知能システム前期末レポート

15422 篠田拓樹

2019 年 7 月 28 日

1　XOR を学習するニューラルネットワーク

XOR を学習するニューラルネットワークを作成した．リスト 1 に示すプログラムを用いて実行を行った．

XOR の各入力に対して学習がうまくいっていることを確認した．

また，拡張として行列演算によって前向き，後ろ向きの計算を行うリスト 2 のプログラムも作成した．なお，リスト 1 と同じ重みを用いて学習を行い，結果が同じになることを確認している．同様に学習がうまくいっていることを確認した．

学習過程の誤差の変化を図 1 に示す．

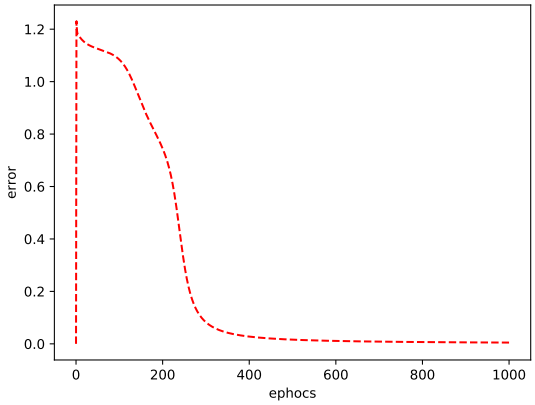

図 1　XOR 学習時の誤差変化

2　魚を識別するニューラルネットワーク

3 プログラムリスト

リスト 1 sampleBP.c

```c
/*
 * NeuralNetwork For XOR
 *
 * Input layer:  2
 * Hidden layer: 2
 * Output layer: 1
 */

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

#define EPSILON 4.0
#define ETA 0.1
#define TIMES 1000
#define INIT_WEIGHT 0.3

double randNum(void)
{
  return ((double)rand()/RAND_MAX-0.5)*2.0*
   INIT_WEIGHT;
}

double sigmoid(double x)
{
  return 1/(1+exp(-1*EPSILON*x));
}

int main(void)
{
  double data[4][3] = {
    {0.0, 0.0, 0.0},
    {0.0, 1.0, 1.0},
    {1.0, 0.0, 1.0},
    {1.0, 1.0, 0.0}
  };
  double wbd, wbe, wcd, wce, wab, wac;
  double offb, offc, offa;
  double outd, oute, outb, outc, outa;
  double xb, xc, xa;
  double deltab, deltac, deltaa;
  int r;
  double error;
  double errorSum;
  int times;
  int seed;
  FILE *fp;

  fp = fopen("error.dat", "w");
  if (fp==NULL) {
    printf("can't open file.\n");
    exit(1);
  }

  //seed = (unsigned int)time(NULL);
  //printf("seed = %d\n", seed);
  seed = 0;
  srand(seed);

  wbd = randNum();
  wbe =  randNum();

  wcd =  randNum();
  wce =  randNum();

  wab =  randNum();
  wac =  randNum();

  offb = randNum();
  offc =  randNum();
  offa =  randNum();

  for(times=0;times<TIMES; times++) {

    errorSum = 0.0;

    for(r=0; r<4; r++) {

      /* ------------ */
      /* Feedforward */
      /* ------------ */

      /* Input layer output */
      outd = data[r][0];
      oute = data[r][1];

      /* Hidden layer output */
      xb = wbd*outd + wbe*oute + offb;
      outb = sigmoid(xb);

      xc = wcd*outd + wce*oute + offc;
      outc = sigmoid(xc);

      /* Output layer output */
      xa = wab*outb + wac*outc + offa;
      outa = sigmoid(xa);

      if(times==TIMES-1) {
        printf("[%d]=%.10f, (%f)\n", r, outa,
          data[r][2]);
      }

      /* ------------------ */
      /* Back Propagation */
      /* ------------------ */
      error = ((outa-data[r][2])*(outa-data[r
       ][2]));
      errorSum += error;

      /*
       *
       * ここに更新式を書く
       *
       * deltaa = ...
       * wab = wab + ...
       *
       */
      deltaa = (outa - data[r][2])*EPSILON
       *(1-outa)*outa;
      deltab = (deltaa * wab)*EPSILON*(1-outb
       )*outb;
      deltac = (deltaa * wac)*EPSILON*(1-outc
       )*outc;

      wab = wab - ETA * deltaa * outb;
      wac = wac - ETA * deltaa * outc;
      offa = offa - ETA * deltaa;

      wbd = wbd - ETA * deltab * outd;
      wbe = wbe - ETA * deltab * oute;
      offb = offb - ETA * deltab;

      wcd = wcd - ETA * deltac * outd;
      wce = wce - ETA * deltac * oute;
      offc = offc - ETA * deltac;
    }
    printf("errorSum = %f\n", errorSum/4.0);
    fprintf(fp, "%f\n", errorSum/4.0);
    //printf(" wab = %f\n wac = %f\n offa=%f\
     n",wab,wac,offa);
    //printf(" wbd = %f\n wbe = %f\n offb=%f\
     n",wbd,wbe,offb);
```

```
    //printf(" wcd = %f\n wce = %f\n offc=%f\n",wcd,wce,offc);
  }
  printf(" wab = %f\n wac = %f\n offa=%f\n", wab,wac,offa);
  printf(" wbd = %f\n wbe = %f\n offb=%f\n", wbd,wbe,offb);
  printf(" wcd = %f\n wce = %f\n offc=%f\n", wcd,wce,offc);


  fclose(fp);

  return 0;
}
```

リスト 2　newral.py

```
import numpy as np
import matplotlib.pyplot as plt

class Newral:
    def __init__(self, in_newrons, mid_newrons, out_newrons):
        self.eps = 4
        self.errors = np.zeros((1,out_newrons))
        self.mid_weight = np.array([[-0.133335,-0.098866],
                                    [0.204113,0.169860],
                                    [-0.063370,0.179064]])
        #np.random.random_sample((in_newrons + 1, mid_newrons))
        self.out_weight = np.array([[0.160938],
                                    [0.246988],
                                    [-0.181469]])
        #np.random.random_sample((mid_newrons + 1, out_newrons))

    """
    シグモイド関数
    ベクトルを計算するために関数を vectorizeで宣言
    """
    def sigmoid(self, vec):
        return np.vectorize(lambda x : 1.0 / (1.0 + np.exp(-1 * self.eps * x)))(vec)

    """
    シグモイド関数の微分関数
    ベクトルを計算するために関数を vectorizeで宣言
    """
    def grad_sigmoid(self, out_vec):
        return np.vectorize(lambda x : self.eps * (1 - x) * x)(out_vec)

    def mean_sqard_error(self, out_vec, t_vec):
        return np.vectorize(lambda x,y : (x - y)**2)(out_vec, t_vec)

    """
    前向きの計算

    行列で書き下すと分かるはず…
    仮想ニューロンを導入しているので(1,x_1, x_2...)^Tと
    の掛け算になる.
    """
    def forward(self, x):
        out_mid = self.sigmoid(np.r_[np.array([1]), x].dot(self.mid_weight))
        out_out = self.sigmoid(np.r_[np.array([1]), out_mid].dot(self.out_weight))

        return (out_mid, out_out)

    """
    学習を行う
    """
    def train(self, x, t, eta, times):
        for i in range(times):
            total_error = 0
            for j, k in zip(x, t):
                total_error += N.BP(j, k, eta)
            self.errors = np.append(self.errors, total_error.reshape(1,-1), axis=0)

    # 1パターン分の学習を行う
    def BP(self, x, t, eta):
        # 前向きの計算
        out_mid, out_out = self.forward(x)
        # 誤差の計算
        error = self.mean_sqard_error(out_out, t)
        # デルタを先に計算する
        out_delta = (out_out - t) * self.grad_sigmoid(out_out)
        mid_delta = self.grad_sigmoid(out_mid) * (self.out_weight[1:,:].dot(out_delta))
        # 更新量を計算
        update_out = np.r_[np.array([1]), out_mid].reshape(-1,1).dot(out_delta.reshape(1,-1))
        update_mid = np.r_[np.array([1]), x].reshape(-1,1).dot(mid_delta.reshape(1,-1))
        # 更新
        self.out_weight -= eta * update_out
        self.mid_weight -= eta * update_mid

        return error

    """
    誤差の変化を確認するグラフ
    """
    def error_graph(self):
        #plt.figure(figsize=(10,20))
        plt.xlabel("ephocs")
        plt.ylabel("error")
        plt.plot(self.errors[:,0],"r—")
        #plt.show()
        plt.savefig("report/img/error1.pdf")

if __name__ == "__main__":
    # 各層のニューロン数
    in_newrons, mid_newrons, out_newrons = (2, 2, 1)
    # 入力
    x = np.array([[0,0],[0,1],[1,0],[1,1]])
    # 出力
    t = np.array([[0],[1],[1],[0]])

    # ニューラルネット
    N = Newral(in_newrons, mid_newrons, out_newrons)
    # 訓練
    N.train(x, t, eta = 0.1, times = 1000)
    # 誤差グラフ
    N.error_graph()
```

```
100
101     # 前向き計算で確認
102     for i,j in zip(x,t):
103         o1,o2 = N.forward(i)
104         print(i,o2,j)
```