ロボットインテリジェンス
レポート課題

機械情報工学科3年
学生証番号:140290
溝花弘登

協力者:林泉(140284)

1. 概要

レポート課題 A を選択した。このレポートでは、三層フィードフォワード型のニューラルネットを生成し、バックプロパゲーションにより学習を行った。ニューラルネットには文字を表す画像を入力し、その認識を行わせた。学習の各パラメータを変化させてシミュレートを行い、ニューラルネットの最適化について考察する。

2. 前提条件

・読み込む画像は A～F の文字を表す 20px*20px の画像を各文字について 5 枚ずつ用意した。
・画像はモノクロの濃淡画像で、各画素値は 0～1 の間の値で与えられる。
・プログラムは c 言語を用いて記述した。
・ニューラルネットのニューロン数は各層ごとに、入力層と中間層は画像の画素数に等しい400、出力層は判定する文字の種類に等しい 6 とした。各層間の初期結合荷重は乱数を用いて-0.5～0.5 の範囲で決定され、中間層と出力層間の結合荷重は学習によって更新される。
・学習にはバックプロパゲーションを用い、慣性項についても考慮する。
・性能評価には教師データとの二乗平均平方根を用いる。

3. シミュレート実験

作成したプログラムはレポートの末尾に貼付する。

(1) 認識画像

以下のような画像 30 種類を認識して 6 種類の文字に判定する。

A					
B					
C					
D					
E					
F					

(2)パラメータ

・ノイズ d : 画像読み込みの際のノイズ。d%の割合で画素値が乱数値に書き換えられる。
・ゲイン alpha : 各ニューロンの出力値を計算する際のシグモイド関数のゲイン。今回は常に 1 にした。
・学習係数 eta
・慣性係数 mu
・学習打ち切り回数 num_learn : 画像の種類(30)×num_learn 回学習を行う。

(3)結果(d=0, eta=0.05, mu=0.150 の場合)

[1]
num_learn = 10
A1.jpg y = 0.442 0.057 0.050 0.092 0.079 0.000 err = -0.558 0.057 0.050 0.092 0.079 0.000 correct
B1.jpg y = 0.017 0.079 0.044 0.046 0.045 0.000 err = 0.017 -0.921 0.044 0.046 0.045 0.000 correct
C1.jpg y = 0.049 0.036 0.045 0.175 0.129 0.000 err = 0.049 0.036 -0.955 0.175 0.129 0.000 error
D1.jpg y = 0.048 0.021 0.147 0.739 0.108 0.000 err = 0.048 0.021 0.147 -0.261 0.108 0.000 correct
E1.jpg y = 0.063 0.251 0.063 0.030 0.426 0.000 err = 0.063 0.251 0.063 0.030 -0.574 0.000 correct
F1.jpg y = 0.095 0.093 0.019 0.038 0.252 0.000 err = 0.095 0.093 0.019 0.038 0.252 -1.000 error
A2.jpg y = 0.458 0.056 0.091 0.088 0.052 0.000 err = -0.542 0.056 0.091 0.088 0.052 0.000 correct
B2.jpg y = 0.051 0.051 0.091 0.148 0.157 0.000 err = 0.051 -0.949 0.091 0.148 0.157 0.000 error
C2.jpg y = 0.037 0.033 0.108 0.164 0.077 0.000 err = 0.037 0.033 -0.892 0.164 0.077 0.000 error
D2.jpg y = 0.020 0.056 0.062 0.198 0.083 0.000 err = 0.020 0.056 0.062 -0.802 0.083 0.000 correct
E2.jpg y = 0.044 0.105 0.026 0.045 0.184 0.000 err = 0.044 0.105 0.026 0.045 -0.816 0.000 correct
F2.jpg y = 0.037 0.134 0.033 0.073 0.064 0.000 err = 0.037 0.134 0.033 0.073 0.064 -1.000 error
A3.jpg y = 0.738 0.071 0.038 0.034 0.056 0.000 err = -0.262 0.071 0.038 0.034 0.056 0.000 correct
B3.jpg y = 0.037 0.118 0.132 0.105 0.066 0.000 err = 0.037 -0.882 0.132 0.105 0.066 0.000 error
C3.jpg y = 0.041 0.027 0.028 0.311 0.112 0.000 err = 0.041 0.027 -0.972 0.311 0.112 0.000 error
D3.jpg y = 0.027 0.009 0.187 0.833 0.081 0.000 err = 0.027 0.009 0.187 -0.167 0.081 0.000 correct
E3.jpg y = 0.064 0.004 0.070 0.203 0.147 0.000 err = 0.064 0.004 0.070 0.203 -0.853 0.000 error
F3.jpg y = 0.027 0.124 0.018 0.084 0.046 0.000 err = 0.027 0.124 0.018 0.084 0.046 -1.000 error
A4.jpg y = 0.427 0.076 0.065 0.036 0.059 0.000 err = -0.573 0.076 0.065 0.036 0.059 0.000 correct
B4.jpg y = 0.047 0.077 0.078 0.110 0.113 0.000 err = 0.047 -0.923 0.078 0.110 0.113 0.000 error
C4.jpg y = 0.042 0.031 0.095 0.125 0.099 0.000 err = 0.042 0.031 -0.905 0.125 0.099 0.000 error
D4.jpg y = 0.032 0.008 0.206 0.831 0.087 0.000 err = 0.032 0.008 0.206 -0.169 0.087 0.000 correct
E4.jpg y = 0.049 0.222 0.019 0.119 0.587 0.000 err = 0.049 0.222 0.019 0.119 -0.413 0.000 correct
F4.jpg y = 0.021 0.073 0.024 0.140 0.116 0.000 err = 0.021 0.073 0.024 0.140 0.116 -1.000 error
A5.jpg y = 0.677 0.071 0.127 0.040 0.062 0.000 err = -0.323 0.071 0.127 0.040 0.062 0.000 correct
B5.jpg y = 0.154 0.070 0.097 0.069 0.317 0.000 err = 0.154 -0.930 0.097 0.069 0.317 0.000 error
C5.jpg y = 0.033 0.037 0.045 0.147 0.270 0.000 err = 0.033 0.037 -0.955 0.147 0.270 0.000 error
D5.jpg y = 0.100 0.012 0.099 0.754 0.191 0.000 err = 0.100 0.012 0.099 -0.246 0.191 0.000 correct
E5.jpg y = 0.143 0.126 0.024 0.039 0.724 0.000 err = 0.143 0.126 0.024 0.039 -0.276 0.000 correct
F5.jpg y = 0.031 0.093 0.018 0.156 0.152 0.000 err = 0.031 0.093 0.018 0.156 0.152 -1.000 error
Erms = 0.002415
correct answers rate = 15 / 30

[2]※以下はパラメータと誤差評価関数の値、正解判定数のみ記す。
num_learn = 30
Erms = 0.000447
correct answers rate = 30 / 30

[3]
num_learn = 50
Erms = 0.000164
correct answers rate = 30 / 30
※Erms は誤差評価関数

[4]
num_learn = 200
Erms = 0.000016
correct answers rate = 30 / 30

(4)結果(num_learn=30, eta=0.05, mu=0.150 の場合)

[1]
d = 5
Erms = 0.000470
correct answers rate = 30 / 30

[2]
d = 15
Erms = 0.000753
correct answers rate = 15 / 30

[3]
d = 20
Erms = 0.000873
correct answers rate = 12 / 30

(5)結果(ノイズ耐性の検証)

[1]
d = 25
eta = 0.050
mu = 0.150
num_learn = 60
Erms = 0.000359
correct answers rate = 23 / 30

[2]
d = 25
eta = 0.050
mu = 0.150
num_learn = 100
Erms = 0.000129
correct answers rate = 29 / 30

[3]
d = 25
eta = 0.050
mu = 0.300
num_learn = 30
Erms = 0.000721
correct answers rate = 17 / 30

[4]
d = 25
eta = 0.050
mu = 0.500
num_learn = 30
Erms = 0.000684
correct answers rate = 21 / 30

4. 考察

ノイズがない場合のシミュレートでは、学習打ち切り回数を大きくするにしたがって精度が向上した。正しい画

像認識を行うためには、num_learn=30 で十分である。num_learn = 200 まで実験したが、過学習は起きなかった。

ノイズがある場合では、ノイズが増えるにしたがって性能が低下し、d=25 では半分以上の判定が間違いとなった。

ノイズ耐性の要因としては、学習打ち切り回数と慣性係数が考えられる。打ち切り回数が多くなるほど、ノイズの効果が打ち消され、正しいデータによる結果に近づくと考えられる。また、慣性係数が大きくなれば、概ね正しいデータの効果が次回の学習にも持ち越される。これにより、次回の学習でのノイズの効果を軽減できると考えられる。3.(5)の結果はこの考察を裏付けるものである。

5. プログラム

```
/*================================================
 This program makes three layer feedforward neural network model
and enhance it by BP Algorithm.
neural network is consist of follow variables.
the value of neurons in input layer             : vector x1
the value of neurons in middle layer            : vector x2
the value of neurons in  output layer           : vector y
synaptic weight between input and middle layer  : vector w1
synaptic weight between middle and output layer : vector w2
================================================*/

/*-- include --*/
#include <stdio.h>
#include <cv.h>
#include <highgui.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

/*-- define --*/
#define N 400   //image size
#define n_type 6  //the number of image type
#define d 25     //noise in input images
#define alpha 1.0 //gain
#define eta 0.05 //learning rate
#define mu 0.5 //inertia factor
#define num_image 5 //the number of images of each letter
#define num_learn 30 //the number of learning

/*-- grobal variables --*/
double w1[N][N];
double w2[N][n_type];
double delta_w2[N][n_type];
double Erms = 0.0;


/*------------------------------------------------
```

```
read_img function
 - read 20x20 input image "name" and convert into gray image
  then put image data into q
-----------------------------------------------------*/
int read_img(double *q,char *name){
 IplImage *img;
 IplImage *gray;
 int x,y;
 unsigned int p;
 int noise;

 img = cvLoadImage(name, CV_LOAD_IMAGE_COLOR);
 if(img == NULL){
  fprintf(stderr, "couldn't read image!");
  return 1;
 }

 gray = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
 cvCvtColor(img, gray, CV_BGR2GRAY);

 for(y = 0; y < gray->height; y++){
  for(x = 0; x < gray->width; x++){
   p = (int)(unsigned char)gray->imageData[gray->widthStep*y + x];
   noise = rand() % 100;
   if(noise < d){
         p = ((rand() % 255) + 1);
   }
   q[y*gray->width + x] = p/255.0;
  }
 }
 cvReleaseImage(&img);
 cvReleaseImage(&gray);
 return 0;

}


/*-----------------------------------------------------
init_w function
 - Initialize w1 and w2 by random score (-0.5 ~ 0.5)
-----------------------------------------------------*/
void init_w(){
 int i,j,k;
 for (i = 0; i < N; i++){
  for (j = 0; j < N; j++){
   w1[i][j] = ((double)rand() / RAND_MAX) - 0.5;
  }
  for(k = 0; k < n_type; k++){
   w2[i][k] = ((double)rand() / RAND_MAX) - 0.5;
```

```
      delta_w2[i][k] = 0.0;
    }
  }
}


/*-----------------------------------------------
calc_y function
 - calculate x1, x2, y
-----------------------------------------------*/
void calc_y(double *q, double *x1, double *x2, double *y){
  int i,j;

  //input layer
  for (i = 0; i < N; i++){
    x1[i] = q[i];
  }

  //middle layer
  for (i = 0; i < N; i++){
    for (j = 0; j < N; j++){
      if (j == 0){
            x2[i] = x1[j] * w1[j][i];
      } else {
            x2[i] += x1[j] * w1[j][i];
      }
    }
    x2[i] = 1.0 / (1.0 + exp(-alpha * x2[i]));
  }

  //output layer
  for (i = 0; i < n_type; i++){
    for (j = 0; j < N; j++){
      if(j == 0){
            y[i] = x2[j] * w2[j][i];
      } else {
            y[i] += x2[j] * w2[j][i];
      }
    }
    y[i] = 1.0 / (1.0 + exp(-alpha * y[i]));
  }

  printf("y    = ");
  for (i = 0; i < n_type; i++){
    printf("%4.3lf ", y[i]);
  }
  printf("¥n");
}
```

```
/*----------------------------------------------------
calc_err function
 - calculate error
----------------------------------------------------*/
void calc_err(double *y, double *y_ans, double *err){
  int i;
  for (i = 0; i < n_type; i++){
    err[i] = y[i] - y_ans[i];
    Erms += err[i] * err[i];
  }

  printf("err = ");
  for (i = 0; i < n_type; i++){
    printf("%4.3lf ", err[i]);
  }
  printf("¥n");
}


/*----------------------------------------------------
calc_new_w function
 - update w1 and w2
----------------------------------------------------*/
void calc_new_w(double *x1, double *x2, double *y, double *err){
  int i,j;
  double sigmaout[n_type];
  double deltaout[n_type];

  //sigmaout
  for (i = 0; i < n_type; i++){
    sigmaout[i] = alpha * y[i] * (1.0 - y[i]);
  }
  //deltaout
  for (i = 0; i < n_type; i++){
    deltaout[i] = err[i] * sigmaout[i];
  }

  //w2
  for (j = 0; j < n_type; j++){
    for (i = 0; i < N; i++){
      w2[i][j] -= eta * x2[i] * deltaout[j] + mu * delta_w2[i][j];
delta_w2[i][j] = eta * x2[i] * deltaout[j] + mu * delta_w2[i][j];

    }
  }
}
```

```
/*---------------------------------------------------
BPalgorithm function
 - operate BPalgorithm
---------------------------------------------------*/
void BPalgorithm(char* mark, int n){
  double q[N];
  char name[80];
  double x1[N];
  double x2[N];
  int r;
  double y[n_type];
  double y_ans[n_type] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
  double err[n_type];

  switch(*mark){
  case 'A':
    y_ans[0] = 1.0;
    break;
  case 'B':
    y_ans[1] = 1.0;
    break;
  case 'C':
    y_ans[2] = 1.0;
    break;
  case 'D':
    y_ans[3] = 1.0;
    break;
  case 'E':
    y_ans[4] = 1.0;
    break;
  case 'F':
    y_ans[5] = 1.0;
    break;
  default:
    printf("no answer!¥n");
    break;
  }

  sprintf(name,"./images/is%s/%s%d.jpg",mark, mark, n);
  if((r = read_img(q,name)) == 0){
    calc_y(q, x1, x2, y);
    calc_err(y, y_ans, err);
    calc_new_w(x1, x2, y, err);
  }
}


/*---------------------------------------------------
max_y function
```

```
 - return the address in which the maximum value in y is.
----------------------------------------------------*/
int max_y(double *y){
  int i;
  double buf = 0.0;
  int max = 0;
  for (i = 0; i < n_type; i++){
    if(y[i] > buf){
      buf = y[i];
      max = i;
    }
  }
  return max;
}

/*----------------------------------------------------
recognize_test function
 - recognize a image of letter and report the result.
----------------------------------------------------*/
int recognize_test(char* mark, int n, FILE *fp){
  double q[N];
  char name[80];
  double x1[N];
  double x2[N];
  int r,i;
  double y[n_type];
  double y_ans[n_type] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
  double err[n_type];
  int ans;

  switch(*mark){
  case 'A':
    y_ans[0] = 1.0;
    ans = 0;
    break;
  case 'B':
    y_ans[1] = 1.0;
    ans = 1;
    break;
  case 'C':
    y_ans[2] = 1.0;
    ans = 2;
    break;
  case 'D':
    y_ans[3] = 1.0;
    ans = 3;
    break;
  case 'E':
    y_ans[4] = 1.0;
```

```
    ans = 4;
    break;
  case 'F':
    y_ans[5] = 1.0;
    ans = 5;
    break;
  default:
    printf("no answer!¥n");
    ans = 6;
    break;
  }

  sprintf(name,"./images/is%s/%s%d.jpg",mark, mark, n);
  if((r = read_img(q,name)) == 0){
    calc_y(q, x1, x2, y);
    calc_err(y, y_ans, err);
    fprintf(fp, "%s%d.jpg  y = ", mark, n);
    for (i = 0; i < n_type; i++){
      fprintf(fp, "%4.3lf ", y[i]);
    }
    fprintf(fp, " err = ");
    for (i = 0; i < n_type; i++){
      fprintf(fp, "%4.3lf ", err[i]);
    }

    if(ans == max_y(y)){
      fprintf(fp, " correct¥n");
      return 1;
    } else {
      fprintf(fp, "error¥n");
      return 0;
    }
  }
}


/*-------------------------------------------------
test_report function
 - test the efficiency of neural network and report the result in a file.
-------------------------------------------------*/
void test_report(){
  int n, correct;
  FILE *fp;
  char name[80];

  sprintf(name, "./result/d-%d_eta-%4.3lf_mu-%4.3lf_learn-%d.txt", d, eta, mu, num_learn);
  if ((fp = fopen(name, "w")) == NULL){
    printf("result file open error!¥n");
    exit(0);
```

```
  }
  fprintf(fp, "d = %d¥neta = %4.3lf¥nmu = %4.3lf¥nnum_learn = %d¥n",d,eta,mu,num_learn);

  correct = 0;
  Erms = 0;
   for (n = 1; n <= num_image; n++){
     printf("%d image¥n",n);
     correct += recognize_test("A",n,fp);
     correct += recognize_test("B",n,fp);
     correct += recognize_test("C",n,fp);
     correct += recognize_test("D",n,fp);
     correct += recognize_test("E",n,fp);
     correct += recognize_test("F",n,fp);
   }
   Erms = sqrt(Erms)/(n_type * num_image * num_learn * n_type);
   fprintf(fp, "Erms = %lf¥n", Erms);
   fprintf(fp, "correct answers rate = %d / %d¥n", correct, n_type * num_image);
  fclose(fp);

   printf("correct answers rate = %d / %d¥n", correct, n_type * num_image);
   printf("report finished !¥n");

}


void main (){
  int l,n;
  double Erms_;
  srand((unsigned)time(NULL));

  //initialize
  init_w();

  //learning
  for (l = 0; l < num_learn; l++){
    printf("¥n%d learng¥n",(l+1));
    for (n = 1; n <= num_image; n++){
      printf("%d image¥n",n);
      BPalgorithm("A",n);
      BPalgorithm("B",n);
      BPalgorithm("C",n);
      BPalgorithm("D",n);
      BPalgorithm("E",n);
      BPalgorithm("F",n);
    }
  }
  //test and report
  test_report();
}
```