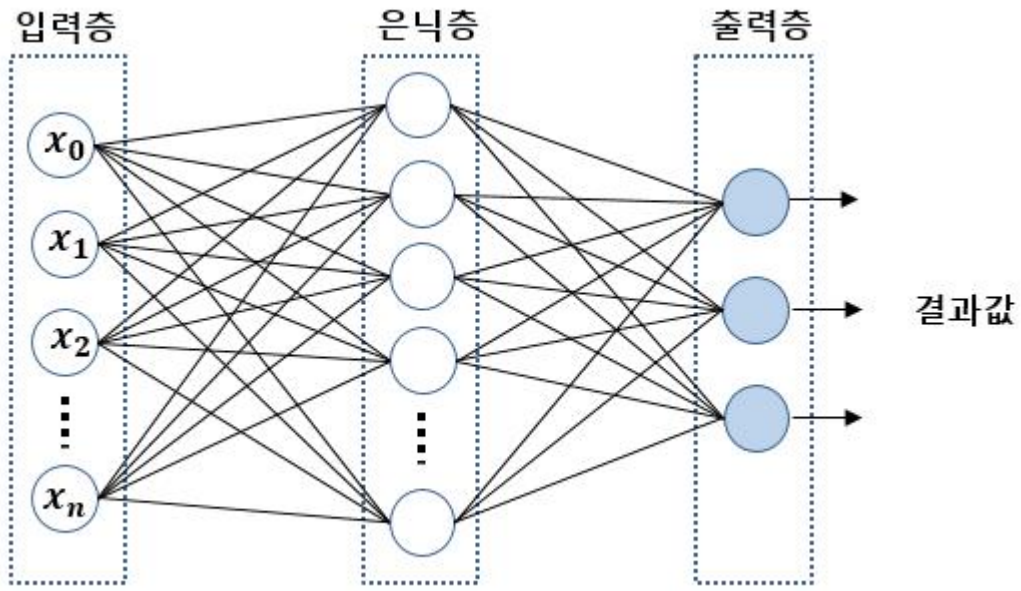
Multilayer Perceptron-리포트

이름 : 박건호

학과 : 컴퓨터공학과

학번 : 201635964



목차

① 코드 흐름 설명

Step 0 : 변수 설정
Step 1 : weight, eta 초기화 및 시그모이드 함수(sigmdid()) 결정
Step 2 : 네트워크 topology와 학습패턴 결정
Step 3 : 전방향 학습과정
Step 4 : 역방향 학습과정(가중치 수정)
Step 5 : 가중치 변화량 저장
Step 6 : 학습패턴, 교사신호 출력 후 확인, epoch 값 출력
Step 7 : 입력패턴 입력 후 확인
Step 8 : 입력패턴 계산후 출력층 출력값(uret[OUTPUT]) 저장
Step 9 : 입력패턴 교사신호 구하고 출력(teach 값 확인)
Step 10 : 입력패턴의 교사신호와 학습패턴의 교사신호 비교(결과 도출)
Step 11 : 결과값 출력

② 실행 결과

③ 전체 코드

④ 소감문 / 패턴수 8 -> 12 / 패턴크기 5*5 -> 8*6 / 은닉층 1개 추가

① 코드 흐름 설명

Step 0 : 변수 설정

```
#include <iostream>
#include <time.h>
#include <math.h>

#define NUMBER 8 //패턴 수
#define HIDE 5 //은닉층
#define OUTPUT 3 //출력층
#define ROW 8 //패턴 행
#define COL 6 //패턴 열
#define SIZE ROW*COL //패턴 크기
#define ERROR 0.000001 //오차범위
```

아래의 변수들은 전역변수로 선언하였습니다.

```
double how[HIDE][OUTPUT], hiw[SIZE][HIDE]; //hidden_output_weight,hidden_input_weight
double bhow[HIDE][OUTPUT], bhiw[SIZE][HIDE]; //before_hidden_output_weight, befor_hidden_input_weight
double ho[HIDE]; //hidden_output
double ret[OUTPUT]; //result
double d[OUTPUT]; //delta
double hd[HIDE]; //hidden_delta
double hoe[OUTPUT][HIDE]; //hidden_output_error
double hie[HIDE][SIZE]; //hidden_input_error
```

아래의 변수들은 class 안에 public 으로 선언하였습니다.

```
class Pat {
public:
    int input[SIZE]; //입력 패턴
    int teach[NUMBER]; //teach 값
```

eta(학습률), offset(활성화 함수에서 사용할 변수 이 값에 따라 1or 0으로 양극화), epoch값 선언 및 초기화

```
double eta = 0.8; //learning rate
double offset = 0.5;
int epoch = 0;
double tmp, sum;
```

leatning_rate : 0.6	leatning_rate : 0.7	leatning_rate : 0.8	leatning_rate : 0.9
Cost after iteration 0: 2.079355	Cost after iteration 0: 2.079355	Cost after iteration 0: 2.079355	Cost after iteration 0: 2.079355
Cost after iteration 1000: 0.004660	Cost after iteration 1000: 0.004001	Cost after iteration 1000: 0.003533	Cost after iteration 1000: 0.485376
Cost after iteration 2000: 0.002274	Cost after iteration 2000: 0.001952	Cost after iteration 2000: 0.001726	Cost after iteration 2000: 0.482797
Cost after iteration 3000: 0.001502	Cost after iteration 3000: 0.001290	Cost after iteration 3000: 0.001142	Cost after iteration 3000: 0.480265
Cost after iteration 4000: 0.001120	Cost after iteration 4000: 0.000963	Cost after iteration 4000: 0.000852	Cost after iteration 4000: 0.479206
Cost after iteration 5000: 0.000893	Cost after iteration 5000: 0.000768	Cost after iteration 5000: 0.000680	Cost after iteration 5000: 0.478764
Cost after iteration 6000: 0.000742	Cost after iteration 6000: 0.000638	Cost after iteration 6000: 0.000566	Cost after iteration 6000: 0.478496
Cost after iteration 7000: 0.000635	Cost after iteration 7000: 0.000546	Cost after iteration 7000: 0.000484	Cost after iteration 7000: 0.478311
Cost after iteration 8000: 0.000555	Cost after iteration 8000: 0.000477	Cost after iteration 8000: 0.000423	Cost after iteration 8000: 0.478136
Cost after iteration 9000: 0.000493	Cost after iteration 9000: 0.000424	Cost after iteration 9000: 0.000376	Cost after iteration 9000: 0.478470
Cost after iteration 10000: 0.000443	Cost after iteration 10000: 0.000381	Cost after iteration 10000: 0.000338	Cost after iteration 10000: 0.478146
Cost after iteration 11000: 0.000402	Cost after iteration 11000: 0.000346	Cost after iteration 11000: 0.000307	Cost after iteration 11000: 0.478032
Cost after iteration 12000: 0.000368	Cost after iteration 12000: 0.000317	Cost after iteration 12000: 0.000281	Cost after iteration 12000: 0.477959
Cost after iteration 13000: 0.000340	Cost after iteration 13000: 0.000293	Cost after iteration 13000: 0.000260	Cost after iteration 13000: 0.477903
Cost after iteration 14000: 0.000315	Cost after iteration 14000: 0.000272	Cost after iteration 14000: 0.000241	Cost after iteration 14000: 0.477859
Cost after iteration 15000: 0.000294	Cost after iteration 15000: 0.000253	Cost after iteration 15000: 0.000225	Cost after iteration 15000: 0.477822
Cost after iteration 16000: 0.000275	Cost after iteration 16000: 0.000238	Cost after iteration 16000: 0.000211	Cost after iteration 16000: 0.477790
Cost after iteration 17000: 0.000259	Cost after iteration 17000: 0.000224	Cost after iteration 17000: 0.000198	Cost after iteration 17000: 0.477763
Cost after iteration 18000: 0.000245	Cost after iteration 18000: 0.000211	Cost after iteration 18000: 0.000187	Cost after iteration 18000: 0.477740
Cost after iteration 19000: 0.000232	Cost after iteration 19000: 0.000200	Cost after iteration 19000: 0.000177	Cost after iteration 19000: 0.477719

Python에서 구현한 결과를 토대로 eta값(learning_rate)는 0.8로 선언하였습니다. 하지만 C++에서는 0.9까지 해도 정상작동을 하는 모습을 보였습니다.

Step 1 : weight, eta 초기화 및 시그모이드 함수(sigmdid()) 결정

가중치는 랜덤한 값으로 초기화 시켜줬습니다.

```
//가중치 초기화
for (int i = 0; i < HIDE; i++) {
    for (int j = 0; j < OUTPUT; j++) {
        bhow[i][j] = (((rand() % 10) + 1) * 0.01);
    }
}

for (int i = 0; i < SIZE; i++) {
    for (int j = 0; j < HIDE; j++) {
        bhiw[i][j] = (((rand() % 10) + 1) * 0.01);
    }
}
```

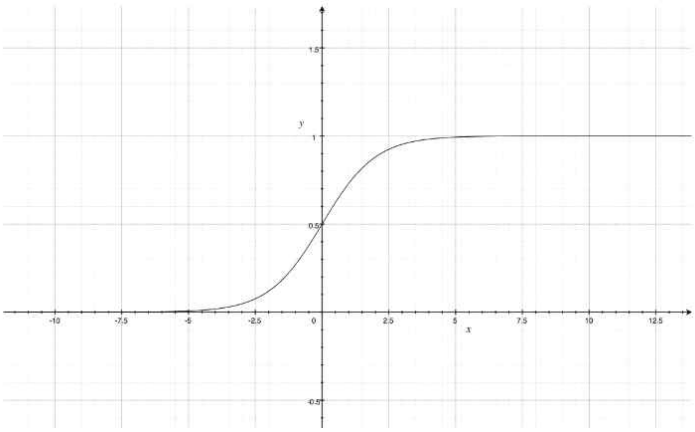
학습률 초기화

저번 과제로 제출한 파이썬으로 실행하였을 때 0.8이후의 값에서는 제대로 학습되지 않아 0.8로 설정시켜주었습니다.

```
double eta = 0.8; //learning rate
```

시그모이드 함수

```
double sigmoid(double a) {
    return 1 / (1 + exp(-a));
}
```



$$s(z) = \frac{1}{1+e^{-z}}$$

목표값이 0 또는 1이기 때문에 y = wx + b를 이용하는 것 만으로는 의미가 없다. 따라서 Odds를 이용하는데 Odds는 다음과 같이 정의 된다. 확률 p가 주어져 있을 때 $Odds(p) := \frac{p}{1-p}$ 로 정의한다. 확률 p의 범위가 (0, 1)이라면 Odds(p)의 범위는 (0, ∞)이 된다. Odds에 로그함수를 취한 log(Odds(p))의 범위가 $(-\infty, \infty)$이 된다. 즉 범위가 실수 전체가 되기 때문에 log(Odds(p))의 범위가 실수이므로 이 값은 의미가 있다. $\log(Odds(p)) = wx + b$ 으로 선형회귀분석을 실시하면 w와 b를 얻을 수 있다. 위 식을 p로 정리하면 $p(x) = \frac{1}{1+e^{-(wx+b)}}$ 을 얻을 수 있는데 이것이 시그모이드이다.

Step 2 : 네트워크 topology와 학습패턴 결정

아래는 학습시킬 패턴과 교사신호이다.

```
//학습패턴
int learn_input[NUMBER][SIZE] = {
    {  //ㄱ 패턴
    1,1,1,1,1,1, 1,1,1,1,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1
},
    { //ㄴ 패턴
    1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1
},
    { //ㄷ 패턴
    1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1
},
    { //ㄹ 패턴
    1,1,1,1,1,1, 1,1,1,1,1,1, 0,0,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1
},
    {  //ㅁ 패턴
    1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1
},
    { //ㅂ 패턴
    1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1
},
    { //ㅈ 패턴
    1,1,1,1,1,1, 1,1,1,1,1,1, 0,0,1,1,0,0, 0,1,1,1,1,0, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1
},
    { //ㅇ 패턴
    0,0,1,1,0,0, 0,1,1,1,1,0, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 0,1,1,1,1,0, 0,0,1,1,0,0
}
};

//교사신호
int learn_teach[NUMBER][3] = {
    {0,0,0}, {0,0,1}, {0,1,0}, {0,1,1}, {1,0,0}, {1,0,1}, {1,0,1}, {1,1,1}
};
```

위의 값들을 main()에서 선언하였고, 아래의 값은 class안에 구현된 함수이다.

```
void Addinput(int arr[SIZE], int brr[NUMBER]) {
    for (int i = 0; i < SIZE; i++) {
        input[i] = arr[i];
    }
    for (int i = 0; i < NUMBER; i++) {
        teach[i] = brr[i];
    }
}
```

아래의 부분은 선언된 학습패턴과 교사신호의 값들을 class에 저장하는 것이다.

```
//학습패턴 입력
for (int i = 0; i < NUMBER; i++) {
    pattern[i].Addinput(learn_input[i], learn_teach[i]);
}
```

아래의 부분은 입력된 패턴을 토대로 학습을 시키고 오차값이 위에서 선언한 ERROR값 보다 작으면 while()루프를 빠져나가도록 설계하였다. 또한 한 싸이클이 실행 될 때마다 epoch값을 1씩 증가시켜 총 몇 번의 싸이클이 실행되었는지 저장한다.

```
    //패턴 학습
while (true) {
start:
    epoch++;//한 싸이클 추가

    ////////////////////////////////////////패턴 학습////////////////////////////////////
    for (int i = 0; i < NUMBER; i++) {
        pattern[i].Learn();
    }


    ////////////////////////////////////////가중치 오차범위 검사////////////////////////////////////
    for (int x = 0; x < NUMBER; x++) {
        for (int i = 0; i < OUTPUT; i++) {
            for (int j = 0; j < HIDE; j++) {
                if (hoe[i][j] > ERROR) {
                    goto start;
                }
                else {
                    goto end;
                }
            }
        }
    }


end:
    for (int x = 0; x < NUMBER; x++) {
        for (int i = 0; i < HIDE; i++) {
            for (int j = 0; j < SIZE; j++) {
                if (hie[i][j] > ERROR) {
                    goto start;
                }
            }
        }
    }

    cout << "학습 완료" << endl;
    cout << endl;
    break;
}
```

step3, step4, step5 전부 class의 void Learn()함수 안에 구현을 하였습니다. 이점 참고해 주시기 바랍니다.

Step 3 : 전방향 학습과정

```
//학습
void Learn() {
    //step3-(1) 입력층 -> 은닉층의 출력 값
    for (int i = 0; i < HIDE; i++) {
        sum = 0;
        for (int j = 0; j < SIZE; j++) {
            sum += input[j] * bhiw[j][i];
        }
        ho[i] = sigmoid(sum);
    }

    //step3-(2) 은닉층 -> 출력층의 출력값
    for (int i = 0; i < OUTPUT; i++) {
        sum = 0;
        for (int j = 0; j < HIDE; j++) {
            sum += (ho[j] * bhow[j][i]);
        }
        ret[i] = sigmoid(sum);
    }

    //step3-(3) 출력층 오차계산(델타)
    for (int i = 0; i < OUTPUT; i++) {
        d[i] = ret[i] * (1 - ret[i])*(teach[i] - ret[i]);
    }

    //step3-(4) 은닉층 오차계산(델타)
    for (int i = 0; i < HIDE; i++) {
        sum = 0;
        for (int j = 0; j < OUTPUT; j++) {
            sum += d[j] * bhow[i][j];
        }
        hd[i] = ho[i] * (1 - ho[i]) * sum;
    }
```

Step 4 : 역방향 학습과정(가중치 수정)

```
//step4-(1) 가중치 수정(은닉층 -> 출력층)
for (int i = 0; i < HIDE; i++) {
    for (int j = 0; j < OUTPUT; j++) {
        bhow[i][j] = bhow[i][j] + (eta * d[j] * ho[i]);
    }
}

//step4-(2) 가중치 수정(입력층 -> 은닉층)
for (int i = 0; i < SIZE; i++) {
    for (int j = 0; j < HIDE; j++) {
        bhiw[i][j] = bhiw[i][j] + (eta * hd[j] * input[i]);
    }
}
```

Step 5 : 가중치 변화량 저장

```
//step5 가중치 변화량 저장
for (int i = 0; i < OUTPUT; i++) {
    for (int j = 0; j < HIDE; j++) {
        hoe[i][j] = how[i][j] - bhow[i][j];
        how[i][j] = bhow[i][j];
    }
}

for (int i = 0; i < HIDE; i++) {
    for (int j = 0; j < SIZE; j++) {
        hie[i][j] = hiw[i][j] - bhiw[i][j];
        hiw[i][j] = bhiw[i][j];
    }
}
```

Step 6 : 학습패턴, 교사신호 출력 후 확인, epoch 값 출력

```
////////////////////////////학습 패턴 출력////////////////////////////////////
for (int i = 0; i < NUMBER; i++) {
    cout << "학습 패턴 " << i << "    ";
    cout << "teach : ";
    for (int x = 0; x < 3; x++) {
        cout << pattern[i].teach[x] << " ";
    }
    cout << endl;

    for (int j = 0; j < SIZE; j++) {
        if (pattern[i].input[j] == 1) {
            cout << "■";
        }
        else {
            cout << "□";
        }
        if (j % 6 == 5) {
            cout << endl;
        }
    }
    cout << endl;
}

cout << endl;

cout << "epoch = " << epoch << endl;
cout << endl;
```

Step 7 : 입력패턴 입력 후 확인

```
////////////////////////////////////입력/////////////////////////////////////////////
int user[SIZE] = {};
double uteach[NUMBER];
double uho[HIDE], uret[OUTPUT];
// user_hidden_output, user_ret, user_teach

cout << "패턴 입력(유효 : 1, 빈칸 : 0)" << endl;
for (int i = 0; i < SIZE; i++) {
    cin >> user[i];
}

//입력패턴 확인
cout << endl;
cout << "입력 패턴" << endl;
for (int i = 0; i < SIZE; i++) {
    if (user[i] == 1) {
        cout << "■";
    }
    else {
        cout << "□";
    }
    if (i % 6 == 5) {
        cout << endl;
    }
}
cout << endl;
```

Step 8 : 입력패턴 계산후 출력층 출력값(uret[OUTPUT]) 저장

```
////////////////////////////////////결과 계산/////////////////////////////////////////
//입력패턴 은닉층 출력값 계산
for (int i = 0; i < HIDE; i++) {
    sum = 0;
    for (int j = 0; j < SIZE; j++) {
        tmp = user[j] * bhiw[j][i];
        sum += tmp;
    }
    uho[i] = sigmoid(sum);
}

//입력패턴 출력층 출력값 계산
for (int i = 0; i < OUTPUT; i++) {
    sum = 0;
    for (int j = 0; j < HIDE; j++) {
        tmp = uho[j] * bhow[j][i];
        sum += tmp;
    }
    uret[i] = sigmoid(sum);
}
```

Step 9 : 입력패턴 교사신호 구하고 출력(teach 값 확인)

```
//입력패턴 교사신호 & 활성함수 적용
for (int i = 0; i < 3; i++) {
    uret[i] = fabs(uret[i]); //절댓값
    cout << uret[i] << "  "; //활성화 함수 적용 전 값
}
cout << endl;

double temp[3] = { offset,offset,offset };

for (int i = 0; i < 3; i++) {
    if (temp[i] < uret[i]) {
        temp[i] = uret[i];
    }
}

for (int i = 0; i < 3; i++) {
    uteach[i] = uret[i] / temp[i];
    if (uteach[i] != 1) {
        uteach[i] = 0;
    }
}


//입력패턴의 교사신호 출력
cout << endl;
cout << "입력 패턴의 teach 값 : ";
for (int i = 0; i < 3; i++) {
    cout << uteach[i] << " ";
}
cout << endl;
cout << endl;
```

입력패턴에 대한 출력층 출력함수에 대해 활성화 함수 적용하여 교사신호에 맞게 변경한다.

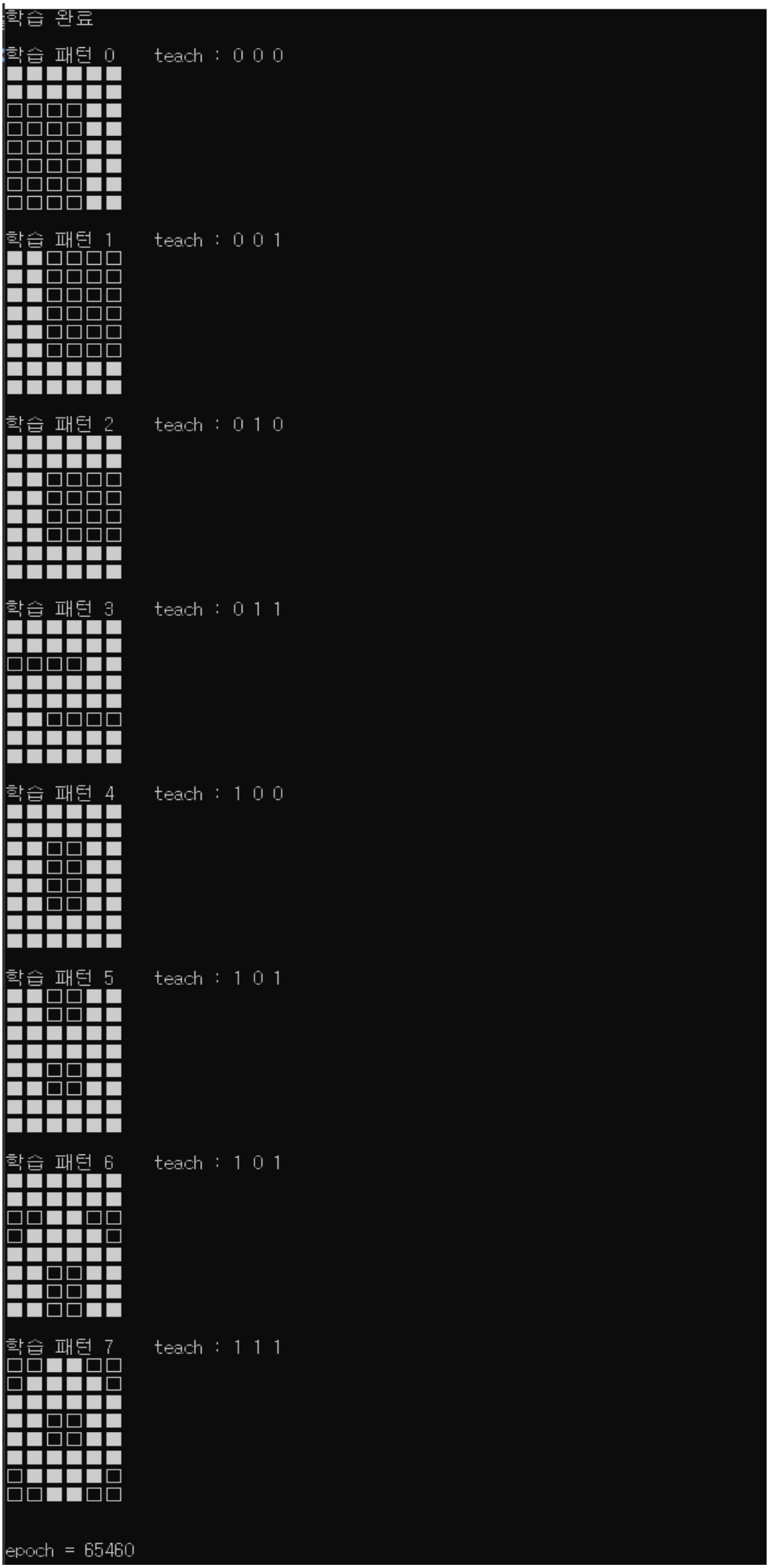
Step 10 : 입력패턴의 교사신호와 학습패턴의 교사신호 비교(결과 도출)

```
//최종결과값 계산
if (uteach[0] != 1 && uteach[1] != 1 && uteach[2] != 1) { //ㄱ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[0].input[i];
    }
}
else if (uteach[0] != 1 && uteach[1] != 1 && uteach[2] == 1) { //ㄴ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[1].input[i];
    }
}
else if (uteach[0] != 1 && uteach[1] == 1 && uteach[2] != 1) { //ㄷ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[2].input[i];
    }
}
else if (uteach[0] != 1 && uteach[1] == 1 && uteach[2] == 1) { //ㄹ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[3].input[i];
    }
}
else if (uteach[0] == 1 && uteach[1] != 1 && uteach[2] != 1) { //ㅁ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[4].input[i];
    }
}
else if (uteach[0] == 1 && uteach[1] != 1 && uteach[2] == 1) { //ㅂ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[5].input[i];
    }
}
else if (uteach[0] == 1 && uteach[1] == 1 && uteach[2] != 1) { //ㅈ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[6].input[i];
    }
}
else if (uteach[0] == 1 && uteach[1] == 1 && uteach[2] == 1) { //ㅇ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[7].input[i];
    }
}
```

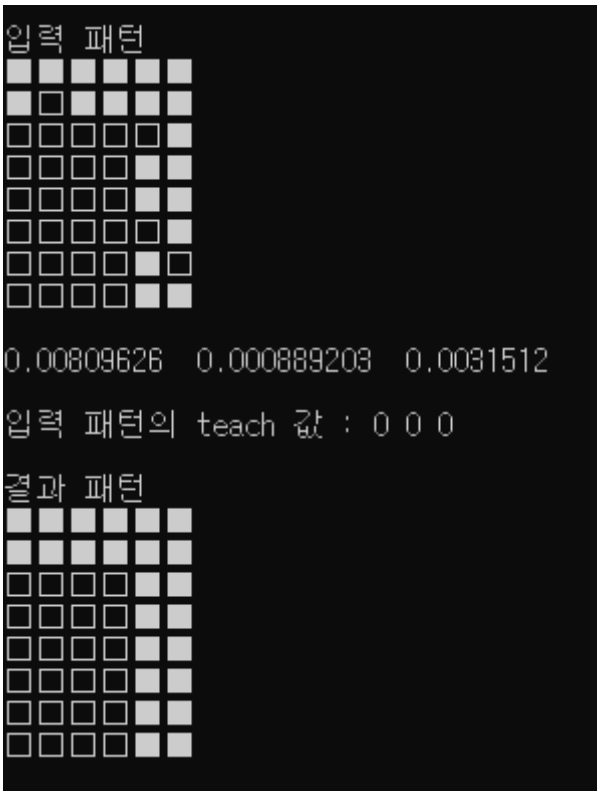
Step 11 : 결과값 출력

```
////////////////////////////////////결과 출력/////////////////////////////////////
cout << "결과 패턴" << endl;
for (int i = 0; i < SIZE; i++) {
    if (user[i] == 1) {
        cout << "■";
    }
    else {
        cout << "□";
    }
    if (i % 6 == 5) {
        cout << endl;
    }
}
cout << endl;

return 0;
```

② 실행 결과


학습 완료
학습 패턴 0 teach : 0 0 0
학습 패턴 1 teach : 0 0 1
학습 패턴 2 teach : 0 1 0
학습 패턴 3 teach : 0 1 1
학습 패턴 4 teach : 1 0 0
학습 패턴 5 teach : 1 0 1
학습 패턴 6 teach : 1 0 1
학습 패턴 7 teach : 1 1 1
epoch = 65460

가중치 값을 random하게 초기화 하여 epoch값이 일정하지 않은 것을 확인할 수 있다.
패턴 ㄱ



기존 학습된 패턴에 노이즈를 발생시켜 입력하였을 때에 정상적으로 값을 찾아내는 모습을 확인 할 수 있다.

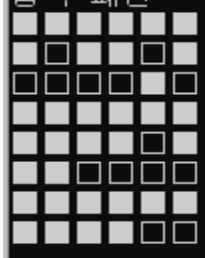
그 외의 나머지 패턴에 대한 결과

```
패턴 입력(유효 : 1, 빈칸 : 0)
1 1 0 0 0 0
1 0 0 0 0 0
0 1 0 0 0 0
0 0 0 0 0 0
1 0 1 0 0 0
1 1 0 0 0 0
1 1 0 1 1 1
1 1 1 0 1 1

입력 패턴
■■□□□□
■□□□□□
□■□□□□
□□□□□□
■□■□□□
■■□□□□
■■□■■■
■■■□■■

0.00146078  0.000380374  0.777228

입력 패턴의 teach 값 : 0 0 1

결과 패턴
■■□□□□
■■□□□□
■■□□□□
■■□□□□
■■□□□□
■■□□□□
■■■■■■
■■■■■■
```

```
패턴 입력(유효 : 1, 빈칸 : 0)
1 0 0 1 1 1
1 1 1 0 1 1
1 1 0 0 0 0
1 0 0 0 0 0
1 1 0 0 0 0
1 1 0 0 0 0
1 1 1 1 1 1
1 1 1 0 1 1

입력 패턴
■□□■■■
■■■□■■
■■□□□□
■□□□□□
■■□□□□
■■□□□□
■■■■■■
■■■□■■

0.00323054  0.847899  0.00468112

입력 패턴의 teach 값 : 0 1 0

결과 패턴
■■■■■■
■■■■■■
■■□□□□
■■□□□□
■■□□□□
■■□□□□
■■■■■■
■■■■■■
```

패턴 입력(유효 : 1, 빈칸 : 0)
1 1 1 1 1 1
1 0 1 1 0 1
0 0 0 0 1 0
1 1 1 1 1 1
1 1 1 1 0 1
1 1 0 0 0 0
1 1 1 1 1 1
1 1 1 1 0 0

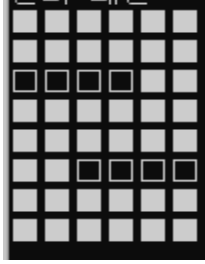
입력 패턴



0.0558757 0.999922 0.998942

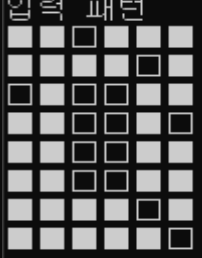
입력 패턴의 teach 값 : 0 1 1

결과 패턴



패턴 입력(유효 : 1, 빈칸 : 0)
1 1 0 1 1 1
1 1 1 1 0 1
0 1 0 0 1 1
1 1 0 0 1 0
1 1 0 0 1 1
1 1 0 0 1 1
1 1 1 1 0 1
1 1 1 1 1 0

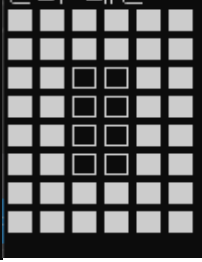
입력 패턴



0.997889 0.00125101 0.0138195

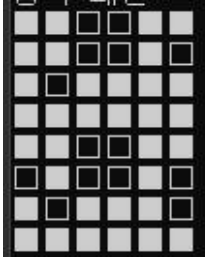
입력 패턴의 teach 값 : 1 0 0

결과 패턴



패턴 입력(유효 : 1, 빈칸 : 0)
1 1 0 0 1 1
1 1 0 0 1 0
1 0 1 1 1 1
1 1 1 1 1 1
1 1 0 0 1 1
0 1 0 0 1 0
1 0 1 1 1 0
1 1 1 1 1 1

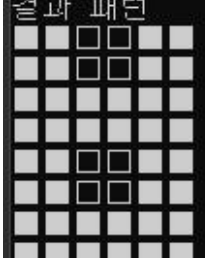
입력 패턴



0.997257 0.00104292 0.999724

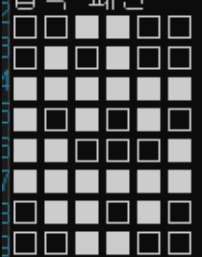
입력 패턴의 teach 값 : 1 0 1

결과 패턴



패턴 입력(유효 : 1, 빈칸 : 0)
0 0 1 1 0 0
0 1 0 1 0 0
1 1 1 1 1 1
1 0 1 0 1 0
1 1 0 0 0 1
1 1 1 1 1 1
0 1 1 0 1 0
0 0 1 1 0 0

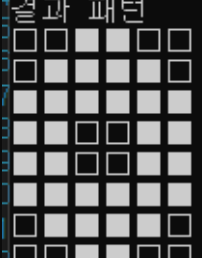
입력 패턴



0.998021 0.988467 0.999252

입력 패턴의 teach 값 : 1 1 1

결과 패턴



ㅈ패턴에 대한 결과값은 제대로 나오지 못했다. 처음에는 ㅅ을 입력하여 학습시켰지만 결과값이 정확히 나오지 못하였다. 그래서 ㅈ을 학습시켰는데 ㅈ또한 결과값이 정확히 나오지 못했다. 아무래도 문자모양 자체가 비슷하여 그런 것 같다.

완전히 다른패턴(ex A,B,C,D 등등)을 학습시켰을 경우에는 정상작동을 한다... 학습된 패턴들과 ㅅ이나 ㅈ의 패턴과의 혼동이 생기는 것 같다....

③ 전체 코드

```
#include <iostream>
#include <time.h>
#include <math.h>

#define NUMBER 8 //패턴 수
#define HIDE 5 //은닉층
#define OUTPUT 3 //출력층
#define ROW 8 //패턴 행
#define COL 6 //패턴 열
#define SIZE ROW*COL //패턴 크기
#define ERROR 0.000001 //오차범위

using namespace std;

double eta = 0.8; //learning rate
double offset = 0.5;
int epoch = 0;
double tmp, sum;

double how[HIDE][OUTPUT], hiw[SIZE][HIDE]; //hidden_output_weight,hidden_input_weight
double bhow[HIDE][OUTPUT], bhiw[SIZE][HIDE]; //before_hidden_output_weight, befor_hidden_input_weight
double ho[HIDE]; //hidden_output
double ret[OUTPUT]; //result
double d[OUTPUT]; //delta
double hd[HIDE]; //hidden_delta
double hoe[OUTPUT][HIDE]; //hidden_output_error
double hie[HIDE][SIZE]; //hidden_input_error

double sigmoid(double a) {
	return 1 / (1 + exp(-a));
}

class Pat {
public:
	int input[SIZE]; //입력 패턴
	int teach[NUMBER]; //teach 값

	void Addinput(int arr[SIZE], int brr[NUMBER]) {
		for (int i = 0; i < SIZE; i++) {
			input[i] = arr[i];
		}
		for (int i = 0; i < NUMBER; i++) {
			teach[i] = brr[i];
		}
	}

	//학습
	void Learn() {
	  //step3-(1) 입력층 -> 은닉층의 출력 값
		for (int i = 0; i < HIDE; i++) {
```

```
                sum = 0;
                for (int j = 0; j < SIZE; j++) {
                        sum += input[j] * bhiw[j][i];
                }
                ho[i] = sigmoid(sum);
        }

        //step3-(2) 은닉층 -> 출력층의 출력값
        for (int i = 0; i < OUTPUT; i++) {
                sum = 0;
                for (int j = 0; j < HIDE; j++) {
                        sum += (ho[j] * bhow[j][i]);
                }
                ret[i] = sigmoid(sum);
        }

        //step3-(3) 출력층 오차계산(델타)
        for (int i = 0; i < OUTPUT; i++) {
                d[i] = ret[i] * (1 - ret[i])*(teach[i] - ret[i]);
        }

        //step3-(4) 은닉층 오차계산(델타)
        for (int i = 0; i < HIDE; i++) {
                sum = 0;
                for (int j = 0; j < OUTPUT; j++) {
                        sum += d[j] * bhow[i][j];
                }
                hd[i] = ho[i] * (1 - ho[i]) * sum;
        }

        //step4-(1) 가중치 수정(은닉층 -> 출력층)
        for (int i = 0; i < HIDE; i++) {
                for (int j = 0; j < OUTPUT; j++) {
                        bhow[i][j] = bhow[i][j] + (eta * d[j] * ho[i]);
                }
        }

        //step4-(2) 가중치 수정(입력층 -> 은닉층)
        for (int i = 0; i < SIZE; i++) {
                for (int j = 0; j < HIDE; j++) {
                        bhiw[i][j] = bhiw[i][j] + (eta * hd[j] * input[i]);
                }
        }


        //step5 가중치 변화량 저장
        for (int i = 0; i < OUTPUT; i++) {
                for (int j = 0; j < HIDE; j++) {
                        hoe[i][j] = how[i][j] - bhow[i][j];
                        how[i][j] = bhow[i][j];
                }
```

```
                }

                for (int i = 0; i < HIDE; i++) {
                        for (int j = 0; j < SIZE; j++) {
                                hie[i][j] = hiw[i][j] - bhiw[i][j];
                                hiw[i][j] = bhiw[i][j];
                        }
                }

        }
};



int main() {
        srand(time(NULL));

        //학습패턴
        int learn_input[NUMBER][SIZE] = {
           {  //ㄱ 패턴
          1,1,1,1,1,1, 1,1,1,1,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1
        },
           { //ㄴ 패턴
          1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1
        },
           { //ㄷ 패턴
          1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1
        },
           { //ㄹ 패턴
          1,1,1,1,1,1, 1,1,1,1,1,1, 0,0,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1
        },
           {  //ㅁ 패턴
          1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1
        },
           { //ㅂ 패턴
          1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1
        },
           { //ㅈ 패턴
          1,1,1,1,1,1, 1,1,1,1,1,1, 0,0,1,1,0,0, 0,1,1,1,1,0, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1
        },
           { //ㅇ 패턴
          0,0,1,1,0,0, 0,1,1,1,1,0, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 0,1,1,1,1,0, 0,0,1,1,0,0
        }
        };

        //교사신호
        int learn_teach[NUMBER][3] = {
           {0,0,0}, {0,0,1}, {0,1,0}, {0,1,1}, {1,0,0}, {1,0,1}, {1,0,1}, {1,1,1}
        };
```

```
	Pat pattern[NUMBER];

	//가중치 초기화
	for (int i = 0; i < HIDE; i++) {
		for (int j = 0; j < OUTPUT; j++) {
			bhow[i][j] = (((rand() % 10) + 1) * 0.01);
		}
	}

	for (int i = 0; i < SIZE; i++) {
		for (int j = 0; j < HIDE; j++) {
			bhiw[i][j] = (((rand() % 10) + 1) * 0.01);
		}
	}

	//학습패턴 입력
	for (int i = 0; i < NUMBER; i++) {
		pattern[i].Addinput(learn_input[i], learn_teach[i]);
	}

		//패턴 학습
	while (true) {
	start:
		epoch++;//한 싸이클 추가

		//////////////////////////////////////////패턴 학습
////////////////////////////////////////////////
		for (int i = 0; i < NUMBER; i++) {
			pattern[i].Learn();
		}


		//////////////////////////////////가중치 오차범위 검사
//////////////////////////////////////////
		for (int x = 0; x < NUMBER; x++) {
			for (int i = 0; i < OUTPUT; i++) {
				for (int j = 0; j < HIDE; j++) {
					if (hoe[i][j] > ERROR) {
						goto start;
					}
					else {
						goto end;
					}
				}
			}
		}


	end:
		for (int x = 0; x < NUMBER; x++) {
			for (int i = 0; i < HIDE; i++) {
```

```
                        for (int j = 0; j < SIZE; j++) {
                                if (hie[i][j] > ERROR) {
                                        goto start;
                                }
                        }
                }
        }

        cout << "학습 완료" << endl;
        cout << endl;
        break;
}


//////////////////////////학습 패턴 출력////////////////////////////////////
for (int i = 0; i < NUMBER; i++) {
        cout << "학습 패턴 " << i << "    ";
        cout << "teach : ";
        for (int x = 0; x < 3; x++) {
                cout << pattern[i].teach[x] << " ";
        }
        cout << endl;

        for (int j = 0; j < SIZE; j++) {
                if (pattern[i].input[j] == 1) {
                        cout << "■";
                }
                else {
                        cout << "□";
                }
                if (j % 6 == 5) {
                        cout << endl;
                }
        }
        cout << endl;
}

cout << endl;

cout << "epoch = " << epoch << endl;
cout << endl;

////////////////////////////////입력////////////////////////////////////////
int user[SIZE] = {};
double uteach[NUMBER];
double uho[HIDE], uret[OUTPUT];
// user_hidden_output, user_ret, user_teach

cout << "패턴 입력(유효 : 1, 빈칸 : 0)" << endl;
for (int i = 0; i < SIZE; i++) {
        cin >> user[i];
```

```
	}

	//입력패턴 확인
	cout << endl;
	cout << "입력 패턴" << endl;
	for (int i = 0; i < SIZE; i++) {
		if (user[i] == 1) {
			cout << "■";
		}
		else {
			cout << "□";
		}
		if (i % 6 == 5) {
			cout << endl;
		}
	}
	cout << endl;



	////////////////////////////////////결과 계산/////////////////////////////////////
	//입력패턴 은닉층 출력값 계산
	for (int i = 0; i < HIDE; i++) {
		sum = 0;
		for (int j = 0; j < SIZE; j++) {
			tmp = user[j] * bhiw[j][i];
			sum += tmp;
		}
		uho[i] = sigmoid(sum);
	}

	//입력패턴 출력층 출력값 계산
	for (int i = 0; i < OUTPUT; i++) {
		sum = 0;
		for (int j = 0; j < HIDE; j++) {
			tmp = uho[j] * bhow[j][i];
			sum += tmp;
		}
		uret[i] = sigmoid(sum);
	}

	//입력패턴 교사신호 & 활성함수 적용
	for (int i = 0; i < 3; i++) {
		uret[i] = fabs(uret[i]); //절댓값
		cout << uret[i] << "  "; //활성화 함수 적용 전 값
	}
	cout << endl;

	double temp[3] = { offset,offset,offset };

	for (int i = 0; i < 3; i++) {
```

```
            if (temp[i] < uret[i]) {
                    temp[i] = uret[i];
            }
    }

    for (int i = 0; i < 3; i++) {
            uteach[i] = uret[i] / temp[i];
            if (uteach[i] != 1) {
                    uteach[i] = 0;
            }
    }


    //입력패턴의 교사신호 출력
    cout << endl;
    cout << "입력 패턴의 teach 값 : ";
    for (int i = 0; i < 3; i++) {
            cout << uteach[i] << " ";
    }
    cout << endl;
    cout << endl;

    //최종결과값 계산
    if (uteach[0] != 1 && uteach[1] != 1 && uteach[2] != 1) { //ㄱ패턴과 비교
            for (int i = 0; i < SIZE; i++) {
                    user[i] = pattern[0].input[i];
            }
    }
    else if (uteach[0] != 1 && uteach[1] != 1 && uteach[2] == 1) { //ㄴ패턴과 비교
            for (int i = 0; i < SIZE; i++) {
                    user[i] = pattern[1].input[i];
            }
    }
    else if (uteach[0] != 1 && uteach[1] == 1 && uteach[2] != 1) { //ㄷ패턴과 비교
            for (int i = 0; i < SIZE; i++) {
                    user[i] = pattern[2].input[i];
            }
    }
    else if (uteach[0] != 1 && uteach[1] == 1 && uteach[2] == 1) { //ㄹ패턴과 비교
            for (int i = 0; i < SIZE; i++) {
                    user[i] = pattern[3].input[i];
            }
    }
    else if (uteach[0] == 1 && uteach[1] != 1 && uteach[2] != 1) { //ㅁ패턴과 비교
            for (int i = 0; i < SIZE; i++) {
                    user[i] = pattern[4].input[i];
            }
    }
    else if (uteach[0] == 1 && uteach[1] != 1 && uteach[2] == 1) { //ㅂ패턴과 비교
            for (int i = 0; i < SIZE; i++) {
                    user[i] = pattern[5].input[i];
```

```
                }
        }
        else if (uteach[0] == 1 && uteach[1] == 1 && uteach[2] != 1) { //ㅈ패턴과 비교
                for (int i = 0; i < SIZE; i++) {
                        user[i] = pattern[6].input[i];
                }
        }
        else if (uteach[0] == 1 && uteach[1] == 1 && uteach[2] == 1) { //ㅇ패턴과 비교
                for (int i = 0; i < SIZE; i++) {
                        user[i] = pattern[7].input[i];
                }
        }

        ////////////////////////////////////결과 출력/////////////////////////////////////
        cout << "결과 패턴" << endl;
        for (int i = 0; i < SIZE; i++) {
                if (user[i] == 1) {
                        cout << "■";
                }
                else {
                        cout << "□";
                }
                if (i % 6 == 5) {
                        cout << endl;
                }
        }
        cout << endl;

        return 0;
}
```

④ 소감문

저번 시간 과제에서는 C++로 완벽히 구현하지 못해 중간에 경로를 바꿔 교수님께서 들으라고 하셨던 코세라 강의를 듣고 그 강의와 연습문제를 활용하여 Python으로 구현하여 제출하였습니다. 하지만 C++로 구현해왔던 코드가 너무 아쉽고 완성시키고 싶다는 욕심으로 계속 코드를 수정하였고 정상적으로 결과값이 나왔을 때 정말 기뻤습니다. 중간고사를 시험으로 봤다면 전체적인 공부를 하는데 시간을 투자하여 C++로 구현을 해내지 못했을 것입니다. 중간고사 대체 과제로 다층 퍼셉트론 구현을 하라고 하신 교수님께 감사한 마음까지 생겼습니다.

이번 레포트에서는 Python으로 구현 했을 때와 같이 6*8크기의 배열로 패턴들을 학습시켰습니다. 구현을 하였을 때에 가중치의 값들을 일정한 값으로 지정하여 초기화 시킨후 학습시킨 후 가중치의 값들에 따라 생기는 차이점에 대해 알아보려고 하였지만 epoch가 정상적으로 돌아가지 않아 random한 값으로 초기화 하여 구현 시켰습니다. 이 부분으로 인해 은닉층과 가중치, eta(learning_rate)의 변화에 따른 차이점들을 확인하기 힘들었습니다. 교사신호 값들을 크기 3인 배열에서 출력값 크기가 학습 패턴수만큼인 배열로도 해보고 학습패턴을 8개에서 12개로도 늘려보고 하면서(결과는 끝에 첨부 하였습니다.) 각각의 크기를 늘려나가면 사진이나 물건에 대해서도 학습시켜 결과값을 찾을 수 있겠다는 생각이 들었습니다. 나중에 기회가 된다면 한번 구현해보고 싶습니다.

홉필드, 다층 퍼셉트론을 직접 날코딩(다른 것을 참고하지 않고 아무것도 없는 상태에서 시작)해보면서 어렵다고만 생각한 것들을 해내고 나니 성취감이 매우 높았습니다. 힘들고 어려웠지만 좋은 경험을 하게 되어서 매우 좋았습니다.

< 학습패턴 8개 -> 12개 >

```
#define NUMBER 12 //패턴 수
#define HIDE 5 //은닉층
#define OUTPUT 12 //출력층
#define ROW 8 //패턴 행
#define COL 6 //패턴 열
#define SIZE ROW*COL //패턴 크기
#define ERROR 0.000001 //오차범위
```

: NUMBER, OUTPUT 12로 변경

```
//학습패턴
int learn_input[NUMBER][SIZE] = {
  { //ㄱ 패턴
  1,1,1,1,1,1, 1,1,1,1,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1, 0,0,0,0,1,1
  },
  { //ㄴ 패턴
  1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1
  },
  { //ㄷ 패턴
  1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1
  },
  { //ㄹ 패턴
  1,1,1,1,1,1, 1,1,1,1,1,1, 0,0,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1
  },
  { //ㅁ 패턴
  1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1
  },
  { //ㅂ 패턴
  1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1
  },
  { //A 패턴
  0,0,1,1,0,0, 0,1,1,1,1,0, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1
  },
  { //B 패턴
  1,1,1,1,1,0, 1,1,1,1,1,1, 1,1,0,0,1,1, 1,1,1,1,1,0, 1,1,1,1,1,0, 1,1,0,0,1,1, 1,1,1,1,1,1, 1,1,1,1,1,1
  },
  { //C 패턴
  0,0,1,1,1,1, 1,1,1,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,1,0,0,0, 0,0,1,1,1,1
  },
  { //D 패턴
  1,1,1,1,0,0, 1,1,1,1,1,0, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,0,0,1,1, 1,1,1,1,1,0, 1,1,1,1,0,0
  },
  { //E 패턴
  1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1
  },
  { //F 패턴
  1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,0,0, 1,1,1,1,1,1, 1,1,1,1,1,1, 1,1,0,0,0,0, 1,1,0,0,0,0, 1,1,0,0,0,0
  }
};
```

: 학습패턴

```
//교사신호
int learn_teach[NUMBER][NUMBER] = {
  {1,0,0,0,0,0,0,0,0,0,0,0},
  {0,1,0,0,0,0,0,0,0,0,0,0},
  {0,0,1,0,0,0,0,0,0,0,0,0},
  {0,0,0,1,0,0,0,0,0,0,0,0},
  {0,0,0,0,1,0,0,0,0,0,0,0},
  {0,0,0,0,0,1,0,0,0,0,0,0},
  {0,0,0,0,0,0,1,0,0,0,0,0},
  {0,0,0,0,0,0,0,1,0,0,0,0},
  {0,0,0,0,0,0,0,0,1,0,0,0},
  {0,0,0,0,0,0,0,0,0,1,0,0},
  {0,0,0,0,0,0,0,0,0,0,1,0},
  {0,0,0,0,0,0,0,0,0,0,0,1}
};
```

: 교사신호

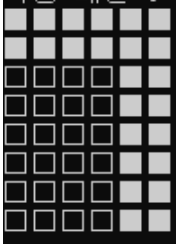
```
//최종결과값 계산
if (uteach[0] == 1) { //ㄱ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[0].input[i];
    }
}
else if (uteach[1] == 1) { //ㄴ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[1].input[i];
    }
}
else if (uteach[2] == 1) { //ㄷ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[2].input[i];
    }
}
else if (uteach[3] == 1) { //ㄹ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[3].input[i];
    }
}
else if (uteach[4] == 1) { //ㅁ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[4].input[i];
    }
}
else if (uteach[5] == 1) { //ㅂ패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[5].input[i];
    }
}
else if (uteach[6] == 1) { //A패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[6].input[i];
    }
}
else if (uteach[7] == 1) { //B패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[7].input[i];
    }
}
else if (uteach[8] == 1) { //C패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[8].input[i];
    }
}
else if (uteach[9] == 1) { //D패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[9].input[i];
    }
}
else if (uteach[10] == 1) { //E패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[10].input[i];
    }
}
else if (uteach[11] == 1) { //F패턴과 비교
    for (int i = 0; i < SIZE; i++) {
        user[i] = pattern[11].input[i];
    }
}
```

: 입력 패턴에 대한 결과값을 학습데이터와 비교

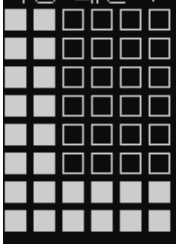
결과 화면

학습 완료

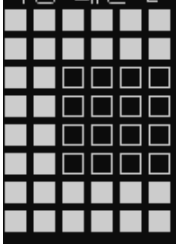
학습 패턴 0 teach : 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0



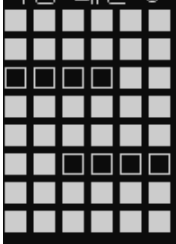
학습 패턴 1 teach : 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0



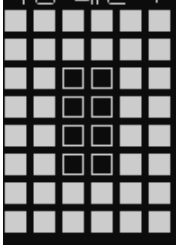
학습 패턴 2 teach : 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0



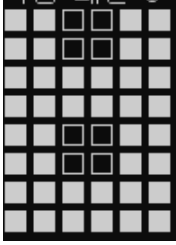
학습 패턴 3 teach : 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0



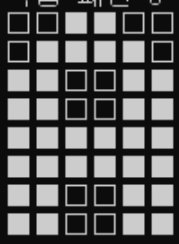
학습 패턴 4 teach : 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0



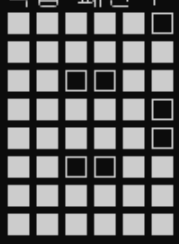
학습 패턴 5 teach : 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0



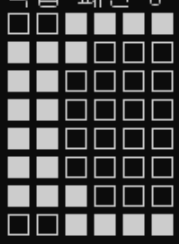
학습 패턴 6 teach : 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0



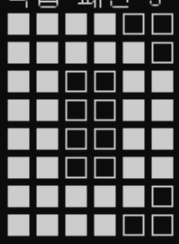
학습 패턴 7 teach : 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0



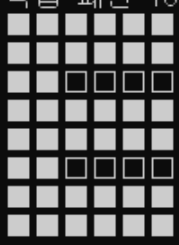
학습 패턴 8 teach : 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0



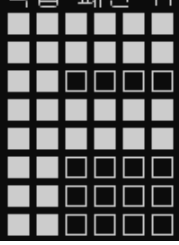
학습 패턴 9 teach : 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

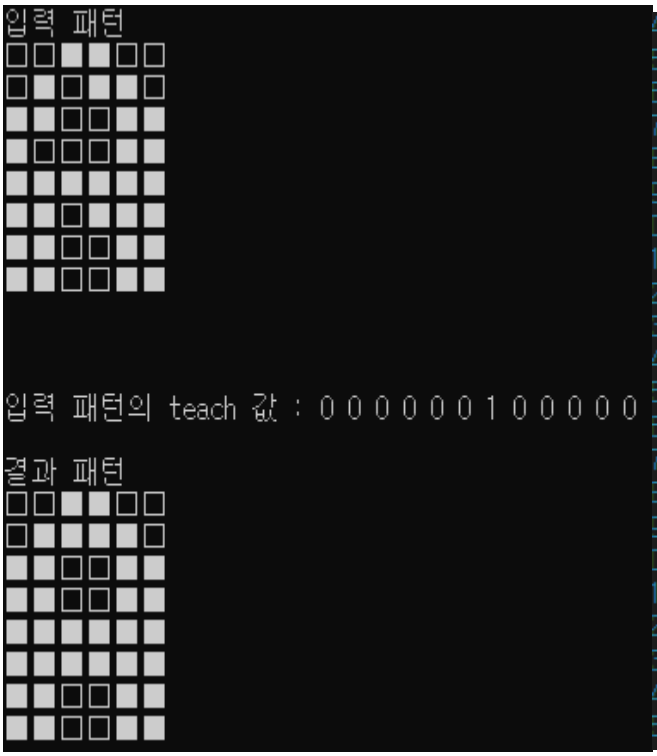


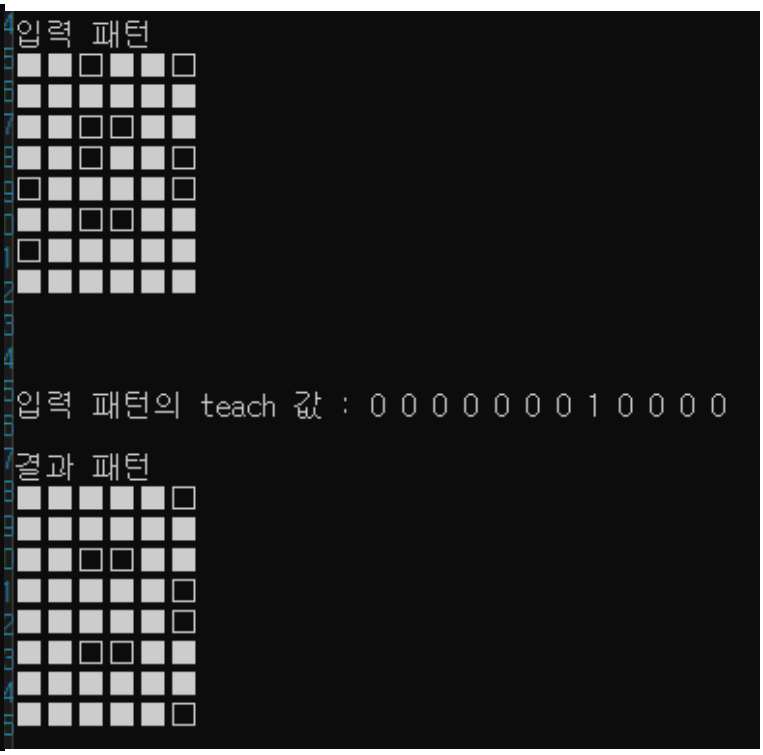
학습 패턴 10 teach : 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

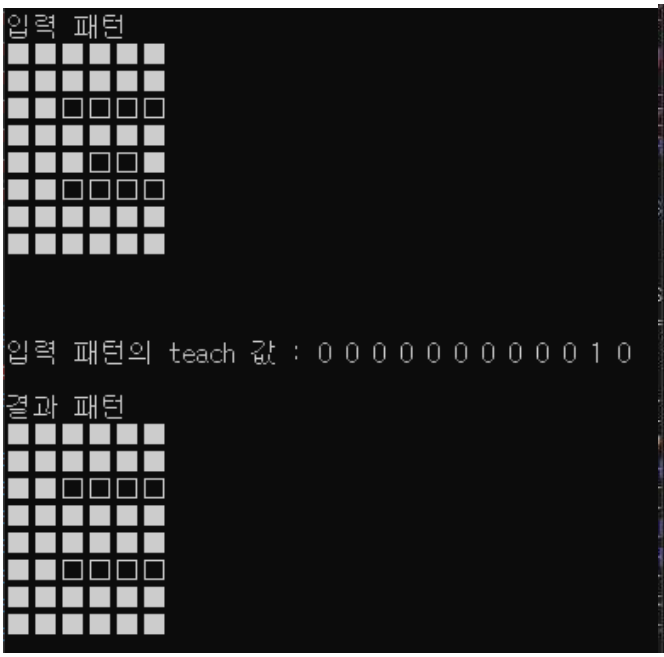


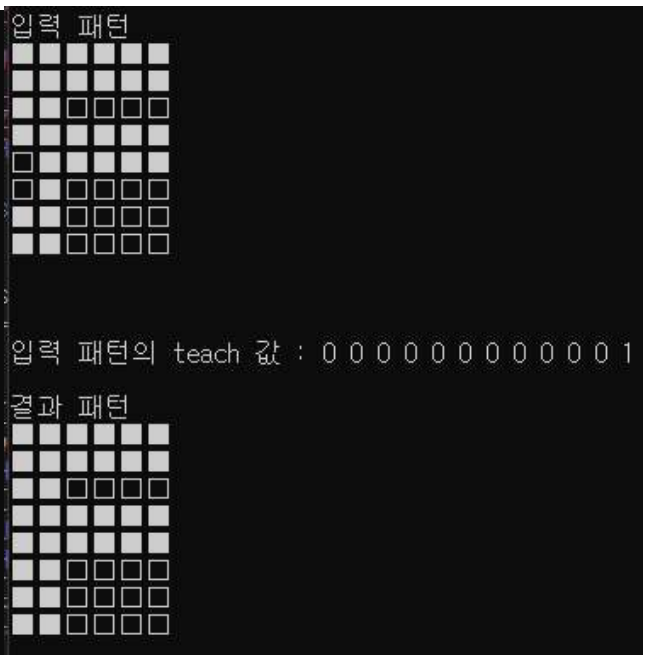
학습 패턴 11 teach : 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0







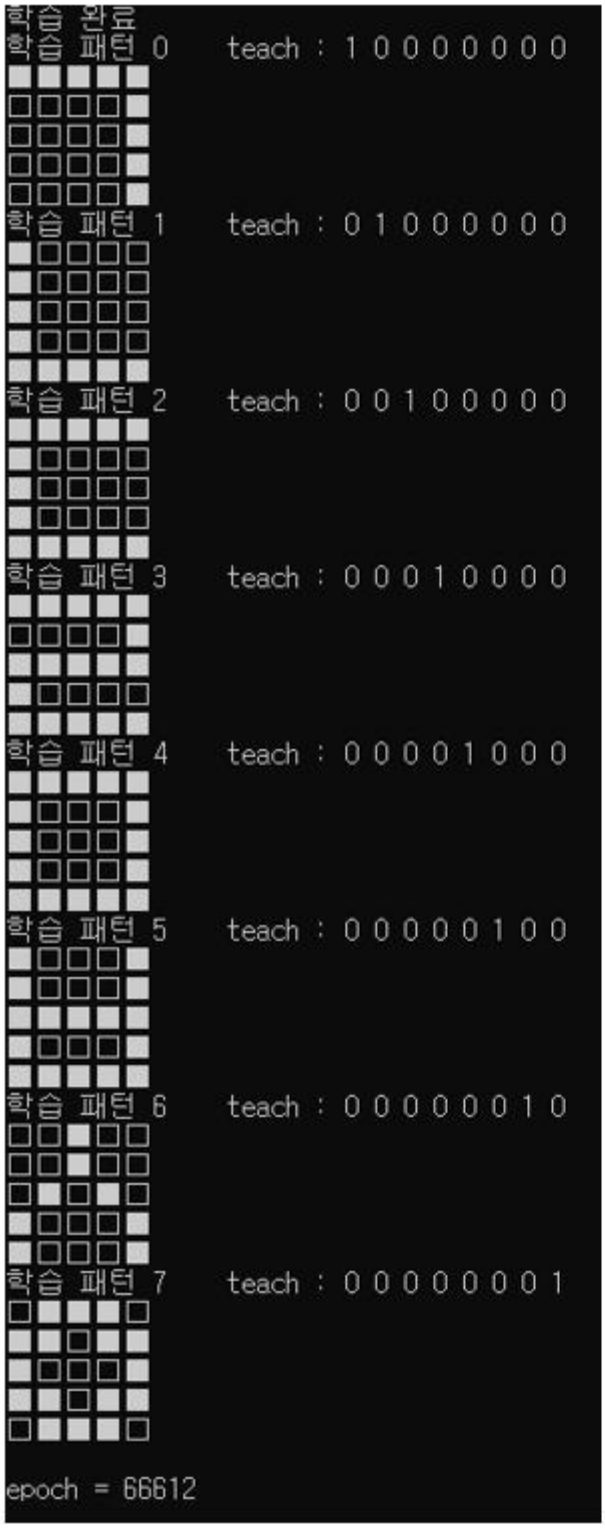


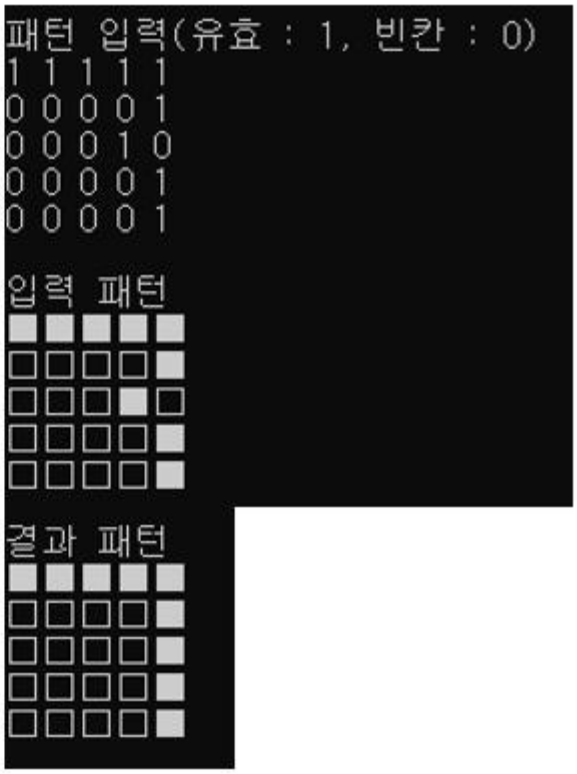


(전부 올리면 너무 길어져 새로운 패턴에 대해서만 몇 개 골라첨부하였습니다.)
캡쳐한 패턴 뿐 아니라 12개의 다른 모든 패턴에 대해서도 정상적으로 결과값을 잘 찾아내는 모습을 볼 수 있습니다. 패턴의 수가 늘어남에 따라 epoch가 너무 많이 돌아 오차율을 0.00001로 수정하여 실행 시켰습니다.

< 패턴 크기 5*5 >

처음 구현을 시켰을 때에는 5*5로 구현을 했었고 최종으로 8*6크기로 제출을 하였습니다. 패턴 크기에도 변화를 주었음을 보이고자 기존에 구현했던 5*5 크기의 결과화면도 첨부하겠습니다.





< 은닉층 추가 >

우선 추가할 은닉층의 각 변수들을 선언해줍니다.

```
double midhiw[HIDE][HIDE];
double midbhiw[HIDE][HIDE];
double midd[OUTPUT];
double midho[HIDE];
double mide[HIDE][HIDE];
```

후에 가중치를 다른 가중치와 마찬가지로 랜덤한 값으로 초기화 시켜줍니다.

```
for (int i = 0; i < HIDE; i++) {
    for (int j = 0; j < HIDE; j++) {
        midbhiw[i][j] = (((rand() % 10) + 1) * 0.01);
    }
}
```

패턴을 학습시키는 void Learn()함수 안에 다음과 같이 식을 구현합니다.

```
//step3-(1)-2 중간은닉층 -> 은닉층의 출력 값
for (int i = 0; i < HIDE; i++) {
    sum = 0;
    for (int j = 0; j < HIDE; j++) {
        sum += midbhiw[i][j] * midbhiw[j][i];
    }
    midho[i] = sigmoid(sum);
}
```

```
//step3-(4)-1 중간은닉층 오차계산(델타)
for (int i = 0; i < HIDE; i++) {
    sum = 0;
    for (int j = 0; j < HIDE; j++) {
        sum += hd[j] * midbhiw[j][i];
    }
    midd[i] = midho[i] * (1 - midho[i]) * sum;
}
```

```
//step4-(2)-1 가중치 수정(중간은닉층 -> 은닉층)
for (int i = 0; i < SIZE; i++) {
    for (int j = 0; j < HIDE; j++) {
        midbhiw[i][j] = midbhiw[i][j] + (eta * midd[j] * midho[i]);
    }
}
```

```
for (int i = 0; i < HIDE; i++) {
    for (int j = 0; j < HIDE; j++) {
        mide[i][j] = midhiw[i][j] - midbhiw[i][j];
        midhiw[i][j] = midbhiw[i][j];
    }
}
```

<위의 사진은 가중치 변화량 저장>

```
next:
    for (int x = 0; x < NUMBER; x++) {
        for (int i = 0; i < HIDE; i++) {
            for (int j = 0; j < HIDE; j++) {
                if (mide[i][j] > ERROR) {
                    goto start;
                }
                else {
                    goto end;
                }
            }
        }
    }
```

<< 가중치 오차범위 검사

입력 패턴에 대해 중간은닉층 가중치와의 출력값도 계산하여 저장

```
//입력패턴 중간은닉층 출력값 계산
for (int i = 0; i < HIDE; i++) {
    sum = 0;
    for (int j = 0; j < HIDE; j++) {
        tmp = uho[j] * midbhiw[j][i];
        sum += tmp;
    }
    midho[i] = sigmoid(sum);
}
```

<출력결과>

```
패턴 입력(유효 : 1, 빈칸 : 0)
1 1 0 0 0 0
1 1 0 0 0 0
1 1 0 0 0 0
1 0 0 0 0 0
1 0 0 0 0 0
1 1 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1

입력 패턴
■■□□□□
■■□□□□
■■□□□□
■□□□□□
■□□□□□
■■□□□□
■■■■■■
■■■■■■

0.108866  0.280194  0.921977

입력 패턴의 teach 값 : 0 0 1

결과 패턴
■■□□□□
■■□□□□
■■□□□□
■■□□□□
■■□□□□
■■□□□□
■■■■■■
■■■■■■
```

다른 패턴들도 정상적으로 출력되는 것을 확인할 수 있었습니다.
(추가된 중간 은닉층으로 입력층과 출력층의 계산식의 변수에도 변화가 생긴점 참고 부탁드립니다.)
[입력층-은닉층-출력층]으로 구현한 것과 [입력층-중간은닉층-은닉층-출력층]으로 구현한 것에 정확도를 비교하고자 하였지만 [입력층-은닉층-출력층]으로 구현한 코드에서도 정확도가 매우 높았기 때문에 값으로 실질적인 수치로 계산하기는 힘들었던 점이 아쉬웠습니다.