**1. 서론**

1.1 연구주제

c언어를 활용하여 다층 퍼셉트론 모델과 오차역전파 알고리즘(back propagation)을 구현하고, 실제 데이터를 분류해보기

1.2 연구목적

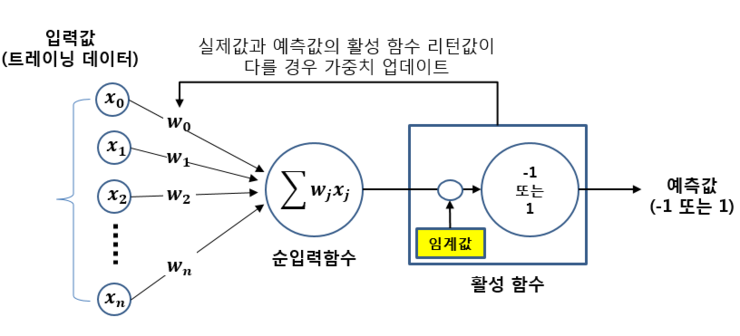
이전에 파이썬의 tensorflow라이브러리를 통해 ANN모델을 만들어 사용해본 적이있었는데, 단순히 코드를 입력해 모델을 쓰기만 할 뿐, 모델을 이해하고 쓴다는 느낌이 들지 않았었다. 그래서 ANN모델과 오차역전파 알고리즘을 다른 코드를 참고하지 않고 수식만 참고하여 다른 프로그래밍 언어로 직접 구현해보면서 ANN모델을 더 깊게 이해하려 하였다.

1.3 연구범위

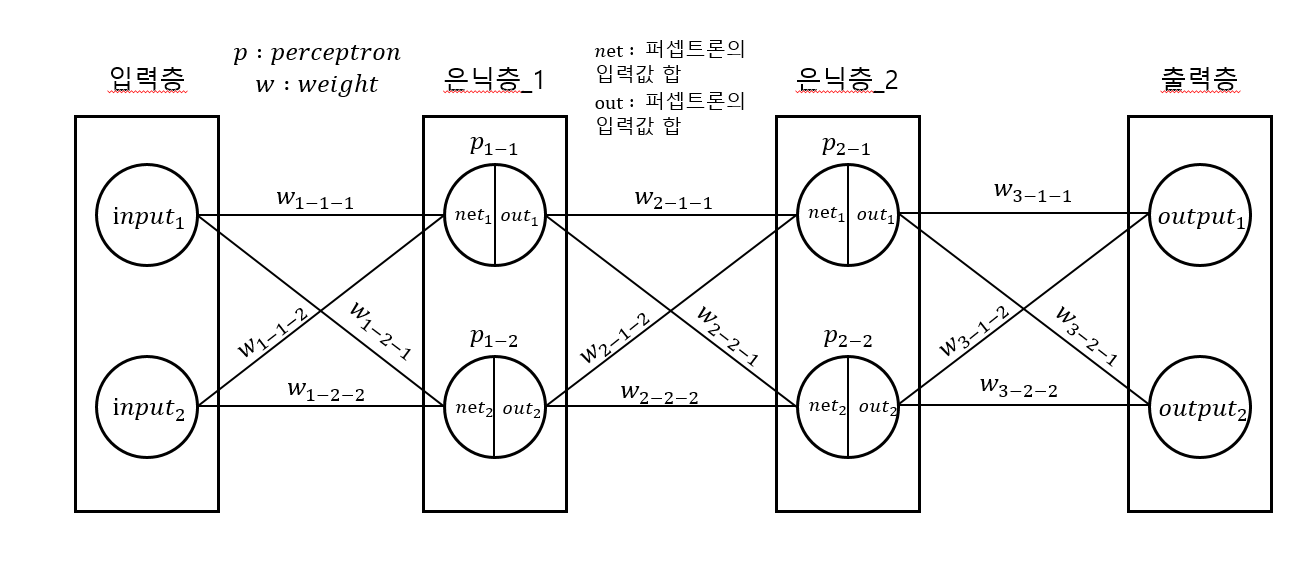
우선 오차역전파 알고리즘을 통해 xor문제를 해결해보고, MNIST데이터 셋중 하나인 붓꽃 데이터를 통해 학습시켜 붓꽃을 분류시켜보았다. 그리고 학습을 해보면서 여러 변수들을 조정하면서 모델의 학습과 성능에 어떤 영향을 미치는 지 알아보았다. 모든 모델은 C언어를 통하여 구현하였다.

**2. 연구 수행 내용**

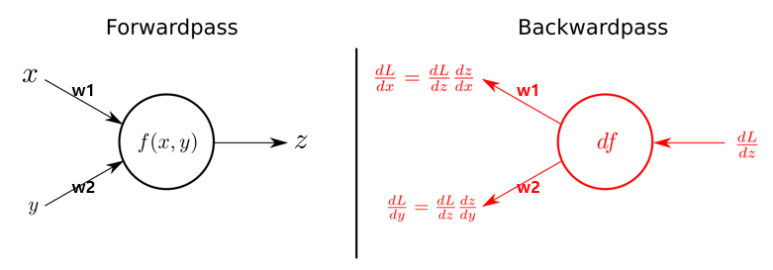
2.1 이론적 배경 및 선행 연구

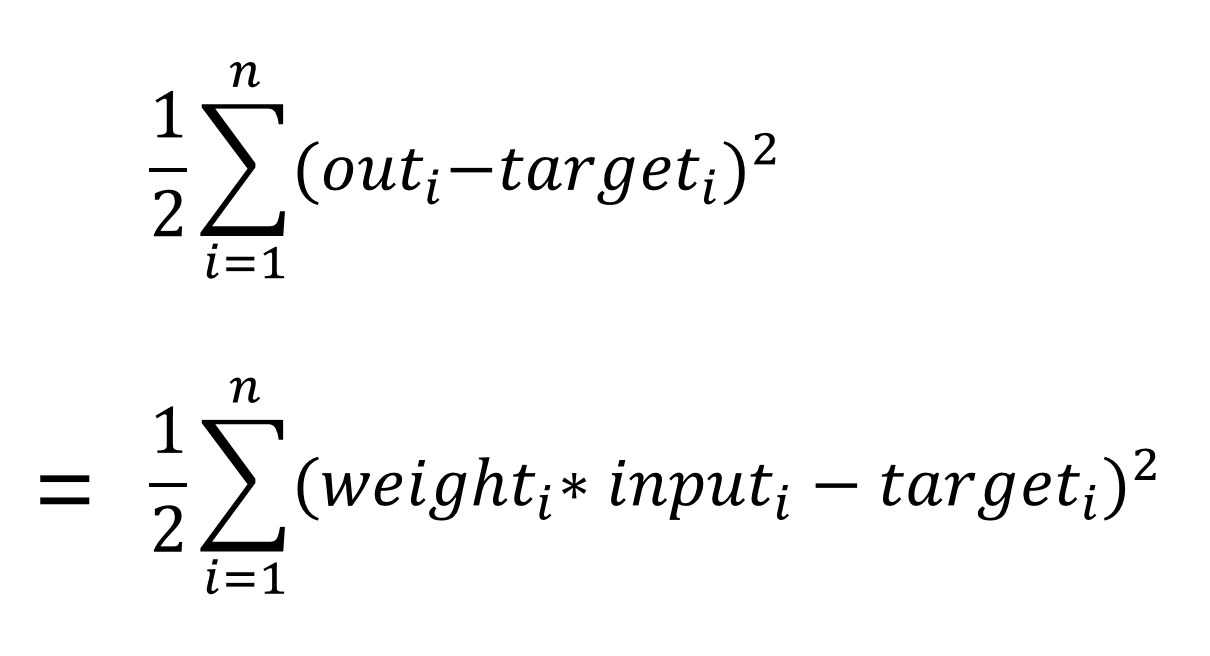


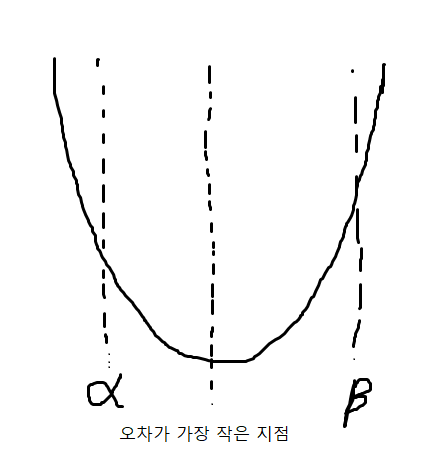
인공신경망에 대해서 알아보기 위해서는 인공신경망의 기초가 되는 퍼셉트론의 이해가 필요하다. 퍼셉트론이란 n개의 입력값에 n개의 가중치를 곱하여서 더한후, 활성화 함수(0에서 1사이의 값을 출력해낸다. 예로는 sigmoid(1/(1+e^-x))함수가 있다.)를 통해 한개의 값을 출력하는 구조를 뜻한다. (깔대기와 비슷하다.) 이 한개의 출력값은 n개의 입력값이 가진 특징들을 내재하고 있는 값이고, 이 값을 통해서 예측을 하는 것이다.

그리고 이런 퍼셉트론을 여러개가 있는 층을 또 여러개 쌓으면 이는 다층 퍼셉트론 모델(신경망 모델 중의 하나)가 된다. 다층 퍼셉트론 모델은 처음에 입력값을 받는 입력층, 예측값을 출력하는 출력층, 그리고 입력층과 출력층 사이에서 입력값의 특징을 추출해내는 은닉층이 있다. 은닉층은 앞서 말했던 퍼셉트론이 여러개 들어 있는 구조로, 만일 은닉층이 여러개 있다면 첫번째 은닉층의 출력값은 두 번째 은닉층의 입력값이 되는 구조로, 서로 연결돼 있다.

그리고 이런 다층 퍼셉트론이 입력값을 통해 적절한 출력값을 도출해 낼 수 있도록 퍼셉트론의 weight들을 수정해가는 학습과정을 통해 인공지능 모델은 성능을 올릴 수 있다. 보통 다층 퍼셉트론은 출력층에서 출력한 output의 값과 입력값에 따른 원하는 출력값 target(label이라고도 불린다.)의 오차를 w에 대해 편미분한 값과 학습률(learning rate)를 곱한 값을 w에 빼주면서 w를 업데이트 해가는 오차역전파(backward pass) 학습법을 통해 학습해나간다.



그리고 모델의 성능을 평가하는 데 사용되는 오차값은 loss또는 cost는 ½\*(모델의 추정치-실제답)^2의 꼴을 가지고 있다. 그래프로 그리면 이차함수의 꼴을 가지고 있게된다.

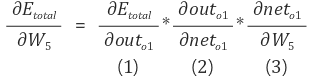


그럼 옆의 그래프를 보면서 왜 오차값을 미분한 값을 빼주는지 알아보자. 우리가 모델을 학습시키면서 최종적으로 는 오차가 가장 작은 지점, 이차함수의 대칭축에 가까워져야한다. 그리고 우리가 현재 α에 지점에 있다고 가정해보자. α지점은 out(출력값)이 target(목표값) 보다 작은 상태이다. α지점에서 미분을 하게되면 음의 값을 갖게 될 것이고, 이 값을 weight에서 빼주면 weight가 증가하여서 out(출력값)이 증가하여서 오차값이 줄어든다.

반대로 β지점은 out(출력값)이 target(목표값) 보다 큰 상태이다. β지점에서 미분을 하게 되면 양의 값이 나오게 되고, 이 값을 weight에서 빼주면 weight가 작아져 out(출력값)이 감소하게 되어 목표값에 가까워진다. 이런식으로 (오차를 미분한 값\*학습률)을 빼주는 학습을 반복하면서 모델은 점차 오차가 작은 지점에 가까워지는 것이다.

2.2 연구방법

오차역전파 수식들.











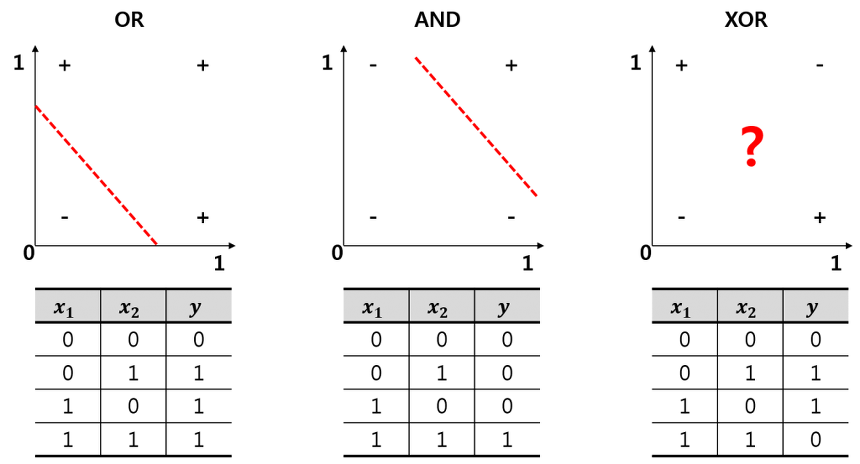








c언어를 통하여 다층퍼셉트론과 오차역전파를 구현하고 MINST의 붓꽃 데이터(<https://github.com/avianinfluenza/ANN-by-c/blob/main/iris(150).csv>)와 xor문제를 해결해본다.



2.3 활동 및 과정

c코드

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#define E 2.718281

#define \_CRT\_SECURE\_NO\_WARNINGS

double learning\_rate = 0.1;

double label[200][2] = {{0.01, 0.99}, {0.01, 0.99}, {0.01, 0.01}, {0.01, 0.01}}; //출력 데이터의 target값(정답)

int input\_n = 150, layer = 1;

struct perceptron{

double sub[200][4];

double delta[200];

double net[200];

double out[200];

double weight[4];

};

struct hidden\_layer{ //은닉층

int length = 2;

struct perceptron perceptron[2];

};

struct input\_layer{ //입력층

int n = 4;

int length = 2;

double input[200][10];

};

struct output\_layer{ //출력층

int length = 2;

struct perceptron perceptron[2];

};

double sigmoid(double input){ //활성화함수 시그모이드

return 1/(1+pow(E, -1\*input));

}

static struct hidden\_layer hidden\_layer[1];

static struct input\_layer input\_layer[1];

static struct output\_layer output\_layer[1];

void foward\_pass(void){ //입력값을 통해 출력값

for(int i = 0; i < input\_layer[0].n; i++){

for(int j = 0; j < hidden\_layer[0].length; j++){

hidden\_layer[0].perceptron[j].net[i] = 0;

for(int o = 0; o < input\_layer[0].length; o++){

hidden\_layer[0].perceptron[j].net[i] += input\_layer[0].input[i][o]\*hidden\_layer[0].perceptron[j].weight[o];

}

hidden\_layer[0].perceptron[j].out[i] = sigmoid(hidden\_layer[0].perceptron[j].net[i]);

}

}

for(int i = 0; i < input\_layer[0].n; i++){

for(int j = 0; j < output\_layer[0].length; j++){

output\_layer[0].perceptron[j].net[i] = 0;

for(int o = 0; o < hidden\_layer[layer-1].length; o++){

output\_layer[0].perceptron[j].net[i] += hidden\_layer[layer-1].perceptron[o].out[i]\*output\_layer[0].perceptron[j].weight[o];

}

output\_layer[0].perceptron[j].out[i] = sigmoid(output\_layer[0].perceptron[j].net[i]);

}

}

}

void back\_prapagation(void){ //오차역전파 : 출력값과 target값을 비교하여서 perceptron들의 weight를 조정

for(int i = 0; i < input\_layer[0].n; i++){

for(int j = 0; j < output\_layer[0].length; j++){

output\_layer[0].perceptron[j].delta[i] = (output\_layer[0].perceptron[j].out[i]-label[i][j])\*(1-output\_layer[0].perceptron[j].out[i])\*output\_layer[0].perceptron[j].out[i];

for(int k = 0; k < hidden\_layer[layer-1].length; k++){

output\_layer[0].perceptron[j].sub[i][k] = output\_layer[0].perceptron[j].delta[i]\*hidden\_layer[layer-1].perceptron[k].out[i];

}

}

}

for(int i = 0; i < input\_layer[0].n; i++){

for(int j = 0; j < hidden\_layer[layer-1].length; j++){

hidden\_layer[layer-1].perceptron[j].delta[i] = 0;

for(int k = 0; k < output\_layer[0].length; k++){

hidden\_layer[layer-1].perceptron[j].delta[i] += output\_layer[0].perceptron[k].delta[i]\*output\_layer[0].perceptron[k].weight[j];

}

hidden\_layer[layer-1].perceptron[j].delta[i] \*= hidden\_layer[layer-1].perceptron[j].out[i]\*(1-hidden\_layer[layer-1].perceptron[j].out[i]);

for(int k = 0; k < input\_layer[0].length; k++){

hidden\_layer[layer-1].perceptron[j].sub[i][k] = hidden\_layer[layer-1].perceptron[j].delta[i]\*input\_layer[layer-1].input[i][k];

}

}

}

for(int i = 0; i < input\_layer[0].n; i++){

for(int j = 0; j < hidden\_layer[layer-1].length; j++){

for(int k = 0; k < input\_layer[0].length; k++){

hidden\_layer[layer-1].perceptron[j].weight[k] -= learning\_rate\*hidden\_layer[layer-1].perceptron[j].sub[i][k];

}

}

}

for(int i = 0; i < input\_layer[0].n; i++){

for(int j = 0; j < output\_layer[0].length; j++){

for(int k = 0; k < hidden\_layer[layer-1].length; k++){

output\_layer[0].perceptron[j].weight[k] -= learning\_rate\*output\_layer[0].perceptron[j].sub[i][k];

}

}

}

}

double cost\_function(void){ //출력값과 타겟값(label)의 오차를 비교, 나중에 모델의 학습정도를 평가하는 기준으로 사용함

double temp = 0;

for(int i = 0; i < input\_layer[0].n; i++){

temp += (output\_layer[0].perceptron[0].out[i]-label[i][0])\*(output\_layer[0].perceptron[0].out[i]-label[i][0]);

temp += (output\_layer[0].perceptron[1].out[i]-label[i][1])\*(output\_layer[0].perceptron[1].out[i]-label[i][1]);

}

return temp;

}

int main(){

int temp;

printf("학습 데이터 :");

scanf("%d", &temp);

if(temp == 1){ //iris(붓꽃)데이터의 csv파일을 불러와 input\_layer에 넣음

char str\_tmp[1024];

FILE \*pFile = NULL;

int index = 0;

pFile = fopen("D:/ANN-by-c-main/ANN-by-c-main/iri", "r" );

if( pFile != NULL )

{

printf("csv file detected!\n");

fgets( str\_tmp, 1024, pFile );

while( !feof( pFile ) ){

fgets( str\_tmp, 1024, pFile );

char \*ptr = strtok(str\_tmp, ",");

int cnt = 0;

double num = atof(ptr);

ptr = strtok(NULL, ",");

while (ptr != NULL)

{

if(cnt == 4){

label[index][0] =0.01;

if(strcmp(ptr, "setosa\n") == 0){

label[index][1] = 0.01;

}

else if(strcmp(ptr, "versicolor\n") == 0){

label[index][1] = 0.55;

}

else{

label[index][1] = 0.99;

}

}

else{

double num = atof(ptr);

input\_layer[0].input[index][cnt] = num/10;

}

ptr = strtok(NULL, ",");

cnt++;

}

double temp = input\_layer[0].input[index][1];

input\_layer[0].input[index][1] = input\_layer[0].input[index][2];

input\_layer[0].input[index][2] = input\_layer[0].input[index][1];

index++;

}

}

fclose( pFile );

}

else{ //xor문제의 데이터를 input\_layer에 넣음

input\_layer[0].input[0][0] = 0.99;

input\_layer[0].input[0][1] = 0.01;

input\_layer[0].input[1][0] = 0.01;

input\_layer[0].input[1][1] = 0.99;

input\_layer[0].input[2][0] = 0.99;

input\_layer[0].input[2][1] = 0.99;

input\_layer[0].input[3][0] = 0.01;

input\_layer[0].input[3][1] = 0.01;

}

//random weight

srand((unsigned)time(NULL));

//first hidden layer

for(int i = 0; i < hidden\_layer[0].length; i++){

for(int j = 0 ; j < input\_layer[0].length; j++){

hidden\_layer[0].perceptron[i].weight[j] = ((rand()\*rand())%1000000)\*0.000001;

}

}

//leftover hidden layer

for(int k = 1; k < layer; k++){

for(int i = 0; i < hidden\_layer[k].length; i++){

for(int j = 0 ; j < hidden\_layer[k-1].length; j++){

hidden\_layer[k].perceptron[i].weight[j] = ((rand()\*rand())%1000000)\*0.000001;

}

}

}

//output layer

for(int i = 0; i < output\_layer[0].length; i++){

for(int j = 0 ; j < hidden\_layer[layer-1].length; j++){

output\_layer[0].perceptron[i].weight[j] = ((rand()\*rand())%1000000)\*0.000001;

}

}

printf("input\_n : ");

scanf("%d", &input\_n);

input\_layer[0].n = input\_n;

for(int i = 0; i < input\_n; i++){

printf("%lf %lf\n", input\_layer[0].input[i][0], input\_layer[0].input[i][1]);

printf("label[%d] : %lf %lf\n", i, label[i][0], label[i][1]);

}

printf("학습횟수 : ");

int cnt;

scanf("%d", &cnt);

for(temp = 1; temp <= cnt; temp++){

foward\_pass();

if(temp%1000 == 1){

printf("\*\*%d번째 학습\*\*\n", temp);

for(int i = 0; i < input\_layer[0].n; i++){

printf("out[%d][1] : %lf\n", i, output\_layer[0].perceptron[1].out[i]);

}

printf("cost : %lf\n", cost\_function());

}

save = cost\_function();

back\_prapagation();

}

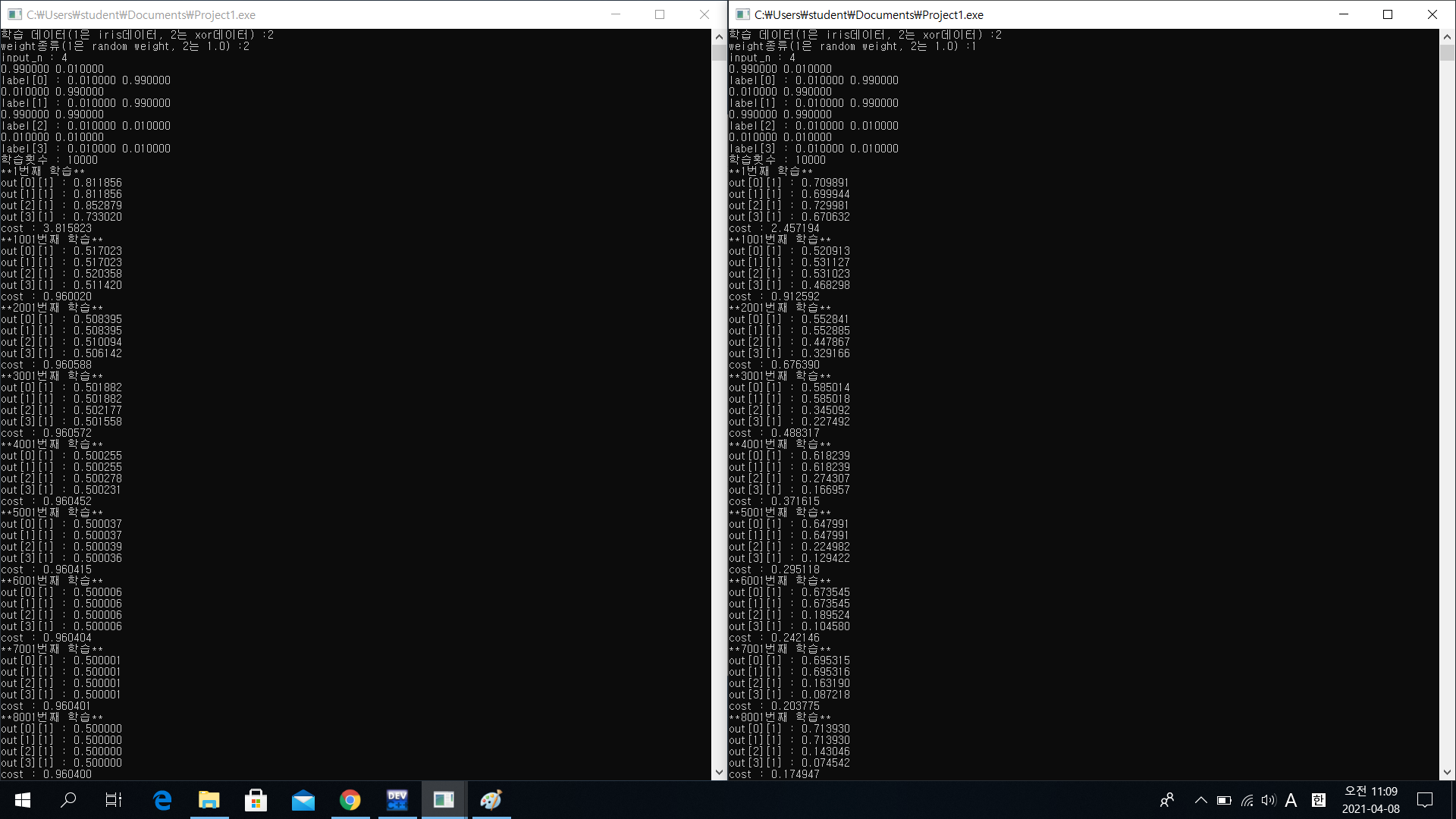
}

**3. 연구결과**

3.1 이론적 배경 및 선행 연구

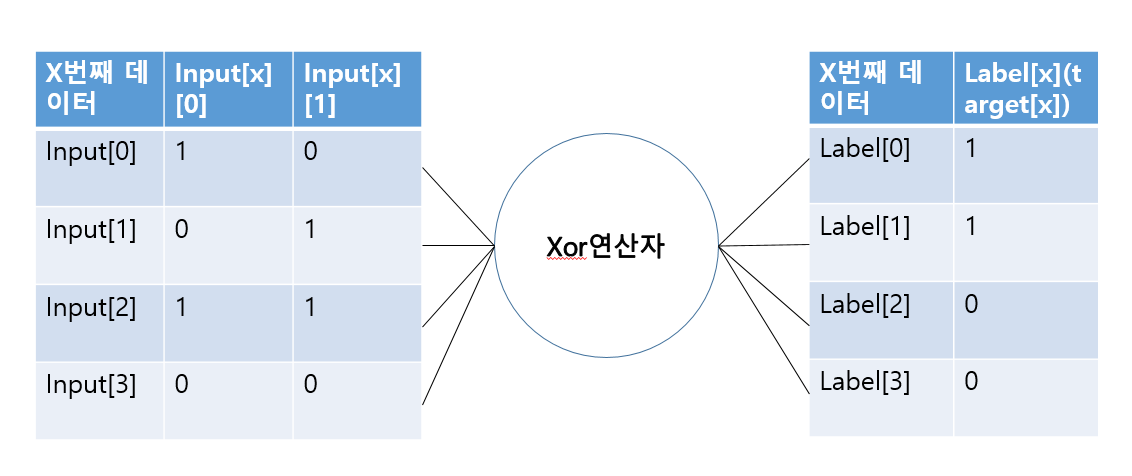
**Q1. 초기 weight를 random이 아닌 모두 같은 값으로 주면 안될까?**

**A1. 안돼!!!!!!!!!!!!!!!!!!!!!!!**

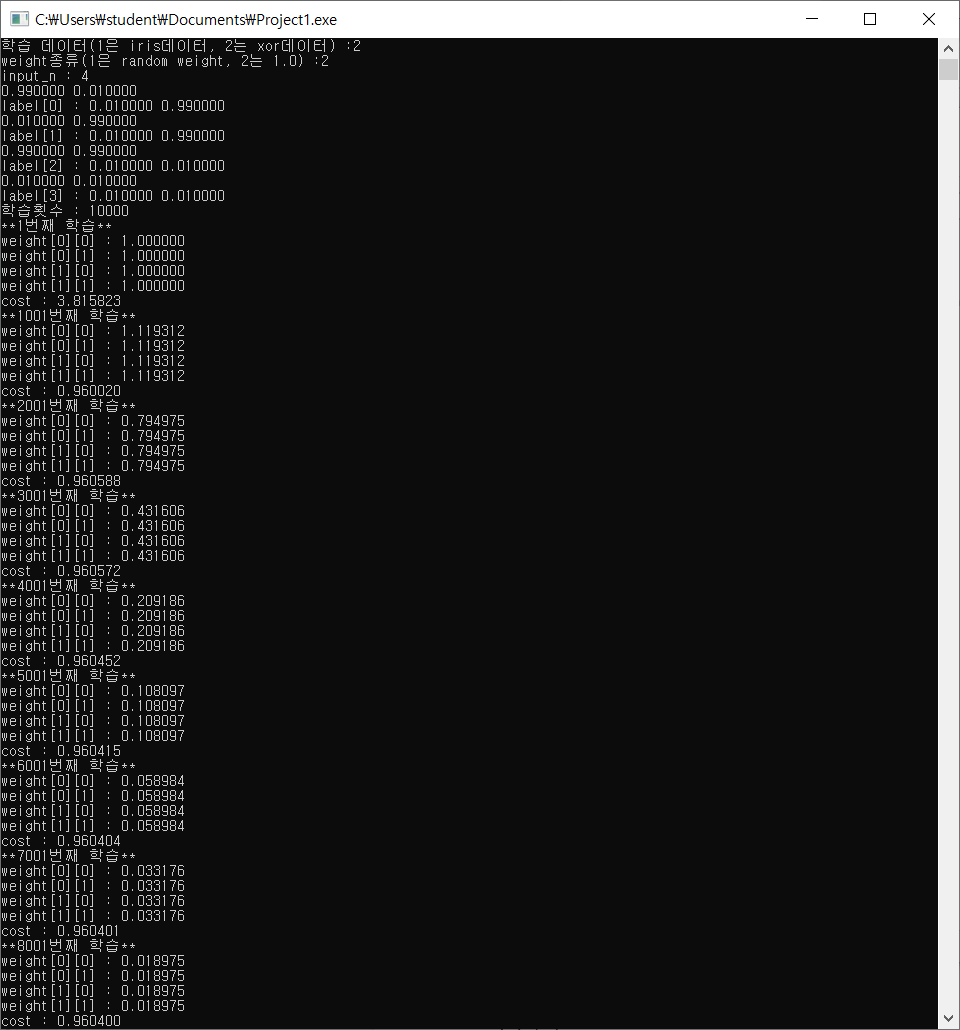


xor문제데이터를가지고 randomweight인 경우와 고정된 값을 주었을 때의 경우를 비교해보았다.(xor는 참과 참이거나 거짓과 거짓일 때는 거짓, 참과 거짓일때는 참을 내놓는 연산자이다.)그 결과 random으로 weight를 할당하였을 때는 올바르게 학습하며 출력값과 목표값의 오차인 cost(loss라고도 불리고, 인공지능의 성능을 평가한다.)가 줄어가는 모습을 보이지만, 고정된 값의 경우에는 오차가 학습을 거듭해도 줄지않는 모습을 보여준다.

이런 모습이 나타나는 이유를 추측해보았을 때, 입력데이터가 대칭적이라서 고정된 값이 주어졌을 때 학습이 불가능하다고 생각했다.



학습에 사용된 xor데이터는 이와같은 대칭적인 모습을 띠고 있다. 그러니 weight들에 빼지는 값들도 같을 것이고 모든 weight가 한개의 값으로 수렴할 것이다. 그래서 올바른 학습을 할 수 없는 것이다.

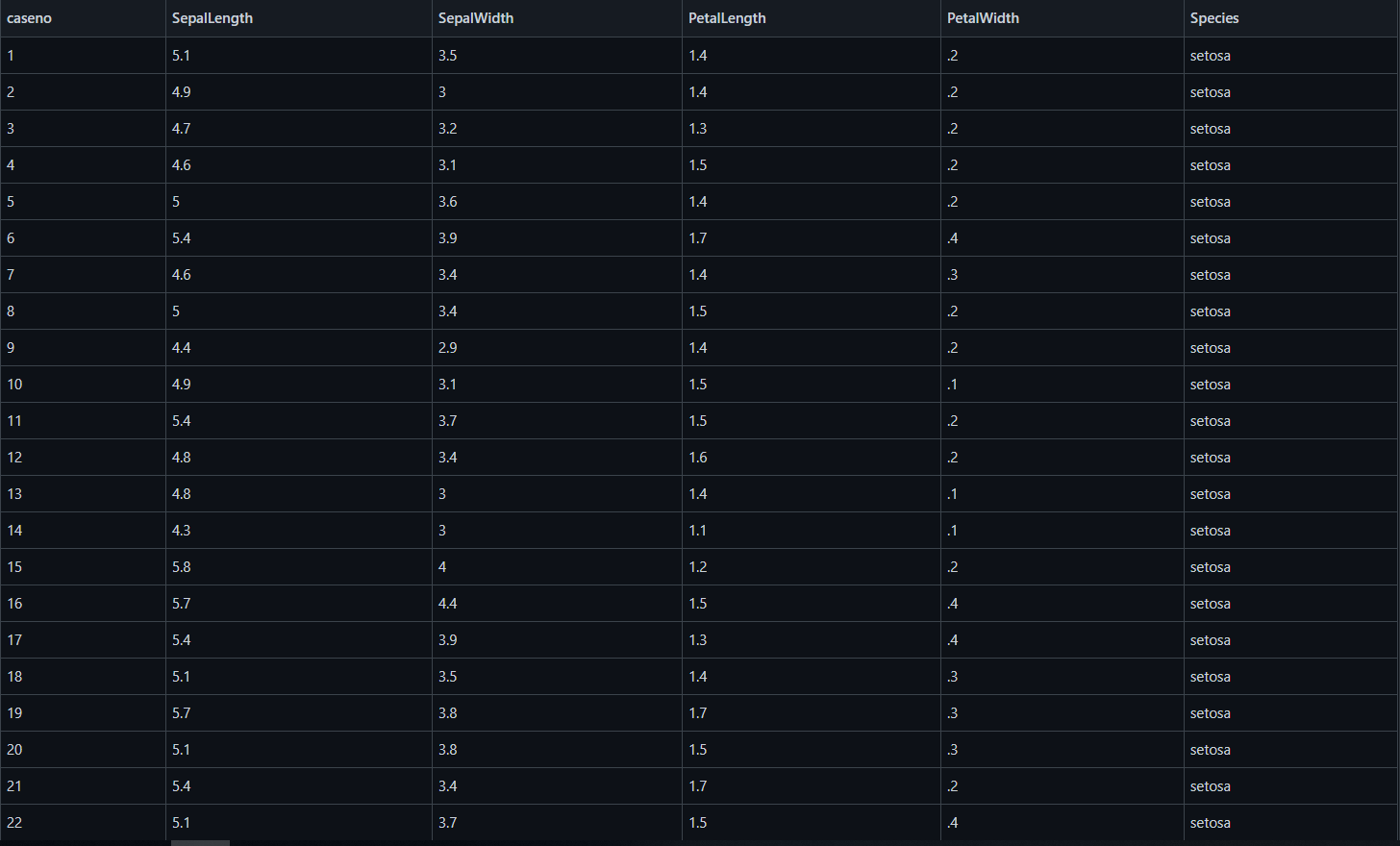


학습 과정에서 perceptron의 weight을 알아보니 예상대로 weight들이 모든 같은 값을 띠고 있다. xor의 데이터처럼 데이터의 값들이 대칭을 이루는 경향을 가지고 있다면, weight를 고정된 값으로 줬을 때 올바른 학습이 어려워 질 수 있다. 그러니 perceptron의 weight들은 random으로 주는 것이 좋을 것이다.

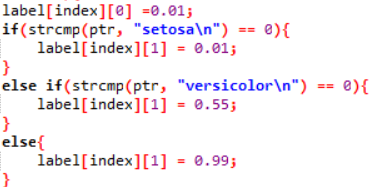
**Q2. 구현한 다중 퍼셉트론을 통해 실제 데이터를 학습할 수 있을까?**

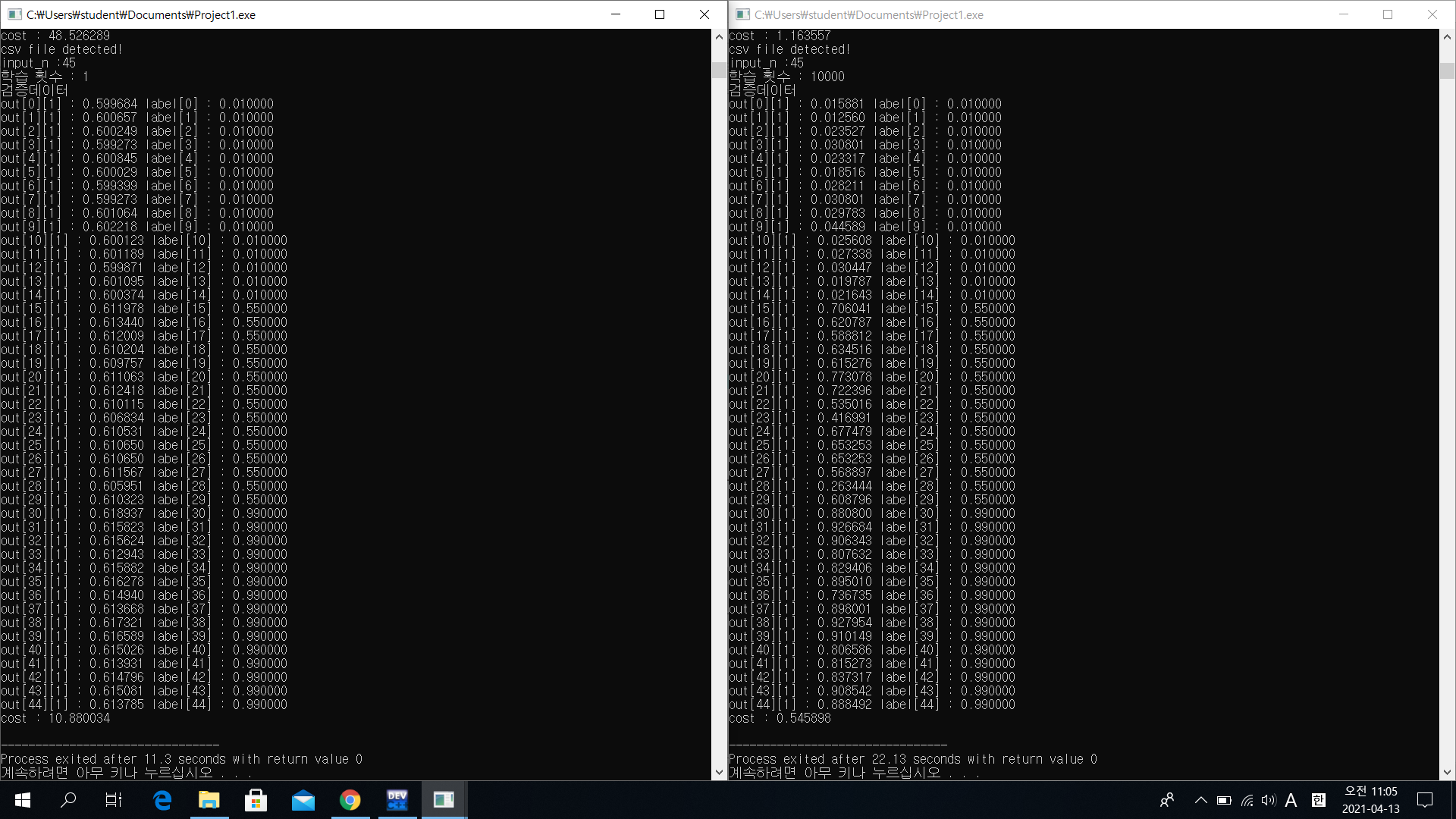
**A2. 돼!!!!!!!!!!!!!!!!!!!!!!!!!!!**

이제 인공신경망을 구현해 보았으니, 실제 데이터를 통해 학습을 시켜보겠다. 학습데이터는 MNIST에서 제공하는 붓꽃 데이터를 사용하였다.

****

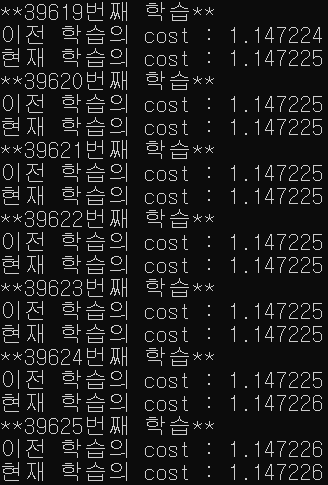
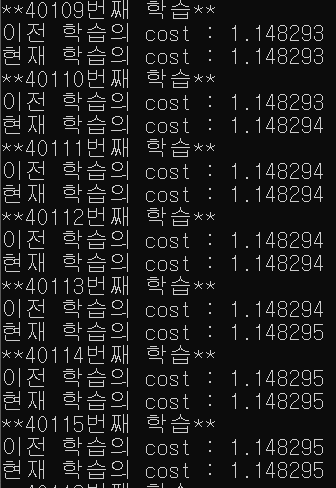
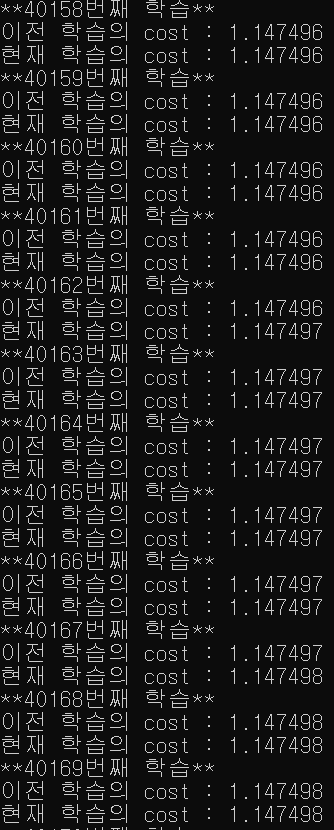
위의 그림과 같이 데이터는 꽃의 종에 따른 Sepal Length, Sepal Width, Petal Length, Petal Width가 주어진다. setosa종, versicolor종, virginica종 의 데이터가 각각 50개씩, 총 150개의 데이터가 있다. 나는 Sepal Length, Petal Length를 통해 꽃의 종을 예측해보겠다.

데이터를 불러온뒤, 종에 따라 옆의 사진과 같이 label값을 지정해 주었다. 

150개의 데이터 중 붓꽃의 종 별로 각 35개의 데이터를 학습데이터로 사용하고, 나머지 각 15개의 데이터로 학습한 모델의 성능을 검증해보았다.

그 결과 학습 이전에는 예측값들이 입력데이터와 관계 없이 전부 비슷한 값을 내놓았으나, 10000회 정도 학습을 하고나니, 다층 퍼셉트론이 입력값에서 특징과 타겟값(지금 상황에서는 꽃이 어떤 종인지에 대해)의 연관 관계를 추출해내 올바른 추측을 할 수 있도록 weight들을 수정하여서 학습에 사용되지 않은 새로운 데이터로도 종에 대한 올바른 추측을 할 수 있게 되었다. ***한마디로, 모델이 올바르게 학습을 해내었다!***

그런데, 모델을 학습시키면서 이상한 점을 발견했다.

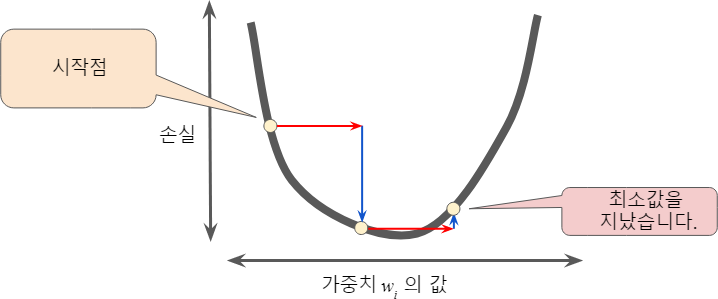
초기 weight의 값을 난수로 주어서 할 때마다 다르긴하지만 대략 40000번째의 학습에서 부터 모델의 cost(모델의 성능)이 이전 모델에 비해 높아지는(성능이 떨어지는) 모습을 보였다. 어째서 모델은 학습을 거치고도 퇴행한걸까?

**Q3. 모델이 학습을 거치고도 왜 성능이 떨어질까?**

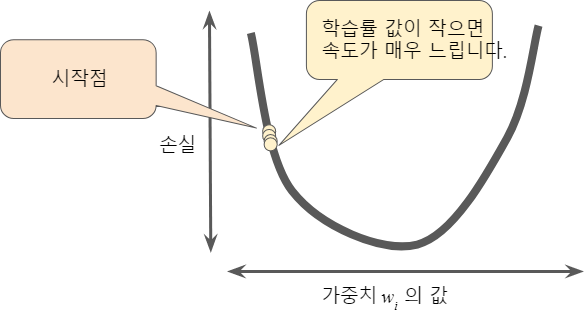
**A3. 몰라!!!!!!!!!!!!!!!!!!!!!!!!!!!**

솔직히 말하자면 왜 이런 지 이유는 정확히 모르겠다. 단순히 내 코드가 잘못된 것일 수 도있고, 학습을 거칠수록 weight에 접근하는 값의 절댓값이 작아지면서 실수의 정확도가 낮아지면서 난 오류일 수도 있다. 그 외에도 내가 모르는 부분에서 오류가 났을 수도 있

다.

문제의 이유는 모르지만, 문제의 해결법은 알고있다. 바로 모델이 오차역전파를 할 때 편미분한 값에 곱해주는 값인 **학습률(learning rate)**을 조절하는 것이다. 학습률의 크기는 모델의 성능과 학습시간에 큰 영향을 미치는 변수이다.

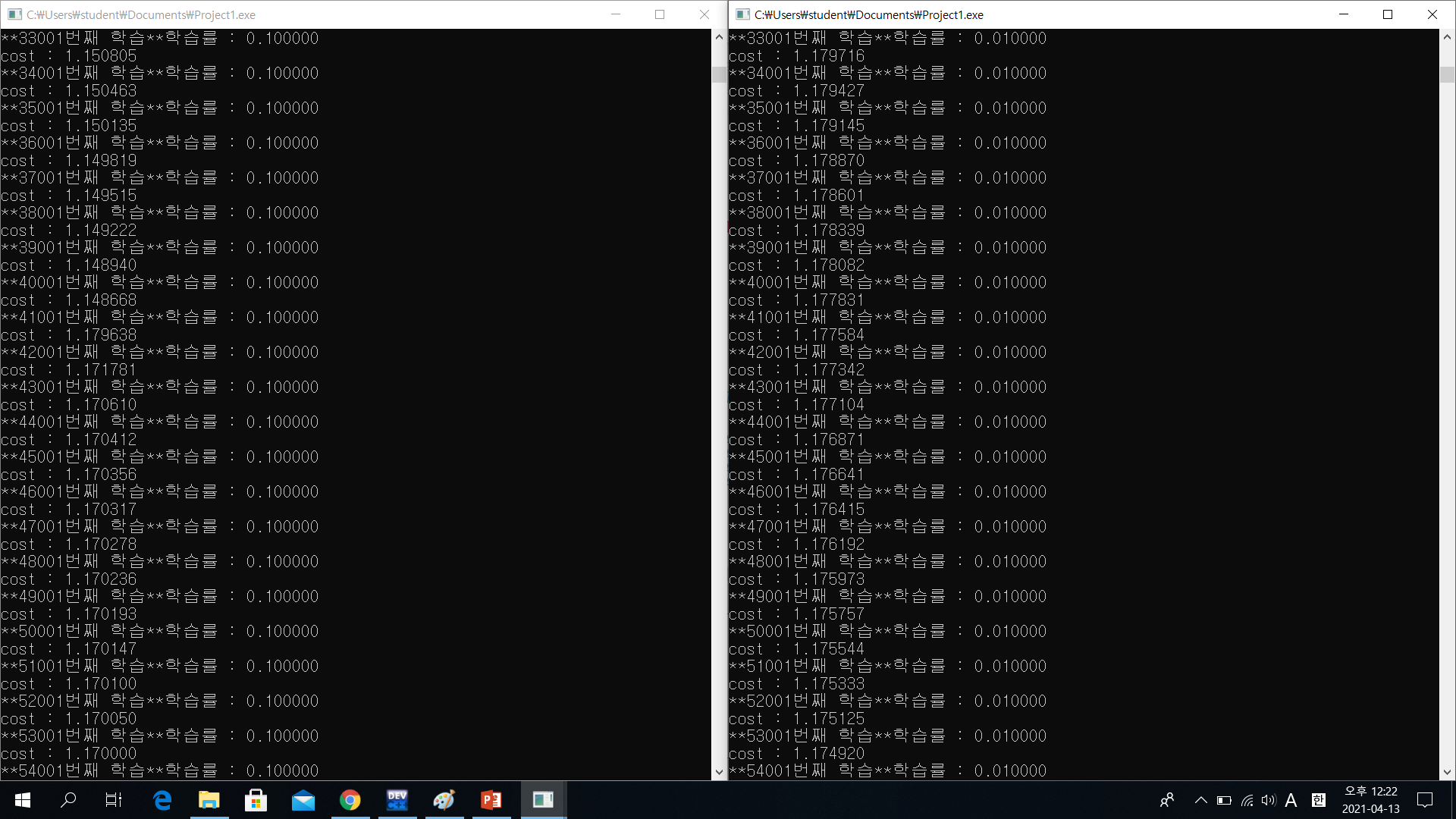
만일 학습률이 크다면 weight에서 빼주는 값이 큼직큼직 하기 때문에 모델이 학습을 마치는 시간(cost가 비슷한 값으로 수렴하는데 걸리는 시간)적게 걸리지만, 빼주는 값이 너무 커서 오차가 가장 작은 지점 근처로 수렴하지못해 모델의 정확도가 떨어질 수 있다.



반면 학습률이 지나치게 작다면 어떻게 될까? weight에서 빼주는 값이 매우 작아져, 오차가 가장 작은 지점에 보다 더 가까워 질 수 있지만, 학습을 마치는 시간이 지나치게 길어질 수 있게 된다.

이와 같은 이유로 모델을 학습시킬때는 자료의 갯수, 특성에 따라 적절한 학습률을 찾아내는

것이 모델의 성능에 중요한 요인이 된다.

왼쪽의 실행창은 기존의 학습진행할 때와 같이 learning rate = 0.1로 설정하고 학습을 진행했을 때의 경우, 오른쪽의 실행창은 learning rate = 0.01로 설정했을 때의 경우이다. 

learning rate를 크게 했을때는 여전히 약 40000 번째 학습 때 cost가 증가하는 모습을 보이지만, learning rate를 작게 했을때는 학습률이 계속 감소하는 모습을 보이고있다. 그리고 학습 횟수를 더 늘리면 learning rate가 작을 때 cost가 더 작은 값으로 수렴해가는 모습을 보였다.

**4. 결론 및 제언**

4.1 결론

다층 퍼셉트론 구조에서는 퍼셉트론의 초기 가중치를 랜덤한 값으로 초기화 해주어야하고, 다중 퍼셉트론 모델을 실제 데이터에도 적용해 사용할 수 있으며, 학습률을 더 작게 할 수록 학습에 더 많은 시간이 걸리지만, 학습횟수가 충분하다면 더 좋은 성능의 모델로 학습시킬 수 있다.

4.2 제언

c언어를 통해 퍼셉트론을 구현해보고 나니 전보다 더 많은 것을 알게 되기도 했지만, 궁금한 점들도 더 많이 생긴 것 같다. 나중에는 전문가에게 제대로 배우면서 인공지능의 원리를 완전히 이해하고 더 깊게 탐구해보고 싶다. 그리고 이번에는 인공 신경망 모델 중 가장 간단하며 원시적인 다중 퍼셉트론을 구현했는데 다음에 시간이 생긴다면 CNN모델 같이 실전에서 자주 사용되는 모델들도 직접 구현해보고 싶다.

코드를 짜면서 어려운 점들도 많았고, 포기하고 싶을 때도 많았지만 스스로 탐구하고 이유를 추론해가는 과정이 너무 재미있는 시간이였던 것 같다.

**5. 참고 문헌**

<https://goofcode.github.io/back-propagation>

<https://bskyvision.com/718>