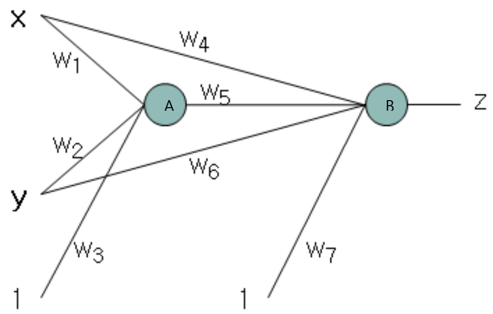
학번 : 2013311659

이름 : 곽창근

1. What are the outputs if x = 1 and y = 1, and x = 1 and y = 0, respectively? A step function is used in each neuron. $w_1 = w_2 = w_4 = w_6 = 1$, $w_3 = -1.5$, $w_5 = -2$ and $w_7 = -0.5$.



그림과 가운데 앞의 노드를 A, 뒤의 노드를 B라고 하자.

i) x=1, y=1의 경우,

A에 들어오는 input은 $xw_1 + yw_2 + w_3 = 1 + 1 - 1.5 = 0.5$ 이다. 이는 0보다 크므로, sigmoid를 통과하면 1이 된다.

그리고 B로 들어오는 input은 $xw_4 + Aw_5 + yw_6 + w_7 = 1 - 2 + 1 - 0.5 = -0.5$ 이다. 이는 0보다 작으므로, 최종 output Z는 0이다.

∴ output Z는 0이다.

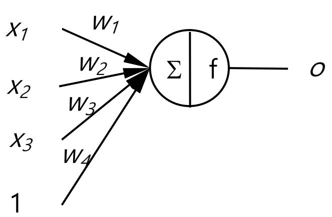
ii) x=1, y=0의 경우,

A에 들어오는 input은 $xw_1 + yw_2 + w_3 = 1 + 0 - 1.5 = -0.5$ 이다. 이는 0보다 작으므로, sigmoid를 통과하면 0이 된다.

그리고 B로 들어오는 input은 $xw_4 + Aw_5 + yw_6 + w_7 = 1 + 0 + 0 - 0.5 = 0.5$ 이다. 이는 0보다 크므로, 최종 output Z는 1이다.

∴ output Z는 1이다.

2. Set w_1, w_2, w_3, w_4 so that the output is 1 only if at least one of inputs is 1s, where f is a step function.



w_4를 -0.5로 하고, w_1, w_2, w_3가 w_4보다 큰 양수, 예를 들어 1이라면 x1, x2, x3가 모두 0이라면 output도 0이고, 이 중 하나라도 1이 된다면 output도 1이 된다. 그러므로,

$\therefore\ w_4 = -0.5,\ w_1, w_2, w_3 = 1$ 이면 된다.

3. Fill out the tables on the slides 4 ~ 7 in "9-3. Neural Networks-EBP-3.pptx" for the second iteration of the XOR neural network training.

아래 테이블들은 Second iteration일 때의 값을 채운 것들이다.

n	x_{n1}	x_{n2}	t_{n1}	net_{n1}	net_{n2}	h_{n1}	h_{n2}	net_{n1}	o_{n1}
1	1	1	0	0.03085	0.01765	0.50771	0.50441	0.06284	0.51570
2	1	0	1	0.00290	0.08775	0.50072	0.52192	0.06355	0.51588
3	0	1	1	0.11980	−0.08025	0.52991	0.47995	0.06248	0.51561
4	0	0	0	0.09185	−0.01015	0.52295	0.49746	0.06319	0.51579

n	$\frac{\partial E_n}{\partial w_{13}}$	$\frac{\partial E_n}{\partial w_{12}}$	$\frac{\partial E_n}{\partial w_{11}}$	$\frac{\partial E_n}{\partial w_{23}}$	$\frac{\partial E_n}{\partial w_{22}}$	$\frac{\partial E_n}{\partial w_{21}}$	$\frac{\partial E_n}{\partial w_{13}}$	$\frac{\partial E_n}{\partial w_{12}}$	$\frac{\partial E_n}{\partial w_{11}}$
1	0.12880	0.06497	0.06539	0.00196	0.00196	0.00196	0.00164	0.00164	0.00164
2	−0.12091	−0.06310	−0.06054	−0.00184	0.00000	−0.00184	−0.00154	0.00000	−0.00154
3	−0.12098	−0.05806	−0.06411	−0.00184	−0.00184	0.00000	−0.00154	−0.00154	0.00000
4	0.12882	0.06408	0.06737	0.00196	0.00000	0.00000	0.00163	0.00000	0.00000

$$\frac{\partial E}{\partial w} = \sum_{n=1}^{N} \frac{\partial E_n}{\partial w}$$

0.01573	0.00789	0.00811

$$\frac{\partial E}{\partial w} = \sum_{n=1}^{N} \frac{\partial E_n}{\partial w}$$

0.00024	0.00012	0.00012	0.00019	0.00010	0.00010

4. Implement the error back propagation algorithm for the $y = x(1 - x)$ neural network on slide 12 in "9-3. Neural Networks-EBP-3.pptx". You may change the number of hidden nodes, iterations, learning rate. Submit the followings:

- the code

- the final values of weights

- draw a graph for the output of neural network for $x = 0.00, 0.01, 0.02, 0.03, 0.04, \ldots, 1$

- the final values of weights

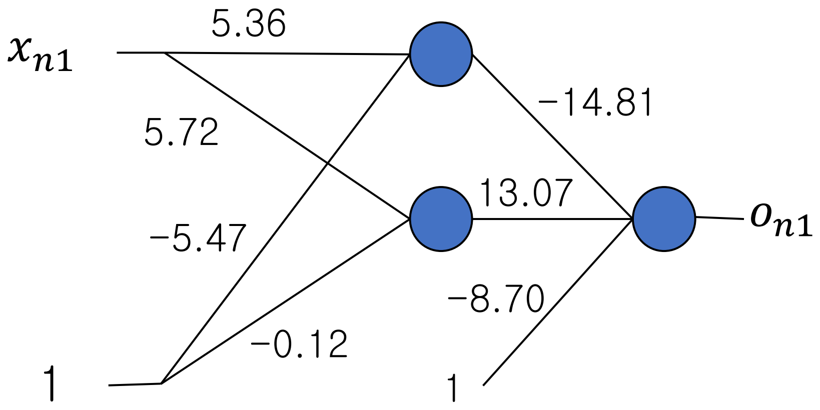
Hidden layer 1층, 노드 수는 3개로 구성하였다. 그 중 한 노드는 무조건 1을 주는 노드이기에 이 노드에 연결된 weight를 제외하면 final values of weights는 아래와 같이 된다.

w0[0][0] = 5.36, w0[0][1] = -5.47

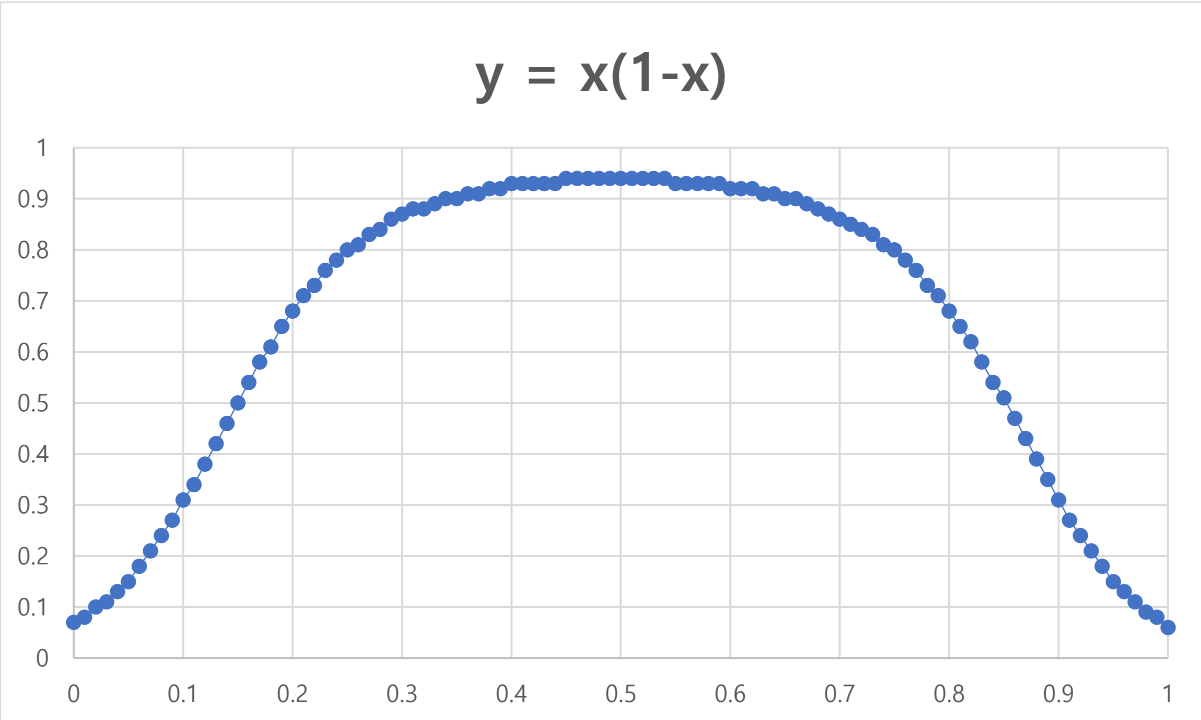
w0[1][0] = 5.72, w0[1][1] = -0.12

w1[0] = -14.81, w1[1] = 13.07, w1[2] = -8.70

이를 그림으로 표현하면 아래와 같이 되었다.



- draw a graph for the output of neural network for $x = 0.00, 0.01, 0.02, 0.03, 0.04, \dots, 1$



Iteration = 100000 으로 진행하였다. X=0, 1 부분에서 살짝 올라가긴 하지만 우리가 원하는 그래

프와 거의 일치하는 그래프가 나왔다. 해당 그래프의 값들은 아래 표와 같다.

0	0.07
0.01	0.08
0.02	0.1
0.03	0.11
0.04	0.13
0.05	0.15
0.06	0.18
0.07	0.21
0.08	0.24
0.09	0.27
0.1	0.31
0.11	0.34
0.12	0.38
0.13	0.42
0.14	0.46
0.15	0.5
0.16	0.54
0.17	0.58
0.18	0.61
0.19	0.65
0.2	0.68
0.21	0.71
0.22	0.73
0.23	0.76
0.24	0.78
0.25	0.8
0.26	0.81
0.27	0.83
0.28	0.84
0.29	0.86
0.3	0.87
0.31	0.88
0.32	0.88
0.33	0.89

0.34	0.9
0.35	0.9
0.36	0.91
0.37	0.91
0.38	0.92
0.39	0.92
0.4	0.93
0.41	0.93
0.42	0.93
0.43	0.93
0.44	0.93
0.45	0.94
0.46	0.94
0.47	0.94
0.48	0.94
0.49	0.94
0.5	0.94
0.51	0.94
0.52	0.94
0.53	0.94
0.54	0.94
0.55	0.93
0.56	0.93
0.57	0.93
0.58	0.93
0.59	0.93
0.6	0.92
0.61	0.92
0.62	0.92
0.63	0.91
0.64	0.91
0.65	0.9
0.66	0.9
0.67	0.89

0.68	0.88
0.69	0.87
0.7	0.86
0.71	0.85
0.72	0.84
0.73	0.83
0.74	0.81
0.75	0.8
0.76	0.78
0.77	0.76
0.78	0.73
0.79	0.71
0.8	0.68
0.81	0.65
0.82	0.62
0.83	0.58
0.84	0.54
0.85	0.51
0.86	0.47
0.87	0.43
0.88	0.39
0.89	0.35
0.9	0.31
0.91	0.27
0.92	0.24
0.93	0.21
0.94	0.18
0.95	0.15
0.96	0.13
0.97	0.11
0.98	0.09
0.99	0.08
1	0.06

- the code

아래는 작성된 코드 내용이다. 따로 첨부파일로도 압축되어 있다.

```
#include <iostream>
```

```
#include <cstdlib>

#include <ctime>

#include <cmath>


#define NUM_TDATA       11

using namespace std;


int main(void) {

        srand(time(NULL));

        int num;

        double w0[2][2] = {0}; //첫번째 weight 층

        double w1[3] = {0}; //두번째 weight 층

        double h[3] = {0, 0, 1}; // hidden layer의 output

        double o[NUM_TDATA] = {0}; //output layer. [4]는 test data가 4개이기 때문

        double net_x1[2] = {0}; // 첫번째 hidden layer의 sum

        double net_x2 = 0;   // output layer의 sum

        double e_w0[NUM_TDATA][2][2] = {0};   // 첫번째 weight층의 dE/dw ; [4]는 test data가 4개이기 때문

        double e_w1[NUM_TDATA][3] = {0}; // 두번째 weight층의 dE/dw

        double sum;

        double l_rate = 0.5; // learning rate

        double x[NUM_TDATA][2] = { {0.00, 1}, {0.10, 1}, {0.20, 1}, {0.30, 1}, {0.40, 1}, {0.50, 1}, {0.60, 1}, {0.70, 1}, {0.80, 1}, {0.90, 1}, {1.00, 1} }; // xor 학습 데이터

        double t[NUM_TDATA] = { 0.00, 0.36, 0.64, 0.84, 0.96, 1.00, 0.96, 0.84, 0.64, 0.36, 0.00}; // xor 학습 데이터의 답


        cout<< "Input the num of Iteration : ";

        cin>>num;
```

```
cout << endl;
cout << "Start Training..." << endl;
/* weight 랜덤하게
for(int i = 0; i < 2; i++) {
	for(int j = 0; j < 3; j++) {
		w0[i][j] = (double) ( rand()%2000 - 1000 ) / 1000;
	}
}
for(int i = 0; i < 3; i++) {
	w1[i] = (double) ( rand()%2000 - 1000 ) / 1000;
}
*/

//ppt에 주어진 weight값
w0[0][0] = -0.089; w0[0][1] = 0.028; w0[1][0] = 0.098; w0[1][1] = -0.07;
w1[0] = 0.056; w1[1] = 0.067; w1[2] = 0.016;


for(int i = 0; i < num; i++) {
	for(int j = 0; j < NUM_TDATA; j++) {
		//Step 1.
		net_x1[0] = x[j][0]*w0[0][0] + x[j][1]*w0[0][1];
		net_x1[1] = x[j][0]*w0[1][0] + x[j][1]*w0[1][1];
		h[0] = 1/(1+exp(-net_x1[0]));
		h[1] = 1/(1+exp(-net_x1[1]));
		h[2] = 1;
		net_x2 = h[0]*w1[0] + h[1]*w1[1] + h[2]*w1[2];
```

```
				o[j] = 1/(1+exp(-net_x2));

				//Step 2.
				for(int k = 0; k < 3; k++) {
					e_w1[j][k] = -(t[j] - o[j]) * o[j] * (1 - o[j]) * h[k];
					//cout << "e_w1" << j << k << " : " << e_w1[j][k] << endl;
				}
				for(int k = 0; k < 2; k++) {
					for(int l = 0; l < 2; l++) {
						e_w0[j][k][l] = -x[j][l] * h[k] * (1 - h[k]) * ( w1[k] * (t[j] - o[j]) * o[j] * (1 - o[j]) );
						//cout << "e_w0" << j << k << l << " : " << e_w0[j][k][l] << endl;
					}
				}
			}
			//Step 3
			for(int j = 0; j < 2; j++) {
				for(int k = 0; k < 2; k++) {
					sum = 0;
					for(int l = 0; l < NUM_TDATA; l++) {
						sum += e_w0[l][j][k];
					}
					w0[j][k] -= l_rate * sum;
				}
			}
			for(int j = 0; j < 3; j++) {
				sum = 0;
```

```
			for(int l = 0; l < NUM_TDATA; l++) {
				sum += e_w1[l][j];
			}
			w1[j] -= l_rate * sum;
		}
	}

	/*
	//Print
	cout << fixed;
	cout.precision(2);
	for(int i = 0; i < NUM_TDATA; i++) {
		cout << x[i][0] <<" : " << o[i] << endl;
	}
	cout<<endl;
	*/

	cout << "Training finish₩n" << endl;
	cout << "values of weight" << endl;
	for(int i = 0; i < 2; i++) {
		for(int j = 0; j < 2; j++) {
			cout << w0[i][j] << ' ';
		}
		cout << endl;
	}

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

```
		cout << w1[i] << ' ';
	}
	cout << endl;
	cout << endl;

	//Test
	double result;
	double v;
	cout << "Start test!" << endl;
	for(int i = 0; i <= 100; i++) {
		v = (double)i/100;

		//Test
		net_x1[0] = v*w0[0][0] + 1*w0[0][1];
		net_x1[1] = v*w0[1][0] + 1*w0[1][1];
		h[0] = 1/(1+exp(-net_x1[0]));
		h[1] = 1/(1+exp(-net_x1[1]));
		h[2] = 1;
		net_x2 = h[0]*w1[0] + h[1]*w1[1] + h[2]*w1[2];
		result = 1/(1+exp(-net_x2));

		cout << v <<"₩t" << result << endl;
	}
	cout << "Test Finished.";

	return 0;
}
```