

문자 인식

학과	기계공학부
학번	200721395
이름	한경수
분반	001

문자인식은 여러가지 방법이 있는데 그 중에 하나가 Perceptron 알고리즘이다.

(1) weight와 threshold의 초기화.

(2) input value에 대응되는 target value의 제시.

(3) actual output 계산 :

$$y(t) = f\left(\sum_i [x_i(t)w_i(t) - \theta]\right)$$

여기서 activation function $f(.)$: $f(u) = \begin{cases} +1 & \text{if } u \geq 0, \\ -1 & \text{if } u < 0 \end{cases}$

(cf) 교재의 공식 : $y(t) = \text{sgn} \sum_i [x_i(t)w_i(t) - \theta]$.

(4) weight 조정 :

$$w_i(t+1) = w_i(t) + \eta[d(t) - y(t)]x_i(t) \quad 0 \leq i \leq N-1$$

여기서 $d(t) = \begin{cases} +1 & \text{if 입력이 } A \text{ class} \\ -1 & \text{if 입력이 } B \text{ class} \end{cases}$, $\eta = 0 \sim 1$ 사이의 값.

(5) goto (3).

강의 자료를 그대로 가져온다면, 위와 같다.

조금 알기 쉬운 말로 풀어쓰면

weight 값을 적당히 구하는데, 구한 weight 값을 모든 문자에 대응하게 행렬곱을 하면 원하는 값이 나오도록 weight 값을 구한다.

먼저 문자를 인식하기 위한 기본 데이터는 9by7 행렬이다. 단순화하여 63의 일차원 배열이라고 생각해도 무방하다. 먼저 이전에 구했던 weight 값을 oldweight 라고 하면, 실제 데이터값 * oldweight 해서 나온 값들을 음수면 -1 양수면 +1 로 세팅을 함.그리고

$$w_i(t+1) = w_i(t) + \eta[d(t) - y(t)]x_i(t) \quad 0 \leq i \leq N-1$$

식에 의해 weight 값을 최종적으로 구하면 된다.

`char* getmatrix(int digit)` : 숫자에 맞는 숫자 매트릭스 반환

`int getweight()` : `weight` 를 구하기 위해서 호출하는 함수, 내부적으로 `subweight` 로 다시 계층적으로 구현

`int recog(char* matrix_digit)` : 숫자 배열을 넣으면 그 숫자가 어떤 숫자인지 알아맞추는 함수

`int correct[10][4]` : 각 숫자별로 정확한 목표값

`double weight[9*7][4]` : 가중치의 변수, 9×7 이 4 개가 필요함. 10 가지를 표현하기 위해선 4bit

전체적인 구조는 `main` 에서 `getweight` 를 호출하고 `getweight` 는 전체적인 루프를 도는 역할을 하고, `getweight` 안에서 호출하는 `subweight` 는 각 `weight` 를 구하기 위해 세부적인 연산을 하게 된다.

```
WEIGHT 는 6 번 돌아서 찾았습니다.
이력한 숫자 0 인입니다.
이력한 숫자 1 인입니다.
이력한 숫자 2 인입니다.
이력한 숫자 3 인입니다.
이력한 숫자 4 인입니다.
이력한 숫자 5 인입니다.
이력한 숫자 6 인입니다.
이력한 숫자 7 인입니다.
이력한 숫자 8 인입니다.
이력한 숫자 9 인입니다.
계속하려면 아무 키나 누르십시오 . . .
```

Weight 를 먼저 구한 후, 각 배열에 대해 테스트 해보니 정확히 `weight` 를 구한 것을 알 수 있었다. 한계점은 많이 노이즈가 들어간다면 인식을 할 수 없는 것이다. 이런 경우엔 다른 알고리즘이 필요하다. 기존 문자에서 조금 바뀐건 인식을 어느정도 하지만, 전체적으로 어떠한 방향으로 시프트 되면은 전혀 인식을 할 수가 없다.

문자인식은 그 활용도가 높아서 이 알고리즘 외에도 다른 수많은 알고리즘이 있겠지만, 구현하기도 쉽고 이해하기도 쉬운 이러한 알고리즘을 설명을 듣고 직접 구현해보는 시간을 가졌는데, 개인적으로 매우 유용한 시간이 되었고, 이것 말고도 다른 알고리즘도 매우 흥미가 간다. C 언어 수업에서 그냥 문법적인 내용만 배울 수 있을 것 같았는데, 이러한 내용을 배우고 직접 구현해 더욱 많은 것을 배울 수 있었다. 다만 아쉬운 건 마감시간은 저번 주였는데, 저도 그렇고 나머지 몇몇 인원도 그렇고 과제 데드라인을 지키느라 다른걸 감수하면서 했을 텐데, 그 당시 과제를 가지고 오지 않은 인원과 차별을 두지 못하는 점에 대해서는 아쉽습니다.

0 차 weight :				1 차 weight :				2 차 생략	3 차 weight :			
0.16	0.39	0.53	0.71	0.16	0.39	0.53	0.71		0.16	0.39	0.53	0.71
0.68	0.69	0.98	0.53	0.68	0.69	0.98	0.53		0.68	0.69	0.98	0.53
0.97	0.86	0.54	0.99	0.97	0.86	0.54	0.99	0.97	0.86	0.54	0.99	
-0.78	-0.41	-2.54	-0.44	-0.78	-0.41	-2.54	-0.44	-0.78	-0.41	-2.54	-0.44	
0.21	0.28	0.14	1.00	0.21	0.28	0.14	1.00	0.21	0.28	0.14	1.00	
0.63	0.93	0.41	0.32	0.63	0.93	0.41	0.32	0.63	0.93	0.41	0.32	
0.63	0.98	0.49	0.46	0.63	0.98	0.49	0.46	0.63	0.98	0.49	0.46	
0.97	0.90	0.99	0.85	0.97	0.90	0.99	0.85	0.97	0.90	0.99	0.85	
-1.46	-0.21	0.32	1.08	-2.70	1.97	3.22	1.08	-2.70	5.52	4.26	1.08	
0.37	-1.02	-0.80	-1.10	0.37	-0.53	-0.77	-1.10	0.37	-0.04	-0.77	-1.10	
-2.48	-2.37	-2.64	-0.83	-2.48	-2.36	-1.21	-0.47	-2.48	-1.78	-1.21	-0.47	
-2.59	-2.03	-1.72	-0.54	-2.59	-5.55	-0.81	-0.54	-2.59	-3.13	-0.81	-0.54	
0.40	-0.82	-0.41	-1.65	0.40	-1.28	0.02	-1.65	0.40	0.22	0.02	-1.65	
0.02	0.02	0.87	0.46	0.02	0.02	0.87	0.46	0.02	0.02	0.87	0.46	
0.74	0.82	0.17	0.72	0.74	0.82	0.17	0.72	0.74	0.82	0.17	0.72	
0.05	-0.70	-0.54	-3.34	-1.24	-1.72	-1.68	-3.34	-1.24	0.18	-1.36	-3.34	
0.96	0.38	0.83	0.79	0.92	0.38	-0.39	0.79	0.92	0.38	-0.39	0.79	
-1.10	-0.71	-0.83	2.41	-1.45	-0.71	-2.27	2.41	-1.45	-0.71	-2.27	2.41	
-0.72	-1.51	-0.07	-1.08	-2.51	-1.51	-0.71	0.88	-2.51	-1.04	-0.71	0.88	
0.15	-1.61	-1.01	1.01	-1.09	-2.11	-1.33	1.01	-1.09	-0.58	-1.33	1.01	
0.69	0.91	0.40	1.26	-1.17	0.91	-0.16	1.26	-1.17	0.91	-0.16	1.26	
0.75	0.51	0.83	0.90	0.75	0.51	0.83	0.90	0.75	0.51	0.83	0.90	
0.48	-0.77	-0.59	-1.64	-1.31	-0.79	-0.54	-1.64	-1.31	1.18	0.04	-1.64	
0.72	0.12	0.85	1.00	0.72	0.12	0.85	1.00	0.72	0.12	0.85	1.00	
-1.06	-0.50	0.64	0.19	-1.06	-0.50	0.64	0.19	-1.06	-0.50	0.64	0.19	
-3.00	-0.58	-0.22	-0.75	-3.00	-0.58	1.38	1.09	-3.00	0.93	1.38	1.09	
0.85	-1.08	-1.54	-0.86	0.85	-4.17	-2.11	-0.86	0.85	-2.23	-2.11	-0.86	
0.81	0.05	0.66	0.16	0.81	0.05	0.66	0.16	0.81	0.05	0.66	0.16	
0.44	0.49	0.39	0.70	0.44	0.49	0.39	0.70	0.44	0.49	0.39	0.70	
-0.06	-0.17	-1.32	-3.20	-1.92	-0.43	-3.28	-3.20	-1.92	0.87	-2.28	-3.20	
1.33	-0.25	-1.31	0.03	1.33	-0.50	-2.39	0.03	1.33	-0.50	-2.39	0.03	
-1.47	-1.83	-1.63	-1.34	-1.47	-4.48	-0.81	-0.73	-1.47	-4.48	-0.81	-0.73	
0.59	0.29	-2.19	-0.02	0.59	-1.99	-2.30	-0.02	0.59	-0.81	-2.30	-0.02	
-0.18	-1.22	0.45	-1.02	-0.18	-1.75	0.48	-1.02	-0.18	-1.44	0.48	-1.02	
0.11	0.56	0.89	0.22	0.11	0.56	0.89	0.22	0.11	0.56	0.89	0.22	
0.73	0.83	0.61	0.49	0.73	0.83	0.61	0.49	0.73	0.83	0.61	0.49	
0.25	-0.62	-2.72	-2.12	-1.23	-1.99	-4.58	-2.12	-1.23	-0.34	-4.42	-2.12	
-0.60	-0.29	0.97	-1.24	-1.56	0.98	2.70	-1.24	-1.56	1.71	4.17	-1.24	
-3.30	-1.10	-1.12	-0.26	-3.92	-0.93	-1.72	-0.26	-3.92	0.76	-0.46	-0.26	
-0.98	-0.26	1.70	0.30	-1.02	0.88	-0.26	2.11	-1.02	2.20	0.15	2.11	
1.25	1.55	-1.89	0.79	-0.32	0.72	-2.64	0.79	-0.32	2.65	-2.64	0.79	
-0.21	2.21	-1.37	0.74	-1.95	2.21	-1.76	0.74	-1.95	2.35	-1.76	0.74	
0.96	0.80	0.50	0.32	0.96	0.80	0.50	0.32	0.96	0.80	0.50	0.32	
0.16	-1.67	-0.74	-3.02	0.16	-2.18	-0.07	-3.02	0.16	-2.18	0.29	-3.02	
-0.71	-1.87	-0.20	-2.53	-0.71	-1.87	0.65	-2.53	-0.71	-1.87	0.65	-2.53	
-1.88	0.22	-0.60	-0.52	-1.88	0.22	1.03	-0.52	-1.88	0.22	1.03	-0.52	
-2.37	-1.88	-0.84	-0.64	-3.88	-1.86	1.55	0.09	-3.88	-0.49	2.23	0.09	
2.46	-1.21	0.50	1.96	1.39	-1.89	0.07	1.96	1.39	-1.26	0.07	1.96	
0.01	-0.67	0.85	-1.30	0.01	-0.67	2.60	-1.30	0.01	-0.67	2.60	-1.30	
0.67	0.73	0.54	0.64	0.67	0.73	0.54	0.64	0.67	0.73	0.54	0.64	
1.68	-1.02	1.95	-1.52	1.68	-0.62	2.47	-1.52	1.68	-0.62	2.71	-1.52	
1.40	0.11	1.12	-0.87	1.40	0.47	1.39	-0.87	1.40	0.47	2.30	-0.87	
-1.31	-2.40	-0.31	-0.31	-1.31	-1.48	-1.16	0.46	-1.31	-1.48	-0.05	0.46	
0.91	0.34	-0.11	-0.39	-1.06	1.36	-2.66	-0.39	-1.06	1.59	-2.02	-0.39	
2.09	0.18	2.26	0.76	2.09	0.66	2.63	0.76	2.09	1.06	2.63	0.76	
0.71	0.90	0.06	0.30	0.71	0.90	0.06	0.30	0.71	0.90	0.06	0.30	
0.22	0.64	0.84	0.25	0.22	0.64	0.84	0.25	0.22	0.64	0.84	0.25	
0.41	0.07	0.35	0.44	0.41	0.07	0.35	0.44	0.41	0.07	0.35	0.44	
0.73	0.43	0.13	0.58	0.73	0.43	0.13	0.58	0.73	0.43	0.13	0.58	
-0.20	-0.43	-0.58	1.67	-1.37	-0.43	-1.40	1.67	-1.37	-0.43	-1.40	1.67	
0.92	0.99	0.87	0.92	0.92	0.99	0.87	0.92	0.92	0.99	0.87	0.92	
0.26	0.43	0.93	0.23	0.26	0.43	0.93	0.23	0.26	0.43	0.93	0.23	
0.04	0.63	0.85	0.71	0.04	-1.40	0.85	0.71	0.04	-1.40	0.85	0.71	

소스 코드

```
char matrix_0[63] =
{
    0,0,0,1,0,0,0,
    0,0,1,0,1,1,0,
    0,1,0,0,0,1,0,
    0,1,0,0,0,1,0,
    0,1,0,0,0,1,0,
    0,1,0,0,0,1,0,
    0,0,1,0,1,0,0,
    0,0,0,1,0,0,0,
    0,0,0,0,0,0,0,
};

char matrix_1[63] = {
    0,0,0,1,0,0,0,
    0,0,0,1,0,0,0,
    0,0,0,1,0,0,0,
    0,0,0,1,0,0,0,
    0,0,0,1,0,0,0,
    0,0,0,1,0,0,0,
    0,0,0,1,0,0,0,
    0,0,0,1,0,0,0,
    0,0,0,1,0,0,0,
};

char matrix_2[63] = {
    0,0,0,0,0,0,0,
    0,0,0,1,0,0,0,
    0,0,0,0,1,0,0,
    0,0,0,0,1,0,0,
    0,0,0,0,1,0,0,
    0,0,0,1,0,0,0,
    0,0,1,0,0,0,0,
    0,1,1,1,1,0,1,
    0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,
};

char matrix_3[63] = {
    0,0,0,0,0,0,0,
    0,0,0,1,0,0,0,
    0,0,0,0,1,0,0,
    0,0,0,0,1,0,0,
    0,0,0,1,0,0,0,
    0,0,0,0,1,0,0,
    0,0,0,0,1,0,0,
    0,0,0,1,0,0,0,
    0,0,0,0,0,0,0,
};

char matrix_4[63] =
{
    0,0,0,0,0,0,0,
    0,1,0,0,1,0,0,
    0,1,0,0,1,0,0,
    0,1,0,0,1,0,0,
    0,1,0,0,1,0,0,
    0,1,1,1,1,1,1,
    0,0,0,0,1,0,0,
    0,0,0,0,1,0,0,
    0,0,0,0,0,0,0,
};

char matrix_5[63] =
{
    0,0,0,0,0,0,0,
    0,1,0,0,0,0,0,
    0,1,1,1,1,1,1,
    0,1,0,0,0,0,0,
    0,1,0,0,0,0,0,
    0,1,1,1,1,1,1,
    0,0,0,0,1,1,0,
    0,0,0,0,1,0,0,
    0,0,0,1,0,0,0,
};

char matrix_6[63] =
{
    0,0,0,0,0,0,0,
    0,1,0,0,0,0,0,
    0,1,0,0,0,0,0,
    0,1,0,0,0,0,0,
    0,1,0,0,0,0,0,
    0,1,1,1,1,0,0,
    0,1,0,0,1,0,0,
    0,1,1,1,1,0,0,
    0,0,0,0,0,0,0,
};

char matrix_7[63] =
{
    0,0,0,0,0,0,0,
    0,1,1,1,1,1,0,
    0,1,0,0,0,1,0,
    0,1,0,0,0,1,0,
    0,0,0,0,0,1,0,
    0,0,0,0,0,1,0,
    0,0,0,0,0,1,0,
    0,0,0,0,0,1,0,
    0,0,0,0,0,0,0,
};

char matrix_8[63] = {
    0,0,0,0,0,0,0,
    0,1,1,1,1,1,0,
    0,1,0,0,0,1,0,
    0,1,0,0,0,1,0,
    0,1,1,1,1,1,0,
    0,1,0,0,0,1,0,
    0,1,1,1,1,1,0,
    0,0,0,0,0,0,0,
};

char matrix_9[63] =
{
    0,0,0,0,0,0,0,
    0,1,1,1,1,1,0,
    0,1,0,0,0,1,0,
    0,1,0,0,0,1,0,
    0,1,1,1,1,1,0,
    0,0,0,0,0,1,0,
    0,0,0,0,0,1,0,
    0,0,0,0,0,1,0,
    0,0,0,0,0,0,5,
};

int correct[10][4] = {
    -1,-1,-1,-1,
    -1,-1,-1,1,
    -1,-1,1,-1,
    -1,-1,1,1,
    -1,1,-1,-1,
    -1,1,-1,1,
    -1,1,1,-1,
    -1,1,1,1,
    1,-1,-1,-1,
    1,-1,-1,1,
};

double weight[9*7][4];
#include <stdio.h>
#include <memory.h>
#include <stdlib.h>
#include <time.h>
char* getmatrix(int digit)
{
    switch(digit)
    {
        case 0: return matrix_0; case 1: return matrix_1;
        case 2: return matrix_2; case 3: return matrix_3;
        case 4: return matrix_4; case 5: return matrix_5;
```

```

        case 6:      return matrix_6; case 7:      return matrix_7;
        case 8:      return matrix_8; case 9:      return matrix_9;
    }
    return 0;
}

int subgetweight(int digit)
{
    int whether[4] = {0,};
    char* matrix_d = getmatrix(digit);
    int i, j, z;
    int modify = 0;
    for(j=0; j<4; j++)
    {
        for(i = 0; i<=62; i++)
        {
            whether[j] += matrix_d[i] * weight[i][j];
        }
        if(whether[j] < 0 )
            whether[j] = -1;
        else
            whether[j] = 1;
    }
    for(i=0; i<4; i++)
    {
        if(whether[i] != correct[digit][i])
        {
            modify = 1;
            for(z=0; z<=62; z++)
            {
                weight[z][i] = weight[z][i] +
                    (double)rand()/(double)RAND_MAX*(correct[digit][i] -
whether[i])*matrix_d[z];
            }
        }
    }
    return modify;
}

int getweight()
{
    int i=0;
    int entcounter = 0;
    while(1)
    {
        int t = 0;
        for(i=0; i<=9; i++)
        {
            t += subgetweight(i);
        }
        entcounter++;
        if(t == 0)
            break;
    }
    return entcounter;
}

int recog(char* matrix_digit)
{
    int i, j;
    int whether[4] = {0,};
    for(j=0; j<4; j++)
    {
        for(i = 0; i<=62; i++)
        {
            whether[j] += matrix_digit[i] * weight[i][j];
        }
        if(whether[j] < 0 )
            whether[j] = -1;
    }
}

```

```

        else
            whether[j] = 1;
    }

    for(j=0; j<=9; j++)
    {
        int allmatch = 1;
        for(i=0; i<4; i++)
        {
            if(whether[i] != correct[j][i])
            {
                allmatch = 0;
                break;
            }
        }
        if(allmatch)
        {
            printf("입력한 숫자는 %d 입니다.\n", j);
        }
    }
}

int main()
{
    int i, j;
    int findcounter;
    srand(time(0));
    for(i=0; i<9*7; i++)
    {
        for(j=0; j<4; j++)
            weight[i][j] = (double)rand() / (double)RAND_MAX;
    }
    findcounter = getweight();
    printf("WEIGHT 는 %d 번 돌아서 찾았습니다.\n", findcounter);
    recog(matrix_0);
    recog(matrix_1);
    recog(matrix_2);
    recog(matrix_3);
    recog(matrix_4);
    recog(matrix_5);
    recog(matrix_6);
    recog(matrix_7);
    recog(matrix_8);
    recog(matrix_9);
    return 0;
}

```