を線形とする。各素子内での入出力値の関係は

$$y_i = x_i, \quad y_j = x_j \text{(初回のみなのでバイアス無視)}$$

このとき計算によって $\left(\frac{\partial y_i}{\partial x_i}\right) = 1$。
添字 c は c 番目の学習入力パターンを指すとして、理想出力 d と J 層からの実際の出力の誤差 E を

$$E_c = \frac{1}{2}\sum_j (y_{j,c} - d_{j,c})^2 \quad \text{(n:回数)}$$

$$E = \sum_c E_c$$

と決める。
各重み w を BP 法に従って更新するが、最初の変化では慣性項は考えないから

$$w_{n+1} = w_n - \eta \frac{\partial E}{\partial w(n)} \quad (\eta \text{ は適当な係数})$$

で w_{n+1} を決める。w_{ji} は資料「p54」と同じ考え方により、

$$\frac{\partial E}{\partial w_{ji}(n)} = \sum_c \frac{\partial E_c(n)}{\partial w_{ji}(n)} = \sum_c \frac{\partial E_c(n)}{\partial y_{j,c}} \frac{\partial y_{j,c}}{\partial x_{j,c}} \frac{\partial x_{j,c}}{\partial w_{ji}(n)}$$

$$= \sum_c (y_{j,c} - d_{j,c}) y_{i,c}$$

w_{ih} については、

$$\frac{\partial E}{\partial w_{ih}(n)} = \sum_c \frac{\partial E_c(n)}{\partial y_{i,c}} \frac{\partial y_{i,c}}{\partial x_{i,c}} \frac{\partial x_{i,c}}{\partial w_{ih}(n)}$$

$$= \sum_c \sum_j (y_{j,c} - d_{j,c}) w_{ji} y_{h,c}$$

となる。これを使って値を求める。
(c) 使用したとても冗長なプログラム (C) を貼っておきます。参考になるかもしれません。継ぎ足しで作っていったので体裁に手がまわっていません。

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

double f(double x); //素子内の入出力特性

//ニューロン
struct neuron{
 double value;
 double w[3];
 double dw[3];//10 行目
};

int main(void){
 int i,c,sk,j,tj,h;
 double idt[8]={1,0,2,3,0,1,3,0},ddt[8]={1,0,1,0,0,1,0,1},
  odt[8],mdt[8],tdelc,jsum;
 struct neuron n[6]; //生成

 //初期化
 for(i=0;i<6;i++){ //20 行目
  n[i].w[1]=0;
  n[i].w[2]=0;
 }
 n[0].w[1]=1;
 n[2].w[1]=1;
 for(i=0;i<6;i++){
  printf( "(%d)w1:%f w2:%f\n", i+1,n[i].w[1],n[i].w[2] );
 }
```

```
//データ入力 //30 行目
for(c=0;c<4;c++){ //c(パターン番号) ごと
 n[0].value = idt[2*c];   //入力素子に入力
 n[1].value = idt[2*c + 1];
 for(sk=1;sk<3;sk++)
 for(j=0;j<2;j++){  //sk(層番号) ごとに重みつき和
  n[2*sk+j].value = n[2*sk-2].w[j+1] * f( n[2*sk-2].value )
   + n[2*sk-1].w[j+1] * f( n[2*sk-1].value );
 }
 mdt[2*c]=n[2].value;mdt[2*c+1]=n[3].value;//層ごとの出力を保存
 odt[2*c]=n[4].value;odt[2*c+1]=n[5].value;  //40 行目
 printf( "out(%d):%f %f\n", c+1,odt[2*c],odt[2*c+1] );
}

//ih 間で最急降下法による重みの更新量を求める
for(i=0;i<2;i++)
for(h=0;h<2;h++){
 tdelc=0;
 for(c=0;c<4;c++){
  jsum=0;
  for(tj=0;tj<2;tj++){ //50 行目
   jsum+=(odt[2*c+tj]-ddt[2*c+tj])*odt[2*c+tj]*(1-odt[2*c+tj])
    *n[2+tj].w[1+tj];
  }
  tdelc +=
  jsum * mdt[2*c+i]*(1-mdt[2*c+i])* idt[2*c+h];
 }
 n[i].dw[j]+=tdelc;
 printf("w%d:dif=%f\n",1+2*i+h,tdelc);
}
//60 行目
 //ji 間
 for(i=0;i<2;i++)
 for(j=0;j<2;j++){
  tdelc=0;
  for(c=0;c<4;c++){
   tdelc += (odt[2*c+j] - ddt[2*c+j]) *odt[2*c+j]*(1-odt[2*c+j])*mdt[2*c+i];
  }
  n[2+i].dw[j]+=tdelc;
  printf("w%d:dif=%f\n",5+2*i+j,tdelc); //70 行目
```

```
 }
 return 0;
}

double f(double x){
 return (1/(1+exp(-x))); //シグモイド関数
}
```

結果：

```
w1:dif=-0.007115
w2:dif=0.017451
w3:dif=0.000000
w4:dif=0.000000
w5:dif=0.105649
w6:dif=0.000000
w7:dif=0.000000
w8:dif=0.000000
```

(2) のように線形のネットワークにしたときは

```
w1:dif=11.000000
w2:dif=3.000000
w3:dif=11.000000
w4:dif=3.000000
w5:dif=11.000000
w6:dif=-3.000000
w7:dif=0.000000
w8:dif=0.000000
```

※くれぐれも値をまる写ししないでください。
プログラム自体が怪しいので自分は提出時はとりあえず努力した跡を伝えるつもりです。

参考：前年度資料
http://www.k2.t.u-tokyo.ac.jp/~koichi/pattern.html
※シケプリは２月中に何らかのおまけつきで発行する予定です。