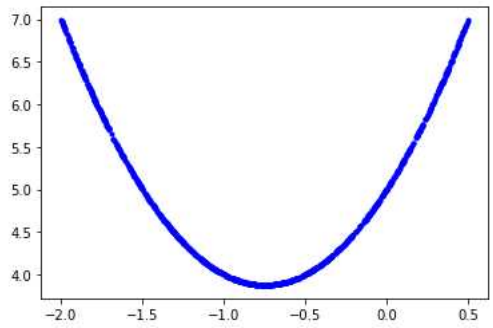
Chapter 04
인공지능의 딥러닝 알고리즘

01 DNN 라이브러리 사용해 보기

우리는 1장에서 구글 코랩 상에서 텐서플로우를 이용하여 2차 함수 근사를 수행해 보았습니다. 여기서는 미리 구현된 DNN 인공 신경망 라이브러리를 이용하여 2차 함수를 ESP32 아두이노 상에서 근사해 봅니다. 구글 코랩 상에서의 수행 속도보다는 늦지만 같은 결과를 내는 것을 확인해 볼 수 있습니다. ESP32 아두이노 상에서 동작하는 DNN 인공 신경망은 뒤에서 독자 여러분이 직접 구현해 보게 됩니다. 또 이 DNN을 이용하여 아두이노 인공지능 드론을 학습시키고 구동시켜 봅니다.

01 2차 함수 근사해 보기

이제 소스와 함께 제공되는 DNN 인공 신경망 라이브러리를 이용하여 다음과 같은 2차 함수를 근사해 봅니다.



$$y = 2x^2 + 3x + 5 \ (-2 \le x \le 0.5)$$

x 좌표의 범위는 -2에서 0.5까지입니다.

2차 함수 그리기

1. 제공되는 소스에서 다음과 같이 [myai_test_1] 파일을 엽니다.

myai_test_1

```
01 #include "myplot.h"
02
03 const long RANDMAX = 2147483647;
04
05 const int NUM_SAMPLES = 1000;
06 const double MIN_X = -2;
07 const double MAX_X = 0.5;
08
09 double xs[NUM_SAMPLES];
10 double ys[NUM_SAMPLES];
11
12 double f(double x) {
13         return 2*x*x + 3*x + 5;
14 }
15
16 void setup() {
17
18         Serial.begin(115200);
19         lcdSetup();
20
21         randomSeed(time(NULL));
22
23         for(int n=0;n<NUM_SAMPLES;n++) {
24                 xs[n] = (double)random(RANDMAX)/RANDMAX*(MAX_X-MIN_X)+MIN_X;
25                 if(n<5) Serial.println(xs[n]);
26         }
27
28         for(int n=0;n<NUM_SAMPLES;n++) {
29                 ys[n] = f(xs[n]);
30                 if(n<5) Serial.println(ys[n]);
31         }
32
33         lcdSetPlotArea(xs, ys, NUM_SAMPLES);
34
35         lcdClear(TFT_BLACK);
36         lcdPlot(xs, ys, TFT_GREEN, NUM_SAMPLES);
37
```

```
38 }
39
40 void loop() {
41
42 }
```

01 : myplot.h 파일을 포함합니다. myplot.h 파일에는 LCD에 그래프를 그리기 위한 함수가 선언되어 있습니다.
02 : RANDMAX 상수를 정의하고 32비트 양수의 최대값으로 설정합니다. RANDMAX 상수는 24줄에서 random 함수를 호출하여 생성하는 임의 숫자 범위를 설정합니다.
05 : NUM_SAMPLES 상수를 생성한 후, 1000으로 초기화합니다. NUM_SAMPLES 상수는 생성할 난수 데이터의 개수 값을 나타냅니다.
06 : MIN_X 상수를 생성한 후, -2로 초기화합니다. MIN_X는 x 좌표 범위의 최소값을 나타냅니다.
07 : MAX_X 상수를 생성한 후, 0.5로 초기화합니다. MAX_X는 x 좌표 범위의 최대값을 나타냅니다.
09 : xs 실수 배열을 선언합니다. 배열 항목의 개수는 NUM_SAMPLES입니다.
10 : ys 실수 배열을 선언합니다. 배열 항목의 개수는 NUM_SAMPLES입니다.
12~14 : f 함수를 정의합니다.
13 : 신경망 학습에 사용할 함수를 정의합니다.
18 : 시리얼을 초기화합니다.
19 : lcdSetup 함수를 호출하여 LCD를 초기화합니다.
21 : randomSeed 함수를 호출하여 난수 생성기 모듈을 초기화합니다.
23~26 : random 함수를 호출하여 (-2, 0.5) 범위에서 NUM_SAMPLES 만큼의 임의 실수 값을 추출하여 xs 배열에 저장합니다.
25 : Serial.println 함수를 호출하여 xs에 저장된 값 중, 앞에서 5개까지 출력합니다.
28~31 : 다음 식을 이용하여 추출된 x 값에 해당하는 y 값을 얻어내어 ys 변수에 저장합니다. y 값도 NUM_SAMPLES 개수만큼 추출됩니다.

$$y = 2x^2 + 3x + 5$$

29 : Serial.println 함수를 호출하여 ys에 저장된 값 중, 앞에서 5개까지 출력합니다.
33 : lcdSetPlotArea 함수를 호출하여 화면에 그릴 xs, ys의 영역을 설정합니다. 이 때, NUM_SAMPLES 값도 함수로 넘겨줍니다.
35 : lcdClear 함수를 호출하여 화면을 검정색 채웁니다.
36 : lcdPlot 함수를 호출하여 xs, ys 좌표 값에 맞추어 그래프를 그립니다. 그래프의 색깔은 초록색으로 그립니다. 이 때, NUM_SAMPLES 값도 함수로 넘겨줍니다.

2. [툴] 메뉴를 이용하여 보드, 포트를 다음과 같이 선택합니다.

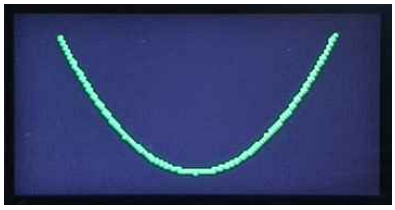
3. 업로드를 수행합니다.

4. 다음은 실행 결과 화면입니다.

```
-0.99
-0.09
-0.41
0.25
-1.26
3.99
4.76
4.11
5.87
4.40
```

❶ xs에 저장된 값 중 앞에서 5개까지 출력 결과입니다. 이 값은 실행할 때마다 달라집니다.
❷ ys에 저장된 값 중 앞에서 5개까지 출력 결과입니다. 이 값은 실행할 때마다 달라집니다.

다음과 같이 LCD 화면에 그려진 그래프도 확인합니다.



$y = 2x^2 + 3x + 5$ 함수의 (-2, 0.5) 범위에서의 그래프입니다.

실제 데이터 생성하기

이번엔 y값을 일정한 범위에서 위아래로 흩뜨려 실제 데이터에 가깝게 만들어 봅니다. 이 과정은 y값에 잡음을 섞어 실제 데이터에 가깝게 만드는 과정입니다.

1. 제공되는 소스에서 다음과 같이 [myai_test_2.py] 파일을 엽니다.

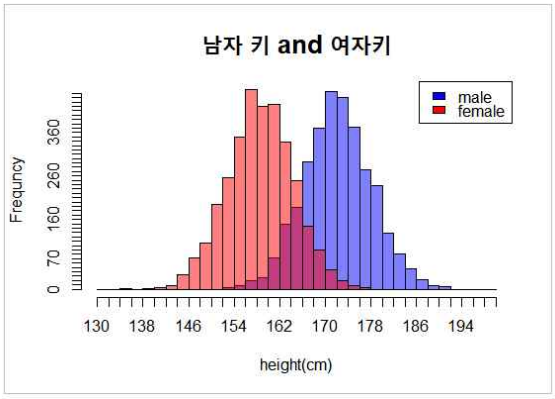
myai_test_2

```
01 #include "myplot.h"
02 #include "myrandn.h"
03
04 const long RANDMAX = 2147483647;
05
06 const int NUM_SAMPLES = 1000;
07 const double MIN_X = -2;
08 const double MAX_X = 0.5;
09
10 double xs[NUM_SAMPLES];
11 double ys[NUM_SAMPLES];
12
13 double f(double x) {
14         return 2*x*x + 3*x + 5;
15 }
16
17 void setup() {
18
19         Serial.begin(115200);
20         lcdSetup();
21
22         randomSeed(time(NULL));
23
24         for(int n=0;n<NUM_SAMPLES;n++) {
25                 xs[n] = (double)random(RANDMAX)/RANDMAX*(MAX_X - MIN_X)+MIN_X;
26                 if(n<5) Serial.println(xs[n]);
27         }
28
29         for(int n=0;n<NUM_SAMPLES;n++) {
30                 ys[n] = f(xs[n]);
31                 if(n<5) Serial.println(ys[n]);
32         }
33
34         lcdSetPlotArea(xs, ys, NUM_SAMPLES);
35
36         lcdClear(TFT_BLACK);
```

```
37        lcdPlot(xs, ys, TFT_GREEN, NUM_SAMPLES);
38
39        for(int n=0;n<NUM_SAMPLES;n++) {
40                ys[n] += 0.1*randn();
41        }
42
43        lcdClear(TFT_BLACK);
44        lcdPlot(xs, ys, TFT_YELLOW, NUM_SAMPLES);
45
46 }
47
48 void loop() {
49
50 }
```

02 : myrandn.h 파일을 포함합니다. myrandn.h 파일은 40줄에서 사용할 randn 함수가 선언되어 있습니다.

39~41 : randn 함수를 호출하여 정규분포에 맞춰 임의 숫자를 NUM_SAMPLES의 개수만큼 생성합니다. 정규분포는 가우스분포라고도 하며, 종 모양과 같은 형태의 자연적인 분포 곡선입니다. 예를 들어, 키의 분포나 체중의 분포와 같이 자연적인 분포를 의미합니다.



<출처 : https://hsm-edu.tistory.com/1015>

생성된 숫자에 0.1을 곱해 ys에 더해줍니다. 이렇게 하면 ys값은 원래 값을 기준으로 상하로 퍼진 형태의 자연스런 값을 갖게 됩니다.

43 : lcdClear 함수를 호출하여 화면을 검정색으로 채웁니다.

44: lcdPlot 함수를 호출하여 xs, ys 좌표 값에 맞추어 그래프를 그립니다. 그래프의 색깔은 노란색으로 그립니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

그래프의 모양이 원래 모양에서 상하로 퍼진 형태로 나타나게 됩니다. 여기서 생성된 데이터는 인공 신경망 학습에 사용되며 원래 곡선에 가까운 근사 곡선을 생성하는 인공 신경망 함수를 만들게 됩니다.

훈련, 실험 데이터 분리하기

여기서는 앞에서 생성한 x, y 데이터를 훈련 데이터와 실험 데이터로 분리해 봅니다. 훈련 데이터는 인공 신경망을 학습시키는데 사용하는 데이터이며, 실험 데이터는 학습이 잘 되었는지 확인하는 데이터로 사용합니다.

1. 제공되는 소스에서 다음과 같이 [myai_test_3] 파일을 엽니다.

myai_test_3

```
01 #include "myplot.h"
02 #include "myrandn.h"
03
04 const long RANDMAX = 2147483647;
05
06 const int NUM_SAMPLES = 1000;
07 const double MIN_X = -2;
08 const double MAX_X = 0.5;
09
10 double xs[NUM_SAMPLES];
11 double ys[NUM_SAMPLES];
12
13 double f(double x) {
14   return 2*x*x + 3*x + 5;
15 }
16
17 void setup() {
18
19       Serial.begin(115200);
20       lcdSetup();
21
22       randomSeed(time(NULL));
23
24       for(int n=0;n<NUM_SAMPLES;n++) {
```

```
25                  xs[n] = (double)random(RANDMAX)/RANDMAX*(MAX_X - MIN_X)+MIN_X;
26                  if(n<5) Serial.println(xs[n]);
27          }
28
29          for(int n=0;n<NUM_SAMPLES;n++) {
30                  ys[n] = f(xs[n]);
31                  if(n<5) Serial.println(ys[n]);
32          }
33
34          lcdSetPlotArea(xs, ys, NUM_SAMPLES);
35
36          lcdClear(TFT_BLACK);
37          lcdPlot(xs, ys, TFT_GREEN, NUM_SAMPLES);
38
39          for(int n=0;n<NUM_SAMPLES;n++) {
40                  ys[n] += 0.1*randn();
41          }
42
43          lcdClear(TFT_BLACK);
44          lcdPlot(xs, ys, TFT_YELLOW, NUM_SAMPLES);
45
46          const int NUM_LEARNING = 0.8*NUM_SAMPLES;
47          const int NUM_THINKING = NUM_SAMPLES - NUM_LEARNING;
48
49          double * x_train = xs, * x_test = xs+NUM_LEARNING;
50          double * y_train = ys, * y_test = ys+NUM_LEARNING;
51
52          lcdClear(TFT_BLACK);
53          lcdPlot(x_train, y_train, TFT_GREEN, NUM_LEARNING);
54          lcdPlot(x_test, y_test, TFT_RED, NUM_THINKING);
55
56 }
57
58 void loop() {
59
60 }
```

46 : NUM_SAMPLES에 0.8을 곱한 후, NUM_LEARNING 정수 상수에 할당합니다. 현재 예제의 경우 NUM_LEARNING 상수는 800의 값을 가집니다. 1000개의 x, y 데이터 값 중 800개는 훈련 데이터로, 40개는 실험 데이터로 사용합니다.

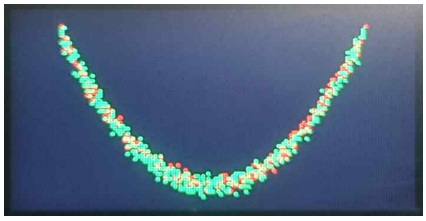
47 : NUM_THINKING 정수 상수를 선언한 후, NUM_SMPLES에서 NUM_LEARNING을 뺀 값으로 초기화합니다. NUM_THINKING 상수는 실험 데이터의 개수를 나타냅니다.

49 : 1000개의 값을 가진 xs를 800개, 200개로 나누어 각각 x_train, x_test 실수 주소 변수에 할당합니다. x_train 실수 주소 변수는 xs의 시작 위치를 가리키고, x_test 실수 주소 변수는 xs의 800번째 항목의 위치를 가리킵니다.

50 : 1000개의 값을 가진 ys를 800개, 200개로 나누어 각각 y_train, y_test 실수 주소 변수에 할당합니다. y_train 실수 주소 변수는 ys의 시작 위치를 가리키고, y_test 실수 주소 변수는 ys의 800번째 항목의 위치를 가리킵니다.
52 : lcdClear 함수를 호출하여 화면을 검정색으로 채웁니다.
53 : lcdPlot 함수를 호출하여 x_train, y_train 좌표 값에 맞추어 그래프를 그립니다. 그래프의 색깔은 초록색으로 그립니다. NUM_LEARNING은 화면에 그릴 좌표의 개수를 나타냅니다.
54 : lcdPlot 함수를 호출하여 x_test, y_test 좌표 값에 맞추어 그래프를 그립니다. 그래프의 색깔은 빨간색으로 그립니다. NUM_THINKING은 화면에 그릴 좌표의 개수를 나타냅니다.

2. 업로드를 수행합니다.

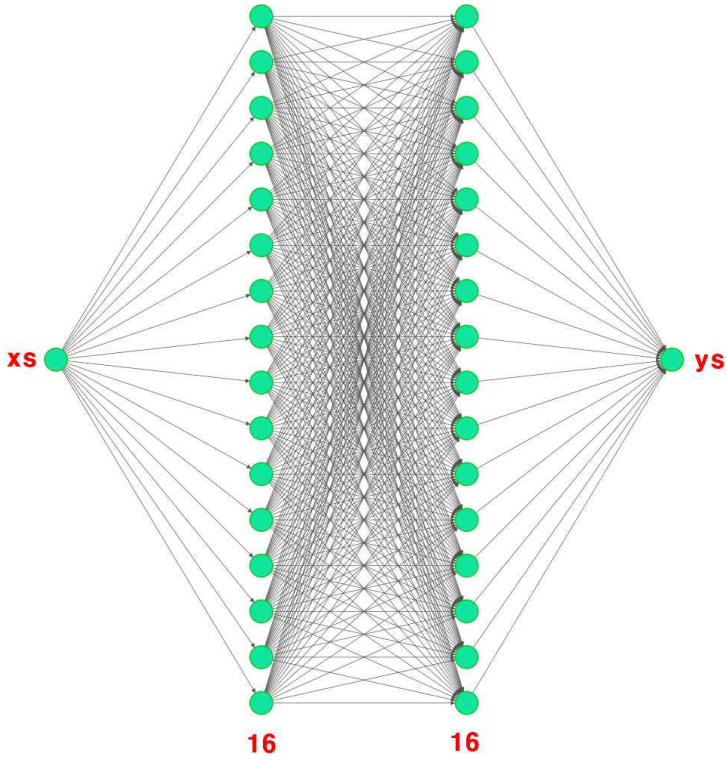
3. 다음은 실행 결과 화면입니다.



초록색 점은 x_train, y_train의 분포를 나타내며, 빨간색 점은 x_test, y_test의 분포를 나타냅니다. x_train, y_train 데이터는 인공 신경망 학습에 사용되며 원래 곡선에 가까운 근사 곡선을 생성하는 인공 신경망 함수를 만들게 됩니다. x_test, y_test 데이터는 학습이 끝난 인공 신경망 함수를 시험하는데 사용합니다.

인공 신경망 구성하고 학습시키기

이번엔 인공 신경망 함수를 구성하고, 훈련 데이터를 이용하여 학습 시킨 후, 시험 데이터를 이용하여 예측을 수행하고 그래프를 그려봅니다. 여기서는 다음과 같은 모양의 인공 신경망을 구성합니다. 입력 층 xs, 출력 층 ys 사이에 단위 인공 신경 16개로 구성된 은닉 층 2개를 추가하여 인공 신경망을 구성합니다.



1. 제공되는 소스에서 다음과 같이 [myai_test_4] 파일을 엽니다.

myai_test_4

```
01 #include "myplot.h"
02 #include "myrandn.h"
03 #include "myai.h"
04
05 const long RANDMAX = 2147483647;
06
07 const int NUM_SAMPLES = 1000;
08 const double MIN_X = -2;
09 const double MAX_X = 0.5;
10
11 double xs[NUM_SAMPLES];
12 double ys[NUM_SAMPLES];
13
14 double f(double x) {
15         return 2*x*x + 3*x + 5;
16 }
17
18 void setup() {
19
20         Serial.begin(115200);
```

```
21      lcdSetup();
22
23      randomSeed(time(NULL));
24
25      for(int n=0;n<NUM_SAMPLES;n++) {
26              xs[n] = (double)random(RANDMAX)/RANDMAX*(MAX_X - MIN_X)+MIN_X;
27              if(n<5) Serial.println(xs[n]);
28      }
29
30      for(int n=0;n<NUM_SAMPLES;n++) {
31              ys[n] = f(xs[n]);
32              if(n<5) Serial.println(xs[n]);
33      }
34
35      lcdSetPlotArea(xs, ys, NUM_SAMPLES);
36
37      lcdClear(TFT_BLACK);
38      lcdPlot(xs, ys, TFT_GREEN, NUM_SAMPLES);
39
40      for(int n=0;n<NUM_SAMPLES;n++) {
41              ys[n] += 0.1*randn();
42      }
43
44      lcdClear(TFT_BLACK);
45      lcdPlot(xs, ys, TFT_YELLOW, NUM_SAMPLES);
46
47      const int NUM_LEARNING = 0.8*NUM_SAMPLES;
48      const int NUM_THINKING = NUM_SAMPLES - NUM_LEARNING;
49
50      double *x_train = xs, *x_test = xs+NUM_LEARNING;
51      double *y_train = ys, *y_test = ys+NUM_LEARNING;
52
53      lcdClear(TFT_BLACK);
54      lcdPlot(x_train, y_train, TFT_GREEN, NUM_LEARNING);
55      lcdPlot(x_test, y_test, TFT_RED, NUM_THINKING);
56
57      int NUM_PATTERN = NUM_LEARNING;
58      int NUM_I = 1;
59      int NUM_H = 16;
60      int NUM_M = 16;
61      int NUM_O = 1;
62
63      Layer LayerH = {NUM_H, "relu"};
64      Layer LayerM = {NUM_M, "relu"};
65      Layer LayerO = {NUM_O, "linear"};
```

```
66
67        MyAI ai(NUM_I, LayerH, LayerM, LayerO);
68
69        double * I = x_train;
70        double * T = y_train;
71
72        ai.learning(I, T, NUM_PATTERN, 1000, 0.01);
73
74        I = x_test;
75        T = y_test;
76
77        double * y_pred = ai.think(I, NUM_THINKING);
78
79        lcdClear(TFT_BLACK);
80        lcdPlot(x_test, y_test, TFT_GREEN, NUM_THINKING);
81        lcdPlot(x_test, y_pred, TFT_RED, NUM_THINKING);
82
83 }
84
85 void loop() {
86
87 }
```

03 : myai.h 파일을 포함합니다. myai.h 파일은 63~65, 67, 72, 77줄에서 사용합니다. myai.h 파일에는 입력 층, 2개의 은닉 층, 출력 층으로 구성된 인공 신경망 클래스가 선언되어 있습니다. myai.h 파일에 선언된 클래스 멤버 함수들은 독자 여러분이 앞으로 직접 구현해 볼 인공 신경망의 함수들입니다.

57 : NUM_PATTERN 정수 변수를 선언한 후, NUM_LEARNING 값으로 초기화합니다. NUM_PATTERN 변수는 학습할 데이터의 개수를 저장합니다.

58 : NUM_I 정수 변수를 선언한 후, 1로 초기화합니다. NUM_I 변수는 입력층 노드의 개수를 저장합니다.

59 : NUM_H 정수 변수를 선언한 후, 16으로 초기화합니다. NUM_H 변수는 1차 은닉층 노드의 개수를 저장합니다.

60 : NUM_M 정수 변수를 선언한 후, 16으로 초기화합니다. NUM_M 변수는 2차 은닉층 노드의 개수를 저장합니다.

61 : NUM_O 정수 변수를 선언한 후, 1로 초기화합니다. NUM_O 변수는 출력층 노드의 개수를 저장합니다.

63 : 1차 은닉층의 정보를 담을 Layer 형 변수를 선언하고, NUM_H, “relu”로 초기화합니다. Layer 형 변수는 해당 층의 노드의 개수, 활성화 함수의 종류를 값으로 가집니다.

64 : 2차 은닉층의 정보를 담을 Layer 형 변수를 선언하고, NUM_M, “relu”로 초기화합니다.

65 : 출력층의 정보를 담을 Layer 형 변수를 선언하고, NUM_O, “linear”로 초기화합니다.

67 : MyAI 객체 ai를 생성합니다. 첫 번째 인자는 입력층의 노드 개수, 두 번째, 세 번째 인자는 은닉층에 대한 정보를 담은 Layer 변수, 네 번째 인자는 출력층에 대한 정보를 담은

Layer 변수입니다.
69 : I 실수 주소 변수를 선언한 후, x_train으로 초기화합니다.
70 : T 실수 주소 변수를 선언한 후, y_train으로 초기화합니다.
72 : ai.learning 함수를 호출하여 학습을 수행합니다. 첫 번째, 두 번째 인자는 입력 값과 목표 값, 세 번째 인자는 학습 데이터의 개수, 네 번째 인자는 학습 수행 횟수, 다섯 번째 인자는 학습률을 나타냅니다.
74 : I 변수가 x_test을 가리키게 합니다.
75 : T 변수가 y_test을 가리키게 합니다.
77 : ai.think 함수를 호출하여 예측을 수행합니다. 입력 값 I에 대해 예측 값 y_pred 실수 주소를 내어줍니다.
79 : lcdClear 함수를 호출하여 화면을 검정색으로 채웁니다.
80 : lcdPlot 함수를 호출하여 I, T 좌표 값에 맞추어 실제 데이터에 대한 그래프를 그립니다. 그래프의 색깔은 초록색으로 그립니다.
81 : lcdPlot 함수를 호출하여 I, y_pred 좌표 값에 맞추어 예측한 그래프를 그립니다. 그래프의 색깔은 빨간색으로 그립니다.

2. 업로드를 수행합니다.

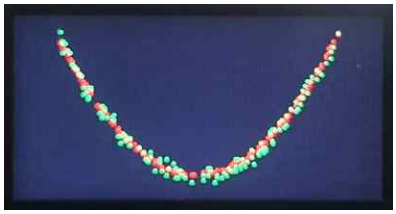
3. 다음은 실행 결과 화면입니다.

```
epoch :   950, sum error : 4.733655 , 5.414 sec
epoch :   960, sum error : 4.829234 , 5.412 sec
epoch :   970, sum error : 4.802743 , 5.412 sec
epoch :   980, sum error : 4.722942 , 5.412 sec
epoch :   990, sum error : 4.714084 , 5.409 sec
epoch :  1000, sum error : 4.916167 , 5.411 sec


Total time taken (in seconds) 544.189
```

10 회기마다 학습 결과와 학습 시간을 출력합니다. 전체 학습시간은 544초(9분 4초) 정도 걸립니다.

다음과 같이 LCD 화면에 그려진 그래프도 확인합니다.



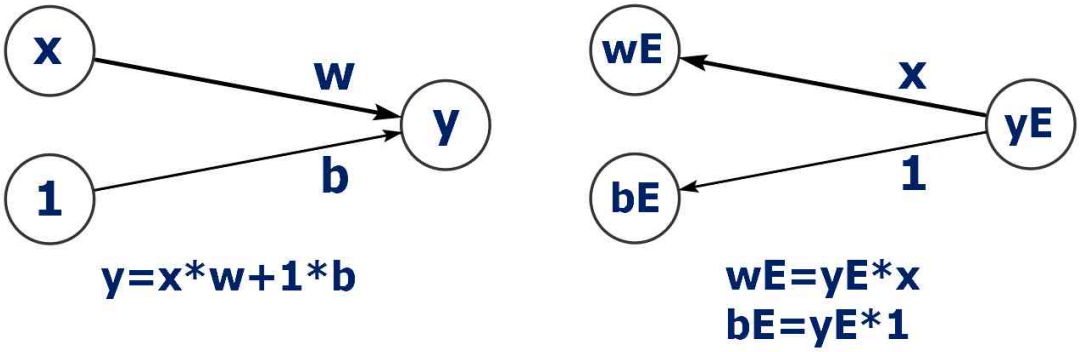
초록색 점은 I, T의 분포를 나타내며, 빨간색 점은 I, y_pred의 분포를 나타냅니다. 인공 신경망이 학습을 수행한 후에 I 값에 대한 예측 값을 실제 함수에 근사해서 생성해 내는 것을 볼 수 있습니다.

지금까지 DNN 인공 신경망 라이브러리를 이용하여 2차 함수에 대한 근사 학습을 수행해 보았습니다. 이후에는 독자 여러분이 인공 신경망에 대한 공부를 해가며 최종적으로 DNN 인공 신경망 라이브러리를 직접 구현해 보게 됩니다. 그 과정에서 인공지능의 딥러닝에 대한 원리를 이해하고 실제로 활용할 수 있는 능력을 키우도록 합니다.

02 딥러닝 동작 원리 이해하기

여기서는 단위 인공 신경(1입력 1출력 인공 신경)의 동작을 상식적인 수준에서 살펴보면서 딥러닝의 동작 원리를 이해해 봅니다. 또 딥러닝과 관련된 중요한 용어들, 예를 들어, 순전파, 목표값, 역전파 오차, 오차 역전파, 학습률과 같은 용어들을 이해해 보도록 합니다.

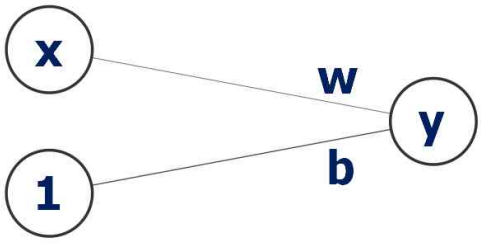
다음 그림은 이 단원에서 살펴볼 인공 신경의 순전파와 역전파를 나타내는 핵심 원리를 표현하고 있습니다.



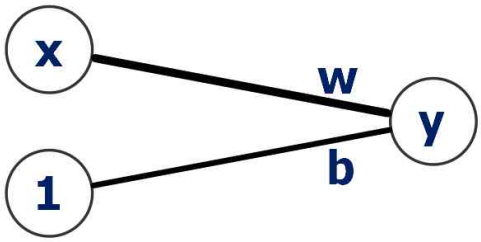
독자 여러분이 이 그림의 내용을 이해한다면 딥러닝의 가장 중요한 원리를 이해하게 되는 것입니다. 딥러닝의 핵심 원리 지금부터 이해해 봅시다!

01 딥러닝 단위 동작 살펴보기

다음은 앞에서 소개한 단일 인공 신경의 그림입니다. 이 인공 신경은 입력 노드 1개, 출력 노드 1개, 편향으로 구성된 단일 인공 신경입니다.



이 인공 신경을 학습 시키면 다음과 같이 가중치와 편향의 값이 바뀌게 됩니다. 즉, 신호 전달 강도가 더 세지거나 약해집니다.



수식으로는 다음과 같이 표현합니다.

$$y = x^*w + 1^*b$$

이 수식에 대해서 구체적으로 생각해 봅니다. 다음과 같이 각 변수에 값을 줍니다.

$$x = 2$$
$$w = 3$$
$$b = 1$$

그러면 식은 다음과 같이 됩니다.

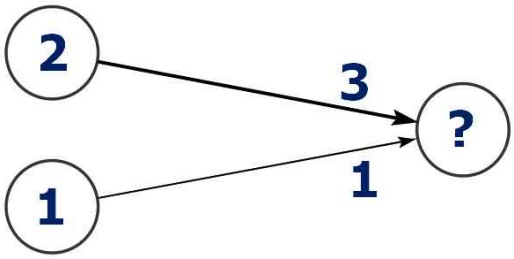
$$y = 2 \times 3 + 1 \times 1$$
$$y = ?$$

y는 얼마가 될까요? 다음과 같이 계산해서 y는 7이 됩니다.

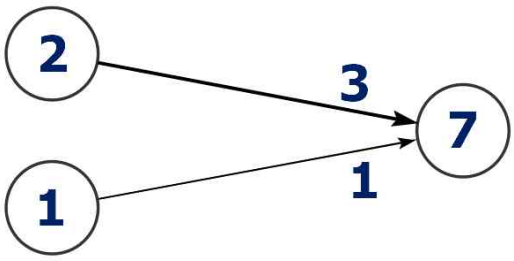
$$2 \times 3 + 1 \times 1 = 7$$

순전파

이 상황을 그림으로 생각해 봅니다. 다음과 같이 x, w, b 값이 y로 흘러가는 인공 신경이 있습니다. 이 과정을 인공 신경의 순전파라고 합니다.

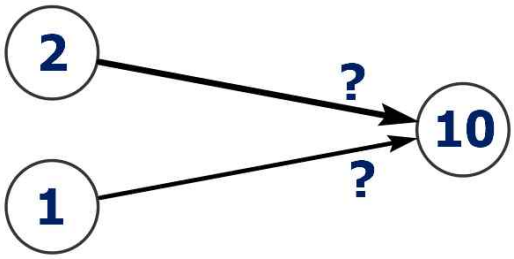


이 경우 y로 얼마가 나올까요? 앞에서 살펴본 대로 다음과 같이 7이 나오게 됩니다.

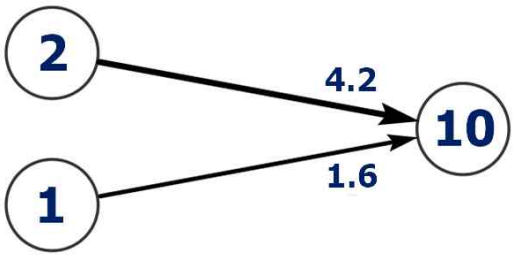


목표값과 역전파 오차

그런데 y로 10이 나오게 하고 싶습니다. 이런 경우에 w와 b의 값을 어떻게 바꿔야 할까요?



y값이 10이 되려면 3이 모자랍니다. y의 오차는 w와 b의 오차로 인해 발생합니다. 따라서 w와 b값을 적당히 증가시키면 y로 10에 가까운 값이 나오게 할 수 있겠죠? 예를 들어, w를 4.2로, b를 1.6으로 증가시키면 y의 값은 10이 됩니다.



그러면 w, b값을 어떤 기준으로 얼마나 증가시켜야 할까요? 이 과정을 이해하는 것이 바로 역전파의 핵심을 이해하는 것이고 나아가서 딥러닝의 핵심을 이해하는 것입니다. 이 과정을 자세히 살펴봅니다.

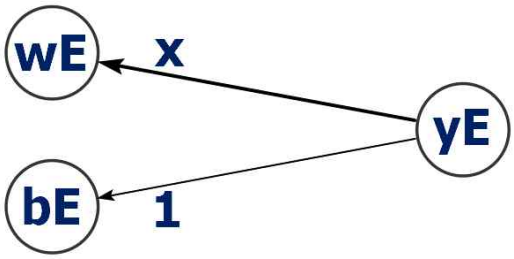
이전 그림에서 y로 7이 흘러나갔는데 우리는 이 값이 10이 되기를 원합니다. 여기서 10은 목표값이 됩니다. 참고로 인공지능 학습 시 목표값은 라벨이라고 합니다. 다음 수식에서 y_T는 목표값 10을 갖습니다.

$$y_T = 10$$
$$y = 7$$
$$y_E = y - y_T = -3$$

y값은 현재값 7인 상태이며, y_E는 현재값에서 목표값을 뺀 값 -3이 됩니다. 이 때, y_E값을 역전파 오차라고 하며, 역전파에 사용할 오차값입니다.

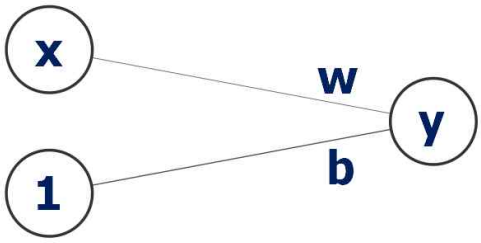
역전파

다음 그림은 역전파를 나타내는 그림입니다. 정확히는 오차 역전파를 나타내는 그림입니다.



이 그림은 오차 yE의 값이 오차 wE, bE로 거꾸로 흘러가는 상황을 나타냅니다. w와 b의 오차 wE와 bE는 y의 오차 yE를 이용하여 구합니다. 참고로 E는 오차(Error)의 약자입니다.

그러면 앞의 역전파 그림은 어떻게 나온 걸까요? 다음은 앞에서 살펴본 순전파 과정을 나타내는 그림입니다.



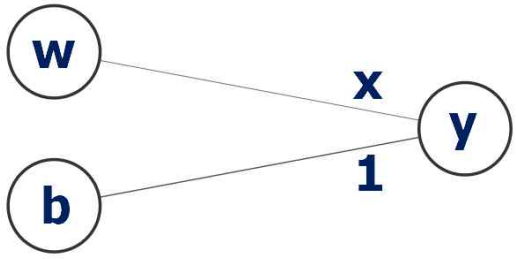
수식으로는 다음과 같이 표현합니다.

$$y = x^*w + 1^*b$$

이 수식은 다음과 같이 바꿀 수 있습니다.

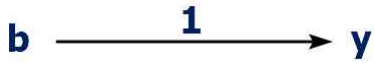
$$y = w^*x + b^*1$$

x와 w, 1과 b의 위치를 바꾼 수식으로 결과적으로 이전과 같은 수식입니다. 바뀐 수식에 의해 순전파 그림은 다음과 같이 표현할 수도 있습니다.



위 그림은 다음과 같이 해석할 수 있습니다.

- w는 x만큼 y로 갔어. 

- b는 1만큼 y로 갔어. 

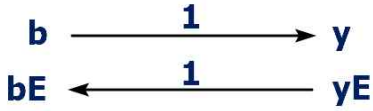
우리는 다음 사항이 궁금합니다.

- yE는 얼마만큼 wE로 가야해?
- yE는 얼마만큼 bE로 가야해?

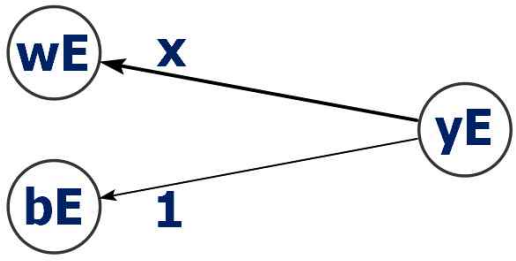
이에 대한 답은 다음과 같습니다.

- w가 x만큼 y로 왔으니 yE도 x만큼 wE로 가야하는 거 아냐?

- b가 1만큼 y로 왔으니 yE도 1만큼 bE로 가야하는 거 아냐?



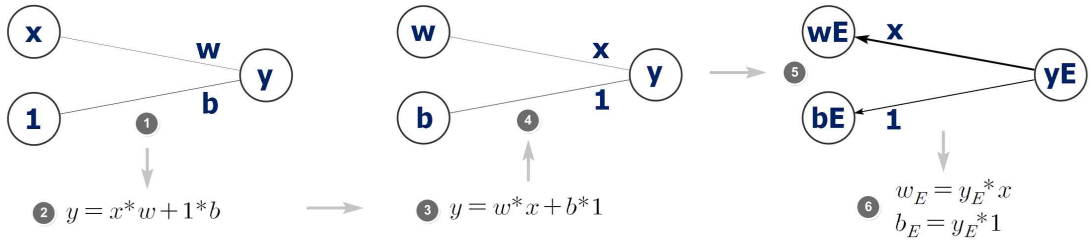
이 상황을 그림으로 나타내면 다음과 같습니다. 이 그림은 역전파를 나타내는 그림입니다.



수식으로는 다음과 같이 표현합니다.

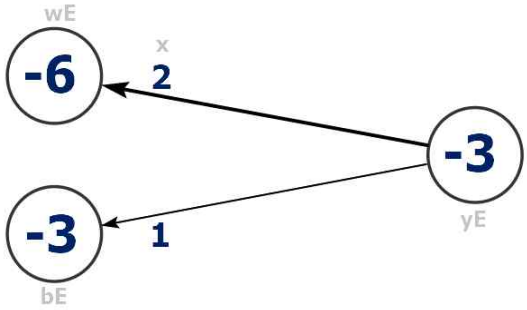
$$w_E = y_E{}^* x$$
$$b_E = y_E{}^* 1$$

이 방법이 바로 딥러닝에서 사용하는 역전파입니다. 이 방법은 실제로 편미분을 적용하여 얻은 방법으로 이 책에서는 직관적인 방법으로 설명을 대체하였습니다. 편미분을 이용하여 역전파를 유도하는 방법은 부록을 참조합니다. 다음 그림은 지금까지의 과정을 정리한 그림입니다.



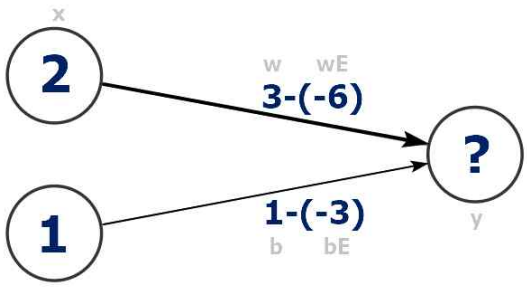
w와 b가 x와 1을 통해 y로 순전파되는 것처럼 yE는 x와 1을 통해 wE와 bE로 역전파됩니다.

위 수식에 의해 wE, bE는 다음 그림과 같이 계산됩니다.



최적화하기

이렇게 구한 값을 다시 다음과 같이 밀어 넣으면 될까요? 앞에서 구한 wE, bE의 값이 음수가 되기 때문에 일단 빼주어야 합니다. 그래야 원래 값이 증가하기 때문입니다.

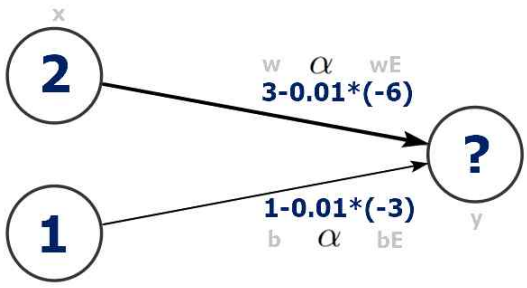


그런데 wE, bE값이 너무 큽니다. 이 상태로 계산을 하면 새로운 y값은 (2*9+1*4)와 같이 계산되어 22가 되게 되며, 우리가 원하는 10보다 더 큰 값이 나오게 됩니다. 구체적인 계산 과정은 다음과 같습니다.

```
x = 2
w = w - wE = 3-(-6) = 9
b = b - bE = 1-(-3) = 4
y = x*w + 1*b = 2*9+1*4 = 22
```

학습률

그러면 이런 방법은 어떨까요? wE, bE에 적당한 값을 곱해주어 값을 줄이는 겁니다. 여기서는 0.01을 곱해줍니다. 그러면 다음과 같이 계산할 수 있습니다.



*** 여기서 α는 학습률이라고 하며 뒤에서는 lr이라는 이름으로 구현합니다. lr은 learning rate의 약자로 학습률을 의미합니다.

이렇게 하면 2*(3.06)+1.03=7.15가 나옵니다. 오! 이렇게 조금씩 늘려나가면 10을 만들 수 있겠네요!

여기서 곱해준 0.01은 학습률이라고 하는 값입니다. 일반적으로 학습률 값은 0.01로 시작하여 학습이 진행되는 상황에 따라 조금씩 늘이거나 줄여서 사용합니다.

경사 하강법과 인공 신경망 학습

위 그림에 따라 새로운 w, b값을 구하는 수식은 다음과 같습니다.

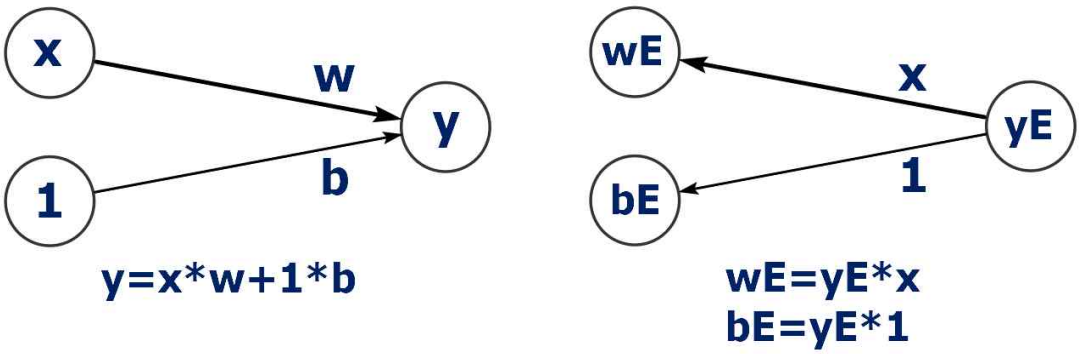
$$w = w - \alpha w_E$$
$$b = b - \alpha b_E$$

이 수식을 경사하강법이라고 합니다. 그리고 이 수식을 적용하여 w, b 값을 갱신하는 과정을 인공 신경망의 학습이라고 합니다. 여러분은 방금 전에 1회의 학습을 수행하는 과정을 보신 겁니다. 이 과정을 컴퓨터를 이용하여 반복하여 수행하면 우리가 원하는 값을 얻게 해 주는 하나짜리 인공 신경망을 만들 수 있습니다.

02 딥러닝 단위 동작 구현해 보기

지금까지의 과정을 그림과 수식을 통해 정리한 후, 구현을 통해 확인해 봅니다.

다음 그림은 지금까지 살펴본 입력1 출력1로 구성된 인공 신경의 순전파와 역전파를 나타냅니다.



입력1 출력1로 구성된 인공 신경의 수식을 정리하면 다음과 같습니다.

순전파

$$y = x^{*}w + 1^{*}b$$

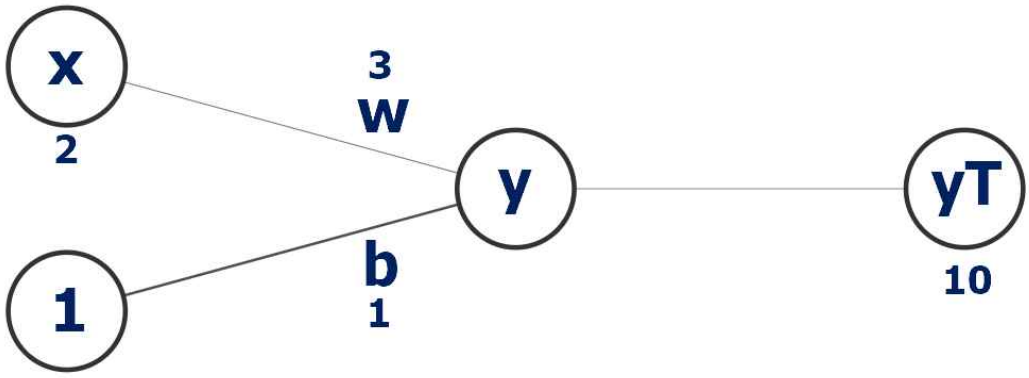
역전파

$$w_E = y_E{}^{*}x$$
$$b_E = y_E{}^{*}1$$

인공 신경망 학습

$$w = w - \alpha w_E$$
$$b = b - \alpha b_E$$

지금까지의 과정을 구현을 통해 살펴봅니다. 다음 그림을 살펴봅니다.



이 그림에서 입력값 x, 가중치 w, 편향 b는 각각 2, 3, 1이고 목표값 yT는 10입니다. x, yT를 이용하여 w, b에 대해 학습을 수행해 봅니다.

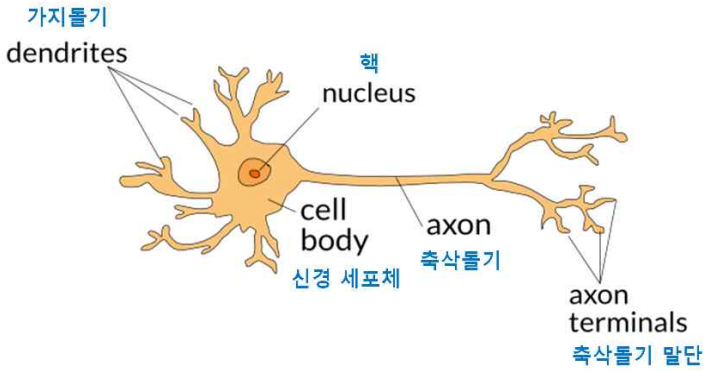
*** 이 값들은 임의의 값들입니다. 다른 값들을 사용하여 학습을 수행할 수도 있습니다.

1. 다음과 같이 예제를 작성합니다.
422.ino

```
01 double x = 2;
02 double yT = 10;
03 double w = 3;
04 double b = 1;
05
06 void dnn_test() {
07
08       double y = x*w + 1*b;
09       printf("y  = %6.3f\n", y);
10
11       double yE = y - yT;
12
13       double wE = yE*x;
14       double bE = yE*1;
15       printf("wE = %6.3f, bE = %6.3f\n", wE, bE);
16
17       double lr = 0.01;
18       w = w - lr*wE;
19       b = b - lr*bE;
20       printf("w  = %6.3f, b  = %6.3f\n", w, b);
21
22 }
```

```
23
24 void setup() {
25
26       Serial.begin(115200);
27       delay(1000);
28
29       dnn_test();
30
31 }
32
33 void loop() {
34
35 }
```

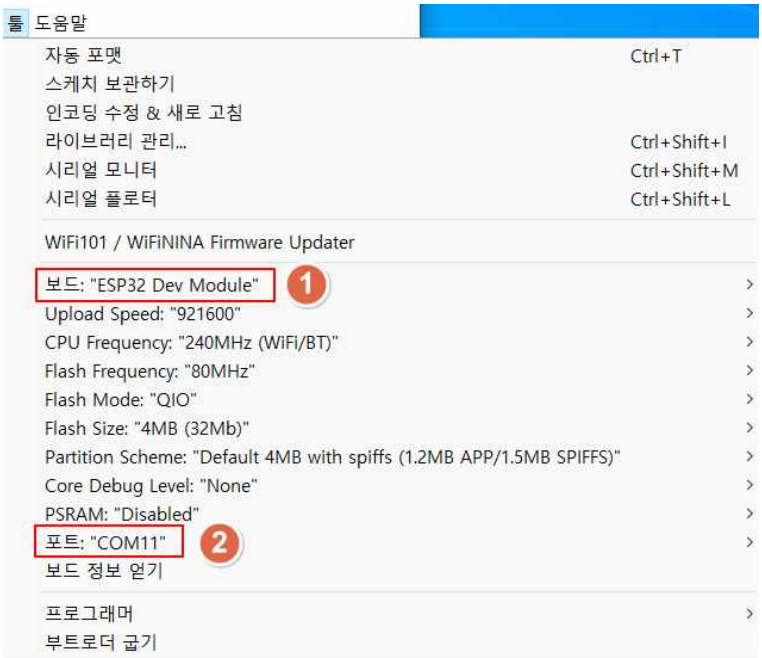
01 : 변수 x를 선언한 후, 2로 초기화합니다.
02 : 변수 yT를 선언한 후, 10으로 초기화합니다.
03 : 가중치 변수 w를 선언한 후, 3으로 초기화합니다. 가중치 w는 입력값의 강도, 세기라고도 하며 입력값을 증폭 시키거나 감소시키는 역할을 합니다. 인공 신경도 가지돌기의 두께에 따라 입력 신호가 증폭되거나 감소될 수 있는데, 이런 관점에서 가중치는 가지돌기의 두께에 해당되는 변수로 생각할 수 있습니다.



04 : 편향 변수 b를 선언한 후, 1로 초기화합니다. 편향은 가중치를 거친 입력값의 합(=전체 입력 신호)에 더해지는 값으로 입력신호를 좀 더 세게 해주거나 약하게 하는 역할을 합니다.
06~22 : dnn_test 함수를 정의합니다.
08 : 순전파 수식을 구현합니다.
09 : printf 함수를 호출하여 순전파 결과값 y를 출력합니다. 소수점 이하 3자리까지 출력합니다.
11 : yE 변수를 선언한 후, 순전파 결과값에서 목표값을 빼 오차값을 넣어줍니다.
13 : wE 변수를 선언한 후, 가중치 값에 대한 역전파 값을 받습니다.
14 : bE 변수를 선언한 후, 편향 값에 대한 역전파 값을 받습니다.
15 : printf 함수를 호출하여 역전파 결과값 wE, bE를 출력합니다. 소수점 이하 3자리까지 출력합니다.

17 : 학습률 변수 lr을 선언한 후, 0.01로 초기화합니다.
18 : wE 역전파값에 학습률을 곱한 후, w값에서 빼줍니다. 이 과정에서 w 변수에 대한 학습이 이루어집니다.
19 : bE 역전파값에 학습률을 곱한 후, b값에서 빼줍니다. 이 과정에서 b 변수에 대한 학습이 이루어집니다.
20 : printf 함수를 호출하여 학습된 결과값 w, b를 출력합니다. 소수점 이하 3자리까지 출력합니다.
26 : 시리얼 통신 속도를 115200bps로 초기화합니다.
27 : 1000밀리초(=1초)간 기다립니다.
29 : dnn_test 함수를 호출합니다.

2. [툴] 메뉴를 이용하여 보드, 포트를 다음과 같이 선택합니다.



3. 업로드를 수행합니다.

4. 다음은 실행 결과 화면입니다.

```
y  =  7.000
wE = -6.000, bE = -3.000
w  =  3.060, b  =  1.030
```

현재 y값은 7입니다. wE, bE 값을 확인합니다. 또, w, b 값을 확인합니다.

반복 학습 2회 수행하기

여기서는 반복 학습 2회를 수행해 봅니다.

1. 다음과 같이 예제를 수정합니다.
422_2.ino

```
01 double x = 2;
02 double yT = 10;
03 double w = 3;
04 double b = 1;
05
06 void dnn_test() {
07
08     for(int epoch=0;epoch<2;epoch++) {
09
10         printf("epoch = %d\n", epoch);
11
12         double y = x*w + 1*b;
13         printf(" y  = %6.3f\n", y);
14
15         double yE = y - yT;
16
17         double wE = yE*x;
18         double bE = yE*1;
19         printf(" wE = %6.3f, bE = %6.3f\n", wE, bE);
20
21         double lr = 0.01;
22         w = w - lr*wE;
23         b = b - lr*bE;
24         printf(" w  = %6.3f, b  = %6.3f\n", w, b);
25
26     }
27
28 }
29
30 void setup() {
31
32   Serial.begin(115200);
33   delay(1000);
34
```

```
35    dnn_test();
36
37 }
38
39 void loop() {
40
41 }
```

08 : epoch값을 0에서 2 미만까지 바꾸어가며 10~24줄을 2회 수행합니다.

10 : printf 함수를 호출하여 epoch 값을 출력해 줍니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
epoch = 0
 y   =  7.000
 wE = -6.000, bE = -3.000
 w   =  3.060, b   =  1.030
epoch = 1
 y   =  7.150
 wE = -5.700, bE = -2.850
 w   =  3.117, b   =  1.058
```

y 값이 7에서 7.150으로 바뀌는 것을 확인합니다. wE, bE 값을 확인합니다. 또, w, b 값을 확인합니다.

반복 학습 20회 수행하기

여기서는 반복 학습 20회를 수행해 봅니다.

1. 다음과 같이 예제를 수정합니다.

```
08        for(int epoch=0;epoch<20;epoch++) {
```

08 : epoch값을 0에서 20 미만까지 수행합니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
epoch = 18
 y  =  8.808
 wE = -2.383, bE = -1.192
 w  =  3.747, b  =  1.374
epoch = 19
 y  =  8.868
 wE = -2.264, bE = -1.132
 w  =  3.770, b  =  1.385
```

y 값이 8.868까지 접근하는 것을 확인합니다.

반복 학습 200회 수행하기

여기서는 반복 학습 200회를 수행해 봅니다.

1. 다음과 같이 예제를 수정합니다.

```
08        for(int epoch=0;epoch<200;epoch++) {
```

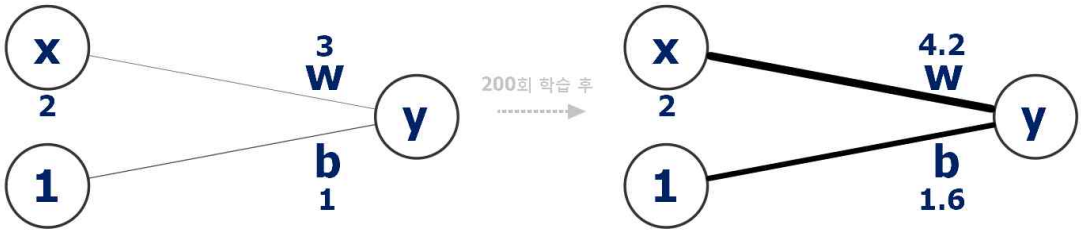
08 : epoch값을 0에서 200 미만까지 수행합니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
epoch = 198
 y  = 10.000
 wE = -0.000, bE = -0.000
 w  =  4.200, b  =  1.600
epoch = 199
 y  = 10.000
 wE = -0.000, bE = -0.000
 w  =  4.200, b  =  1.600
```

y 값이 10.000에 수렴하는 것을 확인합니다. 이 때, 가중치 w는 4.2, 편향 b는 1.6에 수렴합니다.



200회 학습 후에는 그림처럼 w, b의 강도가 세지는 것을 볼 수 있습니다. 인공 신경망의 학습은 이처럼 가중치 w, 편향 b의 값을 조정하는 과정입니다.

오차값 계산하기

여기서는 인공 신경망을 통해 얻어진 예측값과 목표값의 오차를 계산하는 부분을 추가해 봅니다. 오차(error)는 손실(loss) 또는 비용(cost)이라고도 합니다. 오차값이 작을수록 예측을 잘하는 인공 신경망입니다.

1. 다음과 같이 예제를 수정합니다.
422_3.ino

```
01 double x = 2;
02 double yT = 10;
03 double w = 3;
04 double b = 1;
05
06 void dnn_test() {
07
08     for(int epoch=0;epoch<200;epoch++) {
09
10         printf("epoch = %d\n", epoch);
11
12         double y = x*w + 1*b;
13         printf(" y  = %6.3f\n", y);
14
15         double E = (y-yT)*(y-yT)/2;
16         printf(" E  = %.7f\n", E);
17         if (E < 0.0000001)
18             break;
19
20         double yE = y - yT;
21
22         double wE = yE*x;
```

```
23                double bE = yE*1;
24                printf(" wE = %6.3f, bE = %6.3f\n", wE, bE);
25
26                double lr = 0.01;
27                w = w - lr*wE;
28                b = b - lr*bE;
29                printf(" w  = %6.3f, b  = %6.3f\n", w, b);
30
31        }
32
33 }
34
35 void setup() {
36
37   Serial.begin(115200);
38   delay(1000);
39
40   dnn_test();
41
42 }
43
44 void loop() {
45
46 }
```

15 : 변수 E를 선언한 후, 다음과 같은 형태의 수식을 구현합니다.

$$E = \frac{1}{2}(y - y_T)^2$$

y의 값이 yT에 가까울수록 E의 값은 0에 가까워집니다. 즉, 오차값이 0에 가까워집니다. 이 수식을 오차함수 또는 손실함수 또는 비용함수라고 합니다.

16 : printf 함수를 호출하여 오차값 E를 출력합니다. 소수점 이하 7자리까지 출력합니다.

17, 18 : 오차값 E가 0.0000001(1천만분의1)보다 작으면 break문을 수행하여 08줄의 for문을 빠져 나갑니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
epoch = 171
 y   = 10.000
 E   = 0.0000001
 wE = -0.001, bE = -0.000
 w   =  4.200, b   =  1.600
epoch = 172
 y   = 10.000
 E   = 0.0000001
```

epoch 값이 172일 때 for 문을 빠져 나갑니다. y값은 10에 수렴합니다.

학습률 변경하기

여기서는 학습률 값을 변경시켜 보면서 학습의 상태를 살펴봅니다.

1. 다음과 같이 예제를 수정합니다.

```
26                double lr = 0.05;
```

26 : 학습률 값을 0.05로 변경합니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
epoch = 30
 y   =  9.999
 E   = 0.0000001
 wE = -0.001, bE = -0.001
 w   =  4.200, b   =  1.600
epoch = 31
 y   = 10.000
 E   = 0.0000001
```

(31+1)회 째 학습이 완료되는 것을 볼 수 있습니다.

4. 다음과 같이 예제를 수정합니다.

```
26                double lr = 0.005;
```

26 : 학습률 값을 0.005로 변경합니다.

5. 업로드를 수행합니다.

6. 다음은 실행 결과 화면입니다.

```
epoch = 198
 y   =  9.980
 E   = 0.0001991
 wE  = -0.040, bE = -0.020
 w   =  4.192, b  =  1.596
epoch = 199
 y   =  9.981
 E   = 0.0001893
 wE  = -0.039, bE = -0.019
 w   =  4.192, b  =  1.596
```

(199+1)회 째 학습이 완료되지 않은 상태로 종료되는 것을 볼 수 있습니다.

7. 다음과 같이 예제를 수정합니다.

```
08        for(int epoch=0;epoch<2000;epoch++) {
```

08 : epoch값을 0에서 2000 미만까지 수행합니다.

8. 업로드를 수행합니다.

9. 다음은 실행 결과 화면입니다.

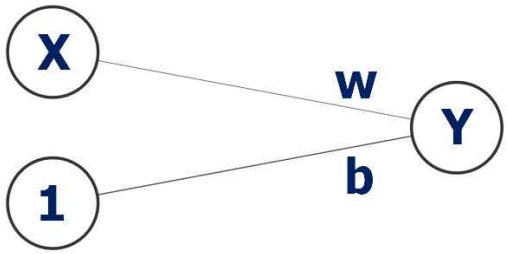
```
epoch = 348
 y   = 10.000
 E   = 0.0000001
 wE  = -0.001, bE = -0.000
 w   =  4.200, b  =  1.600
epoch = 349
 y   = 10.000
 E   = 0.0000001
```

(349+1)회 째 학습이 완료되는 것을 볼 수 있습니다.

03 y=3*x+1 학습시켜 보기

여기서는 다음과 같은 숫자들의 집합 X, Y를 이용하여, 단일 인공 신경을 학습시켜 봅니다.

X:	-1	0	1	2	3	4
Y:	-2	1	4	7	10	13



그래서 다음 함수를 근사하는 인공 신경 함수를 만들어 보도록 합니다.

y = f(x) = 3*x + 1 (x는 실수)

인공 신경을 학습시키는 과정은 w, b 값을 X, Y 값에 맞추어 가는 과정입니다. 그래서 학습이 진행됨에 따라 w 값은 3에 가까운 값으로, b 값은 1에 가까운 값으로 이동하게 됩니다.

*** 이 예제는 1장에서 tensorflow를 이용하여 수행했던 예제를 재구현하고 있습니다.

1. 다음과 같이 예제를 작성합니다.

423.ino

```
01 double xs[] = {-1, 0, 1, 2,  3,  4};
02 double ys[] = {-2, 1, 4, 7, 10, 13};
03 double w = 10;
04 double b = 10;
05
06 void dnn_test() {
07
08       double y = xs[0]*w + 1*b;
09       printf("x  = %6.3f, y  = %6.3f\n", xs[0], y);
10
11       double yT = ys[0];
12       double E = (y-yT)*(y-yT)/2;
13       printf("E  = %.7f\n", E);
```

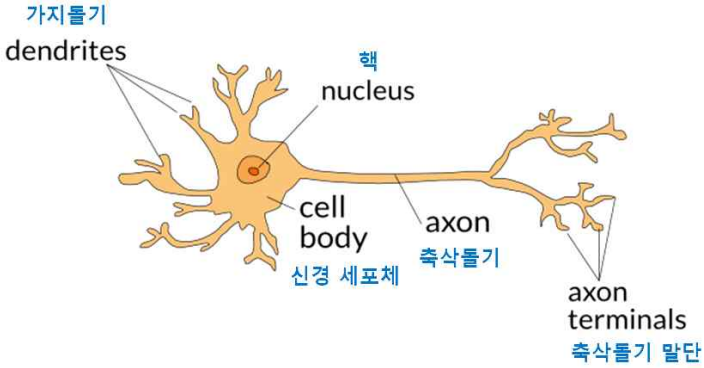
```
14
15        double yE = y - yT;
16        double wE = yE*xs[0];
17        double bE = yE*1;
18        printf("wE = %6.3f, bE = %6.3f\n", wE, bE);
19
20        double lr = 0.01;
21        w = w - lr*wE;
22        b = b - lr*bE;
23        printf("w  = %6.3f, b  = %6.3f\n", w, b);
24
25 }
26
27 void setup() {
28
29        Serial.begin(115200);
30        delay(1000);
31
32        dnn_test();
33
34 }
35
36 void loop() {
37
38 }
```

01, 02 : 실수형 배열 변수 xs, ys를 선언한 후, 다음 X, Y 값으로 초기화합니다.

X:	-1	0	1	2	3	4
Y:	-2	1	4	7	10	13

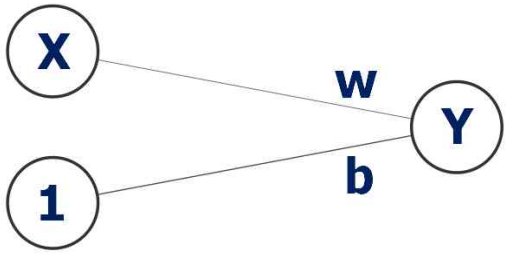
숫자 뒤에 점(.)은 실수를 나타냅니다.

03 : 입력값의 가중치값을 저장할 변수 w를 선언한 후, 10으로 초기화합니다. 10은 임의로 선택한 값입니다. 입력값의 가중치는 입력값의 강도, 세기라고도 하며 입력값을 증폭 시키거나 감소시키는 역할을 합니다. 인공 신경도 가지돌기의 두께에 따라 입력 신호가 증폭되거나 감소될 수 있는데, 이런 관점에서 가중치는 가지돌기의 두께에 해당되는 변수로 생각할 수 있습니다.



04 : 인공 신경의 편향값을 저장할 변수 b를 선언한 후, 10으로 초기화합니다. 10은 임의로 선택한 값입니다. 편향값은 가중치를 거친 입력값의 합(=전체 입력신호)에 더해지는 값으로 입력신호를 좀 더 세게 해주거나 약하게 하는 역할을 합니다.

08 : 다음과 같이 단일 인공 신경을 수식으로 표현합니다.



$$y = xw + 1b$$
$$= xw + b$$

일단 xs[0] 항목을 w에 곱한 후, b를 더해준 후, 변수 y에 대입해 줍니다. 이 과정에서 순전파가 이루어집니다. 즉, xs[0] 항목이 w에 곱해지고 b와 더해져 y에 도달하는 과정을 순전파라고 합니다. 순전파 결과 얻어진 y값을 인공 신경망에 의한 예측값이라고 합니다.

09 : printf 함수를 호출하여 xs[0], y 값을 출력합니다.

11 : 변수 yT를 선언한 후, ys[0]값을 받습니다. ys[0]은 인공 신경망에 대한 xs[0]값의 목표값입니다.

12 : 변수 E를 선언한 후, 다음과 같은 형태의 수식을 구현합니다.

$$E = \frac{1}{2}(y - y_T)^2$$

y의 값이 yT에 가까울수록 E의 값은 0에 가까워집니다. 즉, 오차값이 0에 가까워집니다. 이 수식을 오차함수 또는 손실함수 또는 비용함수라고 합니다.

13 : printf 함수를 호출하여 E 값을 출력합니다. 소수점 이하 7자리까지 출력합니다.

15 : yE 변수를 선언한 후, 순전파 결과값에서 목표값을 빼 오차값을 넣어줍니다.

16 : wE 변수를 선언한 후, 가중치 값에 대한 역전파 값을 받습니다.

17 : bE 변수를 선언한 후, 편향 값에 대한 역전파 값을 받습니다.

18 : printf 함수를 호출하여 역전파 결과값 wE, bE를 출력합니다. 소수점 이하 3자리까지 출력합니다.

20 : 학습률 변수 lr을 선언한 후, 0.01로 초기화합니다.

21 : wE 역전파값에 학습률을 곱한 후, w값에서 빼줍니다. 이 과정에서 w 변수에 대한 학습이 이루어집니다.

22 : bE 역전파값에 학습률을 곱한 후, b값에서 빼줍니다. 이 과정에서 b 변수에 대한 학습이 이루어집니다.

23 : printf 함수를 호출하여 학습이 1회 수행된 w, b 값을 출력합니다. 소수점 이하 3자리까지 출력합니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
x  = -1.000, y  =  0.000
E  = 2.0000000
wE = -2.000, bE =  2.000
w  = 10.020, b  =  9.980
```

w, b 값이 각각 10.020, 9.980으로 표시되는 것을 확인합니다.

전체 입력 데이터 학습 수행하기

이제 다음 좌표값 전체에 대해 1회 학습을 수행해 봅니다.

X:	-1	0	1	2	3	4
Y:	-2	1	4	7	10	13

1. 다음과 같이 예제를 수정합니다.

423_2.ino

```
01 double xs[] = {-1, 0, 1, 2,  3,  4};
02 double ys[] = {-2, 1, 4, 7, 10, 13};
03 double w = 10;
04 double b = 10;
05
06 void dnn_test() {
07
08     for(int n=0;n<6;n++) {
09
10         double y = xs[n]*w + 1*b;
11         printf("x  = %6.3f, y  = %6.3f\n", xs[n], y);
12
13         double yT = ys[n];
14         double E = (y-yT)*(y-yT)/2;
15         printf("E  = %.7f\n", E);
16
17         double yE = y - yT;
18         double wE = yE*xs[n];
19         double bE = yE*1;
20         printf("wE = %6.3f, bE = %6.3f\n", wE, bE);
21
22         double lr = 0.01;
23         w = w - lr*wE;
24         b = b - lr*bE;
25         printf("w  = %6.3f, b  = %6.3f\n", w, b);
26
27         for(int n=0;n<25;n++) printf("=");
28         printf("\n");
29
30     }
31
32 }
33
34 void setup() {
35
36     Serial.begin(115200);
37     delay(1000);
38
39     dnn_test();
40
```

```
41 }
42
43 void loop() {
44
45 }
```

08 : n값을 0에서 6 미만까지 바꾸어가며 10~28줄을 6회 수행합니다.

10, 11, 18 : xs[0]을 xs[n]으로 변경합니다.

13 : ys[0]을 ys[n]으로 변경합니다.

27 : 실행 경계를 표시하기 위해 "="을 25개 출력합니다.

28 : 개행문자 '\n'를 출력합니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
x  = -1.000, y  =  0.000
E  = 2.0000000
wE = -2.000, bE =  2.000
w  = 10.020, b  =  9.980
=========================
x  =  0.000, y  =  9.980
E  = 40.3202000
wE =  0.000, bE =  8.980
w  = 10.020, b  =  9.890
=========================
x  =  1.000, y  = 19.910
E  = 126.5672320
wE = 15.910, bE = 15.910
w  =  9.861, b  =  9.731
=========================
```

```
x  =  2.000, y  = 29.453
E  = 252.0662245
wE = 44.906, bE = 22.453
w  =  9.412, b  =  9.507
=========================
x  =  3.000, y  = 37.742
E  = 384.8117627
wE = 83.226, bE = 27.742
w  =  8.580, b  =  9.229
=========================
x  =  4.000, y  = 43.547
E  = 466.5735925
wE = 122.190, bE = 30.547
w  =  7.358, b  =  8.924
=========================
```

가중치, 편향값 학습과정 살펴보기

가중치와 편향값만 확인해 봅니다.

1. 다음과 같이 예제를 수정합니다.

423_3.ino

```
01 double xs[] = {-1, 0, 1, 2,  3,  4};
02 double ys[] = {-2, 1, 4, 7, 10, 13};
03 double w = 10;
04 double b = 10;
05
06 void dnn_test() {
```

```
07
08      for(int n=0;n<6;n++) {
09
10              double y = xs[n]*w + 1*b;
11
12              double yT = ys[n];
13              double E = (y-yT)*(y-yT)/2;
14
15              double yE = y - yT;
16              double wE = yE*xs[n];
17              double bE = yE*1;
18
19              double lr = 0.01;
20              w = w - lr*wE;
21              b = b - lr*bE;
22              printf("w  = %6.3f, b  = %6.3f\n", w, b);
23
24      }
25
26 }
27
28 void setup() {
29
30      Serial.begin(115200);
31      delay(1000);
32
33      dnn_test();
34
35 }
36
37 void loop() {
38
39 }
```

22 : w, b에 대한 출력만 합니다. 나머지 출력 루틴은 주석처리하거나 지워줍니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
w   = 10.020, b   =  9.980
w   = 10.020, b   =  9.890
w   =  9.861, b   =  9.731
w   =  9.412, b   =  9.507
w   =  8.580, b   =  9.229
w   =  7.358, b   =  8.924
```

학습 회수에 따라 w, b값이 바뀌는 것을 확인합니다.

반복 학습 2회 수행하기

여기서는 반복 학습 2회를 수행해 봅니다.

1. 다음과 같이 예제를 수정합니다.

423_4.ino

```
01 double xs[] = {-1, 0, 1, 2,  3,  4};
02 double ys[] = {-2, 1, 4, 7, 10, 13};
03 double w = 10;
04 double b = 10;
05
06 void dnn_test() {
07
08     for(int epoch=0;epoch<2;epoch++) {
09
10         for(int n=0;n<6;n++) {
11
12             double y = xs[n]*w + 1*b;
13
14             double yT = ys[n];
15             double E = (y-yT)*(y-yT)/2;
16
17             double yE = y - yT;
18             double wE = yE*xs[n];
19             double bE = yE*1;
20
21             double lr = 0.01;
22             w = w - lr*wE;
23             b = b - lr*bE;
24             printf("w   = %6.3f, b   = %6.3f\n", w, b);
```

```
25
26                  }
27
28          }
29
30  }
31
32  void  setup() {
33
34          Serial.begin(115200);
35          delay(1000);
36
37          dnn_test();
38
39  }
40
41  void  loop() {
42
43  }
```

08 : epoch값을 0에서 2 미만까지 바꾸어가며 10~24줄을 2회 수행합니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
w   = 10.020, b   =   9.980
w   = 10.020, b   =   9.890
w   =  9.861, b   =   9.731
w   =  9.412, b   =   9.507
w   =  8.580, b   =   9.229
w   =  7.358, b   =   8.924
w   =  7.393, b   =   8.888
w   =  7.393, b   =   8.809
w   =  7.271, b   =   8.687
w   =  6.947, b   =   8.525
w   =  6.366, b   =   8.331
w   =  5.534, b   =   8.123
```

학습 회수에 따라 w, b값이 바뀌는 것을 확인합니다.

반복 학습 20회 수행하기

여기서는 반복 학습 20회를 수행해 봅니다.

1. 다음과 같이 예제를 수정합니다.

423_5.ino

```
01 double xs[] = {-1, 0, 1, 2,  3,  4};
02 double ys[] = {-2, 1, 4, 7, 10, 13};
03 double w = 10;
04 double b = 10;
05
06 void dnn_test() {
07
08     for(int epoch=0;epoch<20;epoch++) {
09
10         for(int n=0;n<6;n++) {
11
12             double y = xs[n]*w + 1*b;
13
14             double yT = ys[n];
15             double E = (y-yT)*(y-yT)/2;
16
17             double yE = y - yT;
18             double wE = yE*xs[n];
19             double bE = yE*1;
20
21             double lr = 0.01;
22             w = w - lr*wE;
23             b = b - lr*bE;
24             if (epoch%2==1 && n==0)
25                 printf("w  = %6.3f, b  = %6.3f\n", w, b);
26
27         }
28
29     }
30
31 }
32
```

```
33 void setup() {
34
35       Serial.begin(115200);
36       delay(1000);
37
38       dnn_test();
39
40 }
41
42 void loop() {
43
44 }
```

08 : epoch값을 0에서 20 미만까지 바꾸어가며 10~25줄을 20회 수행합니다.

24 : epoch값을 2로 나눈 나머지가 1이고 n값이 0일 때 25줄을 수행합니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
w   =   7.393, b   =   8.888
w   =   4.332, b   =   7.464
w   =   2.897, b   =   6.614
w   =   2.247, b   =   6.051
w   =   1.974, b   =   5.636
w   =   1.882, b   =   5.303
w   =   1.875, b   =   5.016
w   =   1.907, b   =   4.760
w   =   1.956, b   =   4.526
w   =   2.011, b   =   4.309
```

학습 회수에 따라 w, b값이 바뀌는 것을 확인합니다.

반복 학습 200회 수행하기

여기서는 반복 학습 200회를 수행해 봅니다.

1. 다음과 같이 예제를 수정합니다.

225_8.py

```
08        for(int epoch=0;epoch<200;epoch++) {
```

08 : epoch값을 0에서 200 미만까지 바꾸어가며 10~25줄을 200회 수행합니다.

```
24                  if (epoch%20==1 && n==0)
25                        printf("w  = %6.3f, b  = %6.3f\n", w, b);
```

24 : epoch값을 20으로 나눈 나머지가 1이고 n값이 0일 때 25줄을 수행합니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
w  =  7.393, b  =  8.888
w  =  2.067, b  =  4.107
w  =  2.499, b  =  2.660
w  =  2.732, b  =  1.887
w  =  2.857, b  =  1.474
w  =  2.923, b  =  1.253
w  =  2.959, b  =  1.136
w  =  2.978, b  =  1.072
w  =  2.988, b  =  1.039
w  =  2.994, b  =  1.021
```

학습 회수에 따라 w, b값이 바뀌는 것을 확인합니다. w값은 3에, b값은 1에 가까워지는 것을 확인합니다.

반복 학습 2000회 수행하기

여기서는 반복 학습 2000회를 수행해 봅니다.

1. 다음과 같이 예제를 수정합니다.

225_8.py

```
08        for(int epoch=0;epoch<2000;epoch++) {
```

08 : epoch값을 0에서 2000 미만까지 바꾸어가며 10~25줄을 2000회 수행합니다.

```
24                  if (epoch%200==1 && n==0)
25                        printf("w  = %6.3f, b  = %6.3f\n", w, b);
```

24 : epoch값을 200으로 나눈 나머지가 1이고 n값이 0일 때 25줄을 수행합니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
w   =   7.393, b   =   8.888
w   =   2.997, b   =   1.011
w   =   3.000, b   =   1.000
w   =   3.000, b   =   1.000
w   =   3.000, b   =   1.000
w   =   3.000, b   =   1.000
w   =   3.000, b   =   1.000
w   =   3.000, b   =   1.000
w   =   3.000, b   =   1.000
w   =   3.000, b   =   1.000
```

학습 회수에 따라 w, b값이 바뀌는 것을 확인합니다. w값은 3에 b값은 1에 수렴하는 것을 확인합니다.

가중치, 편향 바꿔보기 1

여기서는 가중치와 편향 값을 바꾸어 실습을 진행해 봅니다.

1. 다음과 같이 예제를 수정합니다.

225_9.py

```
03 double w = -10;
04 double b = 10;
```

03 : 가중치 w값을 -10으로 바꿉니다.
04 : 편향 b값은 10으로 둡니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
w   = -6.877, b   = 10.358
w   =  2.993, b   =  1.022
w   =  3.000, b   =  1.000
w   =  3.000, b   =  1.000
w   =  3.000, b   =  1.000
w   =  3.000, b   =  1.000
w   =  3.000, b   =  1.000
w   =  3.000, b   =  1.000
w   =  3.000, b   =  1.000
w   =  3.000, b   =  1.000
```

w값은 3에 b값은 1에 수렴하는 것을 확인합니다.

가중치, 편향 바꿔보기 2

1. 다음과 같이 예제를 수정합니다.

225_10.py

```
03 double w = -100;
04 double b = 200;
```

03 : 가중치 w값을 -100으로 바꿉니다.
04 : 편향 b값은 200으로 바꿉니다.

2. 업로드를 수행합니다.

3. 다음은 실행 결과 화면입니다.

```
w  = -83.789, b  = 194.356
w  =  2.884, b  =  1.385
w  =  3.000, b  =  1.001
w  =  3.000, b  =  1.000
w  =  3.000, b  =  1.000
w  =  3.000, b  =  1.000
w  =  3.000, b  =  1.000
w  =  3.000, b  =  1.000
w  =  3.000, b  =  1.000
w  =  3.000, b  =  1.000
```

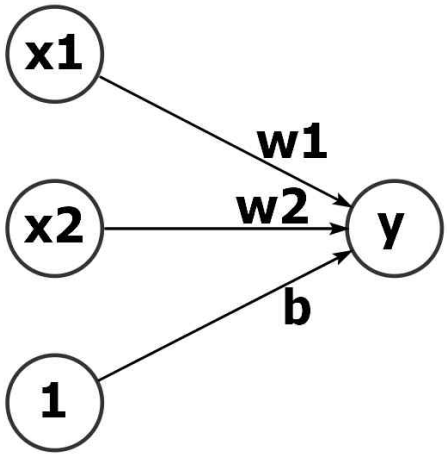
w값은 3에 b값은 1에 수렴하는 것을 확인합니다.

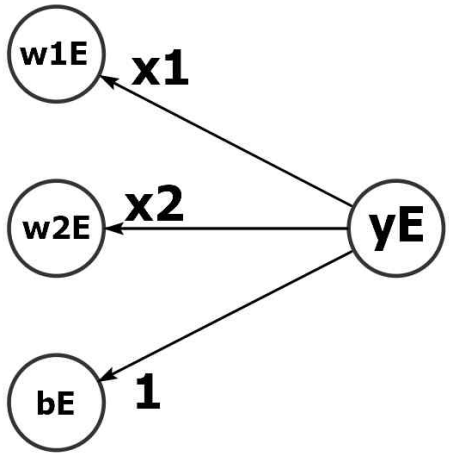
03 다양한 인공 신경망 구현해 보기

우리는 앞에서 입력1 출력1로 구성된 인공 신경의 동작을 살펴보고 구현해 보았습니다. 여기서는 입력2 출력1의 단일 인공 신경과 입력2 출력2로 구성된 인공 신경망의 구조를 살펴보고 수식을 세운 후, 해당 수식에 맞는 인공 신경망을 구현해 봅니다. 출력의 개수가 2이상은 인공 신경망이 됩니다.

01 2입력 1출력 인공 신경 구현하기

다음 그림은 입력2 출력1로 구성된 인공 신경의 순전파와 역전파를 나타냅니다.





y=x1*w1+x2*w2+1*b

w1E=yE*x1
w2E=yE*x2
bE=yE*1

입력2 출력1로 구성된 인공 신경의 수식을 정리하면 다음과 같습니다.

순전파

$$y = x_1{}^* w_1 + x_2{}^* w_2 + 1{}^* b$$

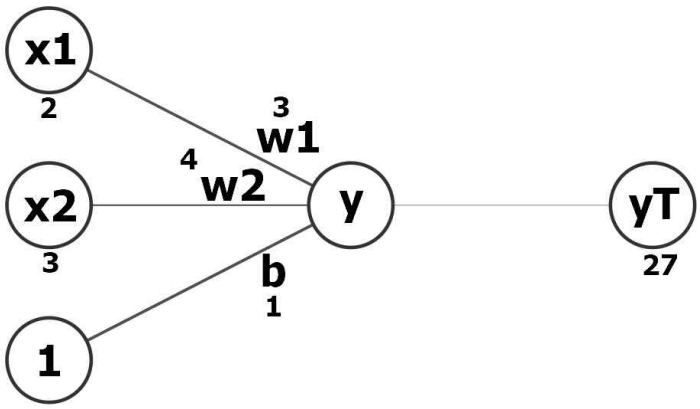
역전파

$$w_{1E} = y_E{}^* x_1$$
$$w_{2E} = y_E{}^* x_2$$
$$b_E = y_E{}^* 1$$

인공 신경망 학습

$$w_1 = w_1 - \alpha w_{1E}$$
$$w_2 = w_2 - \alpha w_{2E}$$
$$b = b - \alpha b_E$$

지금까지 정리한 수식을 구현을 통해 살펴봅니다. 다음 그림을 살펴봅니다.



이 그림에서 입력값 x1, x2는 각각 2, 3, 가중치 w1, w2는 각각 3, 4, 편향 b는 1이고 목표값 yT는 27입니다. x1, x2, yT를 이용하여 w1, w2, b에 대해 학습을 수행해 봅니다.

*** 이 값들은 임의의 값들입니다. 다른 값들을 사용하여 학습을 수행할 수도 있습니다.

1. 422_3.ino를 최종 수정한 예제를 431.ino로 저장합니다.

2. 다음과 같이 예제를 수정합니다.
431.ino

```
01 double x1 =2, x2 = 3;
02 double yT = 27;
03 double w1 = 3;
04 double w2 = 4;
05 double b = 1;
06
07 void dnn_test() {
08
09       for(int epoch=0;epoch<2000;epoch++) {
10
11                printf("epoch = %d\n", epoch);
```

```
12
13              double y = x1*w1 + x2*w2 + 1*b;
14              printf(" y  = %6.3f\n", y);
15
16              double E = (y-yT)*(y-yT)/2;
17              printf(" E  = %.7f\n", E);
18              if(E < 0.0000001)
19                      break;
20
21              double yE = y - yT;
22
23              double w1E = yE*x1;
24              double w2E = yE*x2;
25              double bE = yE*1;
26
27              printf(" w1E, w2E, bE = %6.3f, %6.3f, %6.3f\n", w1E, w2E, bE);
28
29              double lr = 0.01;
30              w1 = w1 - lr*w1E;
31              w2 = w2 - lr*w2E;
32              b = b - lr*bE;
33              printf(" w1,  w2,  b  = %6.3f, %6.3f, %6.3f\n", w1, w2, b);
34
35      }
36
37 }
38
39 void setup() {
40
41      Serial.begin(115200);
42      delay(1000);
43
44      dnn_test();
45
46 }
47
48 void loop() {
49
50 }
```

3. 업로드를 수행합니다.

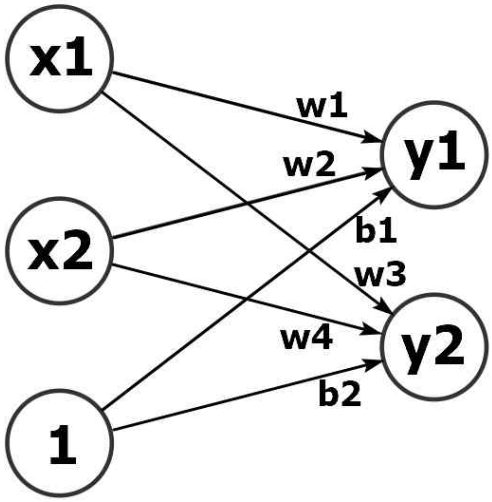
4. 다음은 실행 결과 화면입니다.

```
epoch = 64
 y   = 26.999
 E   = 0.0000001
 w1E, w2E, bE = -0.001, -0.002, -0.001
 w1,  w2,  b  =  4.143,  5.714,  1.571
epoch = 65
 y   = 27.000
 E   = 0.0000001
```

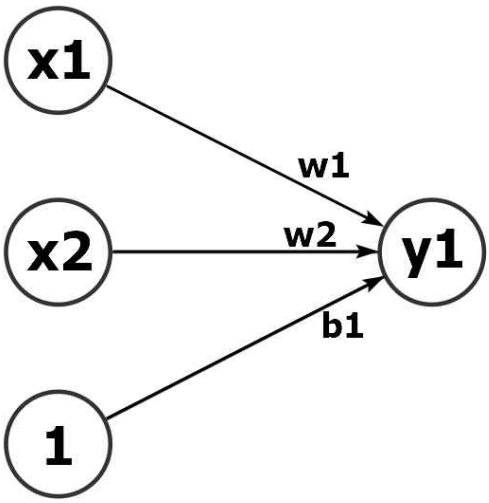
(65+1)회 째 학습이 완료되는 것을 볼 수 있습니다. 가중치 w1, w2는 각각 4.143, 5.714, 편향 b는 1.571에 수렴합니다.

02 2입력 2출력 인공 신경망 구현하기

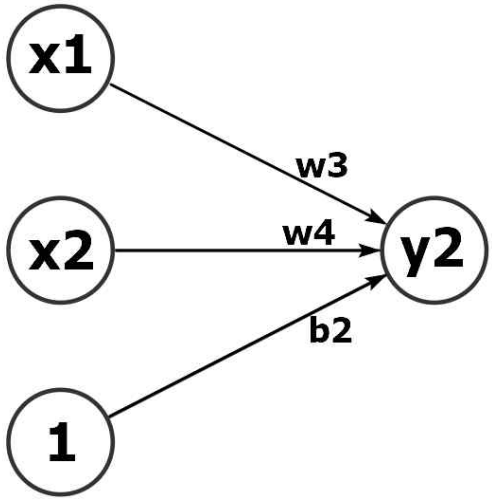
다음 그림은 입력2 출력2로 구성된 인공 신경망의 순전파를 나타냅니다.



이 인공 신경망은 다음과 같이 입력2 출력1 인공 신경 2개로 구성됩니다.

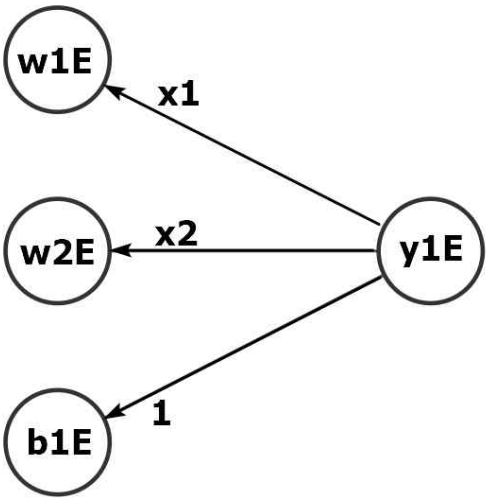


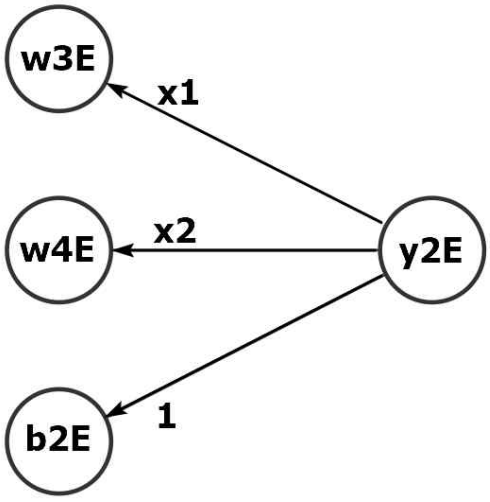
y1=x1*w1+x2*w2+1*b1



y2=x1*w3+x2*w4+1*b2

다음 그림은 이 인공 신경망의 역전파를 나타냅니다.





w1E=y1E*x1
w2E=y1E*x2
b1E=y1E*1

w3E=y2E*x1
w4E=y2E*x2
b2E=y2E*1

입력2 출력2로 구성된 인공 신경의 수식을 정리하면 다음과 같습니다.

순전파

$$y_1 = x_1 * w_1 + x_2 * w_2 + 1 * b_1$$
$$y_2 = x_1 * w_3 + x_2 * w_4 + 1 * b_2$$

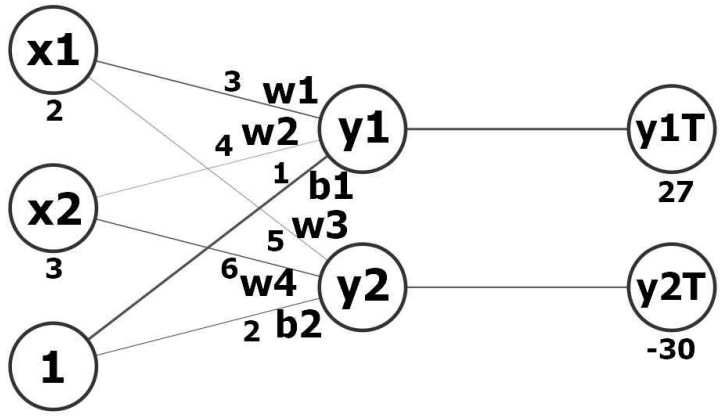
역전파

$$w_{1E} = y_{1E} * x_1$$
$$w_{2E} = y_{1E} * x_2$$
$$w_{3E} = y_{2E} * x_1$$
$$w_{4E} = y_{2E} * x_2$$
$$b_{1E} = y_{1E} * 1$$
$$b_{2E} = y_{2E} * 1$$

인공 신경망 학습

$$
\begin{aligned}
w_1 &= w_1 - \alpha w_{1E} \\
w_2 &= w_2 - \alpha w_{2E} \\
w_3 &= w_3 - \alpha w_{3E} \\
w_4 &= w_4 - \alpha w_{4E} \\
b_1 &= b_1 - \alpha b_{1E} \\
b_2 &= b_2 - \alpha b_{2E}
\end{aligned}
$$

지금까지 정리한 수식을 구현을 통해 살펴봅니다. 다음 그림을 살펴봅니다.



이 그림에서 입력값 x1, x2는 각각 2, 3, 가중치 w1, w2, 편향 b1은 각각 3, 4, 1, 가중치 w3, w4, 편향 b2는 각각 5, 6, 2이고 목표값 y1T, y2T는 각각 27, -30입니다. x1, x2, y1T, y2T 를 이용하여 w1, w2, b1, w3, w4, b2에 대해 학습을 수행해 봅니다.

*** 이 값들은 임의의 값들입니다. 다른 값들을 사용하여 학습을 수행할 수도 있습니다.

1. 이전 예제 431.ino를 432.ino로 저장합니다.

2. 다음과 같이 예제를 수정합니다.
432.ino

```
01 double x1 =2, x2 = 3;
02 double y1T = 27, y2T = -30;
03 double w1 = 3, w3 = 5;
04 double w2 = 4, w4 = 6;
05 double b1 = 1, b2 = 2;
06
07 void dnn_test() {
08
```

```
09      for(int epoch=0;epoch<2000;epoch++) {
10
11              printf("epoch = %d\n", epoch);
12
13              double y1 = x1*w1 + x2*w2 + 1*b1;
14              double y2 = x1*w3 + x2*w4 + 1*b2;
15              printf(" y1,  y2  = %6.3f, %6.3f\n", y1, y2);
16
17              double E = (y1-y1T)*(y1-y1T)/2 + (y2-y2T)*(y2-y2T)/2;
18              printf(" E  = %.7f\n", E);
19              if(E < 0.0000001)
20                      break;
21
22              double y1E = y1 - y1T, y2E = y2 - y2T;
23
24              double w1E = x1*y1E, w3E = x1*y2E;
25              double w2E = x2*y1E, w4E = x2*y2E;
26              double b1E = 1*y1E, b2E = 1*y2E;
27              printf(" w1E, w3E = %6.3f, %6.3f\n", w1E, w3E);
28              printf(" w2E, w4E = %6.3f, %6.3f\n", w2E, w4E);
29              printf(" b1E, b2E = %6.3f, %6.3f\n", b1E, b2E);
30
31              double lr = 0.01;
32              w1 = w1 - lr*w1E, w3 = w3 -lr*w3E;
33              w2 = w2 - lr*w2E, w4 = w4 -lr*w4E;
34              b1 = b1 - lr*b1E, b2 = b2 - lr*b2E;
35              printf(" w1,  w3  = %6.3f, %6.3f\n", w1, w3);
36              printf(" w2,  w4  = %6.3f, %6.3f\n", w2, w4);
37              printf(" b1,  b2  = %6.3f, %6.3f\n", b1, b2);
38
39      }
40
41 }
42
43 void setup() {
44
45      Serial.begin(115200);
46      delay(1000);
47
48      dnn_test();
```

```
49
50 }
51
52 void loop() {
53
54 }
```

3. 업로드를 수행합니다.

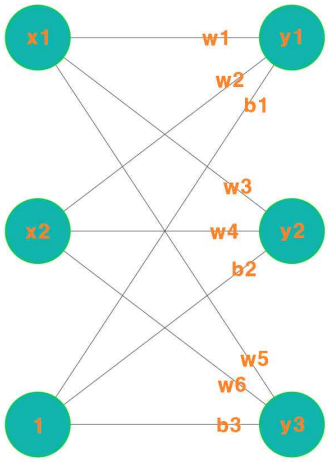
4. 다음은 실행 결과 화면입니다.

```
epoch = 78
 y1,  y2  = 27.000, -30.000
 E   = 0.0000001
 w1E, w3E = -0.000,  0.001
 w2E, w4E = -0.000,  0.001
 b1E, b2E = -0.000,  0.000
 w1,  w3  =  4.143, -3.571
 w2,  w4  =  5.714, -6.857
 b1,  b2  =  1.571, -2.286
epoch = 79
 y1,  y2  = 27.000, -30.000
 E   = 0.0000001
```

(79+1)회 째 학습이 완료되는 것을 볼 수 있습니다. 가중치 w1, w2는 각각 4.143, 5.714, 편향 b1은 1.571, 가중치 w3, w4는 각각 -3.571, -6.857 편향 b2는 -2.286에 수렴합니다.

연습문제

1. 다음은 입력2 출력3의 인공 신경망입니다. 이 인공 신경의 순전파, 역전파 수식을 구합니다.



2. 앞에서 구한 수식을 이용하여 다음과 같이 초기화된 인공 신경망을 구현하고 학습시켜 봅니다.