神經網에서 活性化函數의 具現에 관한 硏究

박 인 정*

요 약

신경세포의 기능을 등가적으로 표현할 때 활성화 함수는 비선형적이며 이의 구현을 위해 고가의 수치계산 전용 칩이나 고속 컴퓨터를 사용하여왔다. 본논문에서는 이 함수를 근사화하여 구현하기 위해 오차의 종류와 그 크기에 따른 선형함수의 갯수를 조사하고 이를 구현하기 위한 하드웨어 구성 회로를 고찰하였다.

I. 序 論

신경세포를 등가적으로 표현할 때 그 기능은 곱셈, 합산, 변환 및 결합계수 조정으로 나누어진다. 곱셈은 입력신호와 결합계수 사이의 곱으로 주어지고 합산은 이들 곱셈한 값들의 합을 말하며 변환은 합한값이 일정 크기의 범위내에 존재하도록 하는 기능을 가지며 일반적으로 비선형적인 활성화함수에 의해 이루어진다. 변환을 행하는 활성화함수로서는 보통 학습에 효과적인 unipolar Sigmoid함수가[1,2] 일반적이지만 그외에 bipolar Sigmoid함수, 선형변환함수, 선형임계치전달함수, 하드 클리핑 함수, Gaussian 함수, Hyperbolic 함수[3] 등이 있다.

특히 비선형성이 강한 함수의 구현은 쉽지않으며 이를 위해 高價의 수치계산전용칩 (Coprocessor)이나 웍스테이션을 사용하기도 한다. 본논문에서는 이들 함수를 하드웨어로 구현하였을 때 발생하는 문제점을 해결하기 위해 기능의 저하는 거의 없으면서 처리속도를 향상시킬 수 있도록 활성화함수를 몇개의 선형영역으로 분할하여 근사적으로 변환하는 기법을 사용하고 이러한 분할을 기초하여 하드웨어 구현을 고찰하였다.

II. 活性化函數의 근사화

비선형 함수를 y(x), 이의 近似函數를 p(x)라 하면 誤差 e(x)는 다음과 같이 된다.

$$e(x)=y(x)-p(x) \tag{1}$$

이 오차는 x의 모든 값에 따라서 상이하지만 허용되는 오차범위내에서는 몇개의 구간으로 나눌 수 있다고 생각하며, 본논문에서는 먼저 허용오차가 주어졌을 때 그 구간을 자동으로 나누는 방법을 구하고자 한다.

여기서 y(x)를 unipolar Sigmoid 함수라 하고 p(x)는 구간에따라 線形的인 함수라고 하자. y(x)의 그래프는 이득(Gain)을 1, 중심값(Center)을 0으로 했을 때 그림 1과 같다.

*단국대학교 공학대학 전자공학과

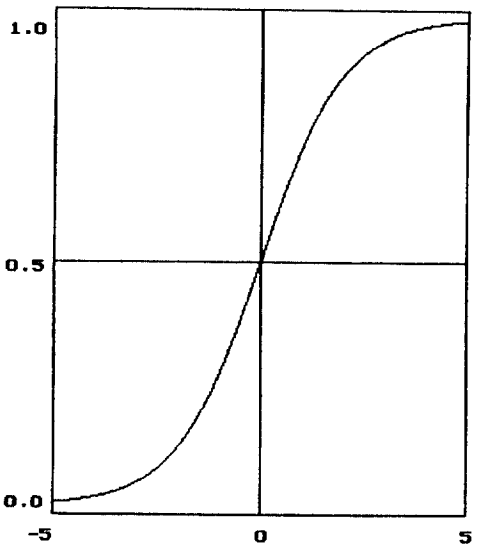

그림 1. unipolar Sigmoid 함수의 그래프.

그림 1에서 보면 이 함수는 x=0, y=0.5인 점에 대해 대칭이며, 그 크기는 최소 0에서 최대 1 사이의 값을 갖으며, 그 표현식은 식(2)로 주어진다.

$$y(x)=1/(1+\exp(-\text{Gain}\cdot(x-\text{Center}))) \tag{2}$$

임의의 구간에서 함수 p(x)는 식(3)으로 주어질 수 있다.

$$p(x)=ax+b$$
$$ax+0.5 \quad \text{만일 } x>0\text{일 때} \tag{3}$$

여기서 a는 기울기, b는 절편이다.

허용오차범위를 d라고 하면 식(4)와 같이 주어진다.

$$|e(x)|<d \text{ 또는 } |y(x)-p(x)|<d \tag{4}$$

여기서 $y(x)-p(x)$ 는 절대오차이며, 허용오차를 상대오차로 하면 이는 식(5)와 같이 주어진다.

$$\text{상대오차}=|y(x)-p(x)|/y(x) \tag{5}$$

III . 근사함수의 특성 고찰

허용오차범위내에 존재하는 선형함수의 갯수는 절대오차와 상대오차를 판별 파라메터로 했을 때 각각 표 1과 표 2에 다음과 같이 주어진다.

표 1과 표 2에 의하면 오차의 값이 0.01인 경우를 제외하곤, 일반적으로 상대오차를 판별 파라메터로 했을 때 선형함수의 갯수가 약간 많다. 오차에 따른 선형함수 그래프를 다음 그림에 보인다.

표 1.

절대오차	선형함수 갯수
0.0001	121
0.0005	121
0.0010	119
0.0050	112
0.0100	120
0.0500	93
0.1000	81

표 2.

상대오차	선형함수 갯수
0.0001	121
0.0005	120
0.0010	120
0.0050	114
0.0100	110
0.0500	97
0.1000	84

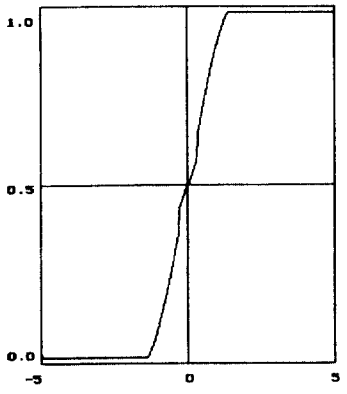

그림 2. 상대오차=0.0005일 때.

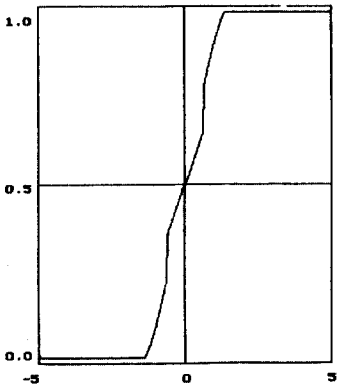

그림 3. 상대오차=0.005일 때.

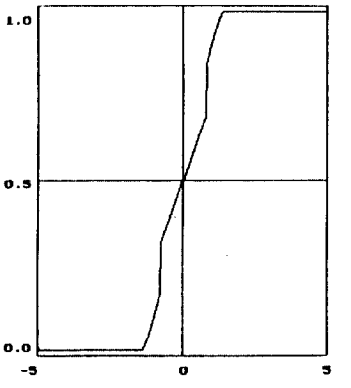

그림 4. 상대오차=0.01일 때.

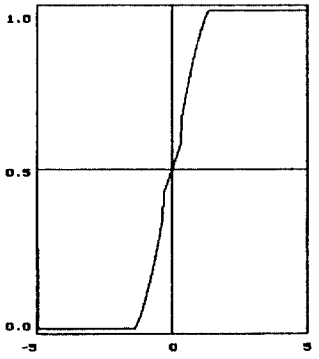

그림 5. 절대오차=0.01일 때.

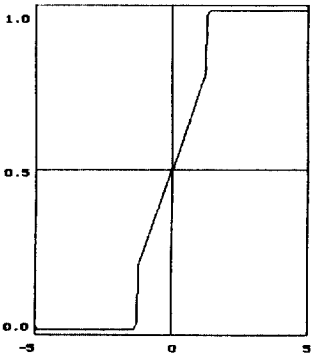

그림 6. 상대오차=0.05일 때.

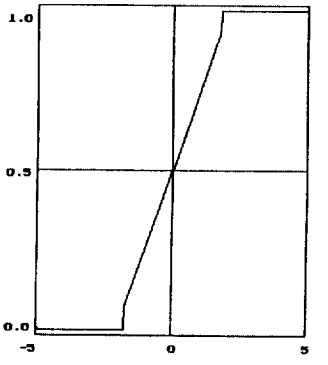

그림 7. 상대오차=0.1일 때.

그림 2에서 그림 7까지를 비교해보면 오차값이 작아질수록 근사화하지 않은 함수와 모양이 유사해짐을 알 수 있다. 특히 그림 4와 그림 5는 각각 상대오차와 절대오차의 크기가 0.01로서 同一하지만 선형근사함수의 모양과 그 갯수가 相異하므로 비교를 위해 제시했다.

부록 1에 근사함수를 구하고 그래프를 그리는 프로그램을 첨부하였다.

IV. 근사함수 구현 回路

표 1과 2에서 보는 바와 같이 선형함수의 갯수가 많으므로 하드웨어화했을 때 계산식을 변경시켜주는 회로가 필요하다. 그 회로는 그림 8과 같이 주어진다.

또다른 방법은 룩업 테이블에 의해 구현하는 것이다. 이를 위해 등간격으로 함수값을 구한 후, 2진수로 변환시켜 ROM에 미리 저장해 놓는 것이다.

부록 2에 룩업 테이블 데이타 생성을 위한 프로그램을 첨부한다.

IV. 結　論

신경세포의 기능을 등가적으로 표현할 때 활성화 함수는 비선형적이며 이의 구현을 위해 고가의 수치계산 전용 칩이나 고속 컴퓨터를 사용하여 왔다. 본논문에서는 이 함수를 몇 개의 구간에 따른 선형함수로 근사화할 때 적용하는 파라메터로서 원래함수값에 비교되는 선형함수의 값을 정하기 위해 오차를 도입하고, 오차의 종류로서 절대오차와 상대오차를 사용하였으며, 이들 오차의 크기에 따른 선형함수의 갯수를 조사하였다.

이 조사에 의하면 오차의 값이 0.01인 경우를 제외하곤, 일반적으로 상대오차를 판별 파라메터로 했을 때 선형함수의 갯수가 약간 많음을 알 수 있었다.

또한 이들 선형함수의 그림을 비교해보면 오차값이 작아질수록 근사화하지 않은 함수와

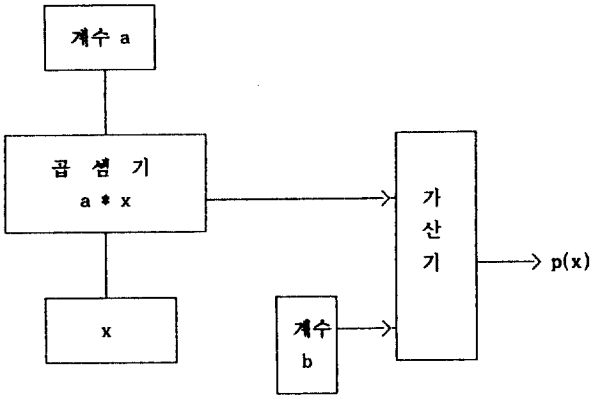

그림 8. 곱셈기와 가산기에 의한 구현회로.

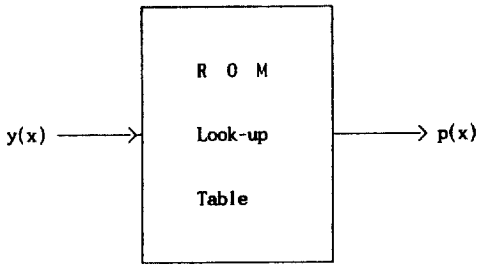

그림 9. ROM에 의한 활성화함수의 구현.

모양이 유사해짐을 알 수 있었다. 특히 그림 4와 그림 5는 각각 상대오차와 절대오차의 크기가 0.01로서 同一하지만 선형근사함수의 모양과 그 갯수가 相異하였다.

위와 같은 경우 선형함수의 갯수가 많으므로 하드웨어화했을 때 계산식을 변경시켜주는 회로가 필요하며 이를 구현하기 위한 하드웨어 구성 회로를 고찰하였다.

참고문헌

1. Yoh-Han Pao, *Adaptive Pattern Recognition and Neural Networks,* Addison-Wesley Publishing Company, Inc. pp.122, 176, 1989.
2. Judith E. Dayhoff, *Neural Network Architectures An Introduction,* Van Nostrand Reinhold, pp.64-65, 1990.
3. Bart Kosko, *Neural Networks for Signal Processing,* Prentice-Hall, pp.191, 1992.

A Study on the Realization of Activation Function in Neural Network

In-Chung Park

ABSTRACT

The activation function is nonlinear when the equivalent model of a neuron is considered. Expensive numeric processors or a high speed computer has been used to realize the function. In this paper two kinds of error are considered and the number of linear functions in a certain interval is investigated. A new hardware implementation is studied to realize a number of linear functions.

부록 1. 근사함수를 구하고 그래프를 그리는 프로그램

```
/* AX-AP005.c */
/* approximated sigmoid function */

#include <math.h>
#include <stdlib.h>
#include <graphics.h>
#define DEGREE 256
#define MAX 300
#define LIM 5

int j, l, m, n=0, index[50];
float signal[300];
main()
{
   int i,x1,y1,x2,y2;
   int graphdrive=DETECT,graphmode;
   float value[300], Gain, Center, tol;
   float y, yy, yyy, p, slope, slope1, slope2;
   float ya[MAX], yo[MAX], x[MAX], deltax, error, aerror, xi, b, bb;
     printf("\7");
     do  {
            printf("Gain is    ");
            scanf("%f", &Gain);
            printf("Center is    ");
            scanf("%f", &Center);
            j=0;
            value[0]=0.0;
            do {
               signal[j]= 1.0 /( 1.0 + exp(-Gain*(-LIM+value[j]-Center)));
               j++;
               value[j] = value[j-1] + (2.*LIM)/256.;
               } while(j < 256);

            initgraph(&graphdrive,&graphmode,"");
            setcolor(LIGHTGREEN);
            line(30,20,286,20); line(30,20,30,330);
            line(30,330,286,330); line(286,330,286,20); /* square */

            line(30,175,286,175); line(158,20,158,330);

            outtextxy(1,30,"1.0"); outtextxy(1,175,"0.5");
            outtextxy(1,320,"0.0");  /* scale of vertical axis */
```

```
        outtextxy(15,340,"-5");
        outtextxy(156,340,"0");
        outtextxy(283,340,"5");
        outtextxy(20,370,"Fig.1  Generalized Sigmoid Function");

  y1=100; x1=30;
  moveto(x1,y1);
  setcolor(WHITE);
  for(i=0; i < DEGREE; i++){
     y1 = (300-300*signal[i] + 25);
     lineto(i+30,y1);
     }
     getch(); closegraph();

 for(i=0; i<MAX; i++) { ya[i]=0.0;  yo[i]=0.0; }

 i=0; j=1;
 xi = 0.0; deltax=10./256.; b=0.5;
   y = 1.0 /( 1.0 + exp(-Gain*(xi-Center)));
   yy = 1.0 /( 1.0 + exp(-Gain*(xi-Center + 3.0*deltax)));
    slope1 = (yy - y) / (3.0*deltax);
    slope = slope1;
    printf("\n\n<< p = slope * xi + 0.5 >> slope = %f\n\n", slope);

tol = 0.005;
/*  First Interval  */
 do
   {
    y = 1.0 /( 1.0 + exp(-Gain*(xi-deltax-Center)));
    p = slope * xi + b;
    aerror = fabs(y - p)/y;
    xi = xi + deltax;
    if ( p > 1.0 ) p = 1.0;
    ya[i] = p;
    i++;
    printf("\ni=%d, b=%f, xi=%3.2f, aer=%lf, slope=%lf, y=%lf, p=%lf",
                i, b, xi, aerror, slope, y, p);
   } while (( i< 128) && (aerror < tol ));
   getch(); printf("\n");

bb = 1.0 /( 1.0 + exp(-Gain*(xi-deltax-Center)));
y = 1.0 /( 1.0 + exp(-Gain*(xi-Center)));
yy = 1.0 /( 1.0 + exp(-Gain*(xi-Center + 2.*deltax)));
slope = (yy - y) / (2.*deltax);
```

```
   /* Second Interval */
   do{
      y = 1.0 /( 1.0 + exp(-Gain*(xi-Center)));
      p = slope * xi + bb;
      aerror = fabs(y - p)/y;
      xi = xi + deltax;
      if ( p > 1.0 ) p = 1.0;
      ya[i] = p;
      i++;
       printf("\n[i=%d] bl=%f, xi=%5.4f, err=%lf, slp=%lf, y=%5.4f,
              p=%5.4f", i, bb, xi, aerror, slope, y, p);
    } while (( i < 128  ) && ( aerror < tol ) && ( p <= 1.0 ));
      getch();
      printf("\n\n");

      /*          */
l = 0;
 do {
      bb = 1.0 /( 1.0 + exp(-Gain*(xi-deltax-Center)));
      y = 1.0 /( 1.0 + exp(-Gain*(xi-Center)));
      yy = 1.0 /( 1.0 + exp(-Gain*(xi-Center + (3.)*deltax)));
      slope = (yy - y) / ((3.)*deltax);

         /* Third Interval */
do  {
      y = 1.0 /( 1.0 + exp(-Gain*(xi-Center)));
      p = slope * xi + bb;
      aerror = fabs(y - p)/y;
      xi = xi + deltax;
      if (p>1.0) p=1.0;
      ya[i] = p;
      i++;
       printf("\n<%d> b%d=%f, xi=%lf, err=%lf, slope=%lf, y=%5.4f,
              p=%5.4f", i, l+2, bb, xi, aerror, slope, y, p);
       l++;
      } while (( i < 128 ) && ( aerror < tol ) && ( p <= 1.0 ));
      if ( i == 127 )
         { printf("\7");  getch(); }
  } while (( i < 129) && (l < 120 ) && ( p <= 1.0 ));
```

```
        /* Fifth Interval */
        do{
           ya[i] = p = 1.0;
           i++;
           } while ( i < 256 );
           getch();

        initgraph(&graphdrive,&graphmode,"");
        setcolor(12);
        line(30,20,286,20); line(30,20,30,330);
        line(30,330,286,330); line(286,330,286,20); /* square */

        line(30,175,286,175); line(158,20,158,330);

        outtextxy(1,30,"1.0"); outtextxy(1,175,"0.5");
        outtextxy(1,320,"0.0");  /* scale of vertical axis */

        outtextxy(15,340,"-5");
        outtextxy(156,340,"0");
        outtextxy(283,340,"5");

        outtextxy(20,370,"Fig.2  Approximated Sigmoid Function");

        for(i=0; i<128; i++) ya[i+128] = ya[i];

        for(i=0; i<128; i++) {
           ya[i] = 1. - ya[255-i];
           }
        setcolor(15);
        y1=320; x1=30;
        moveto(x1,y1);

        for(i=0; i < DEGREE; i++){
           y1 = (300-300*ya[i] + 25);
           lineto(i+31,y1);
           }

           getch(); getch(); closegraph();
        } while(Gain>-1);
}
```

부록 2. 룩업 테이블 데이타 생성 프로그램

```
/* sigmd_3.c and Dec-to-Bin */
/* generalized sigmoid function */

#include <math.h>
#include <stdlib.h>
#include <graphics.h>
#define DEGREE 256
#define LIM 5

int j;
float signal[600];
main()
{
   int jj, num128[600], bit, num32768[600];
   unsigned int mask;

   int i,x1,y1,x2,y2;
   int graphdrive=DETECT,graphmode;
   float value[600], Gain, Center;
   while(1){
            printf("Gain is   ");
            scanf("%f", &Gain);
            printf("Center is   ");
            scanf("%f", &Center);
            j=0;
            value[0]=0.0;
            do {
               signal[j]= 1.0 /( 1.0 + exp(-Gain*(-LIM+value[j]-Center)));
               j++;
               value[j] = value[j-1] + (2.*LIM)/256.;
               printf("\nvalue[%d],",j-1);

               num32768[j-1] = 32768.* (value[j-1]-LIM)/LIM;
               mask = 0x8000;
               for(jj=0; jj<16; jj++)          /* for each bit */
                  {
                  bit = (mask & num32768[j-1]) ? 1 : 0;  /* bit is 1 or 0 */
                  printf("%d ",bit);          /* print bit */
                  if(jj==7)
                     printf("-");
                  mask >>= 1;
                  }
```

```
    printf(". ");
    num128[j-1] = 128.*signal[j-1];
    mask = 0x8000;
    for(jj=0; jj<16; jj++)          /* for each bit */
       {
       bit = (mask & num128[j-1]) ? 1 : 0;  /* bit is 1 or 0 */
       printf("%d ",bit);           /* print bit */
       if(jj==7)
          printf("-");
       mask >>= 1;
       }

    if(j%10==0) printf("\n");

    } while(j < 256);   printf("\7");

initgraph(&graphdrive,&graphmode,"");

line(20,20,276,20); line(20,20,20,330);
line(20,330,276,330); line(276,330,276,20); /* square */

line(20,175,276,175); line(148,20,148,330);

outtextxy(1,30,"1.0"); outtextxy(1,175,"0.5");
outtextxy(1,320,"0.0");  /* scale of vertical axis */

outtextxy(9,340,"-5");
outtextxy(146,340,"0");
outtextxy(273,340,"5");

y1=100; x1=20;
moveto(x1,y1);

for(i=0; i < DEGREE; i++){
    y1 = (300-300*signal[i] + 25);
    lineto(i+20,y1);
    }

    getch(); closegraph();
}

}
```