**3 X 3 포워드 계산**

201801733 김선우

**C 언어 코드**

#include <stdio.h>

#include <math.h>

int main()

{

   double input[3] = {0.9, 0.1, 0.8};

   double Winput\_hidden[3][3] = {{0.9, 0.3, 0.4}, {0.2, 0.8, 0.2}, {0.1, 0.5, 0.6}};

   double Whidden\_output[3][3] = {{0.3, 0.7, 0.5}, {0.6, 0.5, 0.2}, {0.8, 0.1, 0.9}};

   double Xhidden[3] = {0,};

   double Ohidden[3] = {0,};

   double Xoutput[3] = {0,};

   double Ooutput[3] = {0,};

   // Weight input to hidden

   for(int i = 0; i < 3; i++)

   {

      for(int j = 0; j < 3; j++)

      {

         Xhidden[i] += input[j] \* Winput\_hidden[i][j];

      }

      Ohidden[i] = 1 / (1 + exp(-Xhidden[i]));

   }

   // Weight hidden to output

   for(int i = 0; i < 3; i++)

   {

      for(int j = 0; j < 3; j++)

      {

         Xoutput[i] += Ohidden[j] \* Whidden\_output[i][j];

      }

      Ooutput[i] = 1 / (1 + exp(-Xoutput[i]));

   }

   printf("input     Ohidden     Ooutput\n");

   for(int i = 0; i < 3; i++)

   {

      printf("%.2f  --> %.4f  --> %.4f\n", input[i], Ohidden[i], Ooutput[i]);

   }

   return 0;

}

**실행 결과**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**가중치 값 바꿔보기**

가중치의 값이 커질수록 입력 값에 대한 반영 비율이 커진다고 볼 수 있다.

시그모이드 함수 에서 는 x 의 값이 커질수록 작아진다.

가 작아질수록 시그모이드 함수의 결과값은 커지게 된다.

즉, x 의 값이 커지면 시그모이드 함수의 결과는 커지게 되는 것이다.

이를 확인해보기 위해 하나의 퍼셉트론에 들어오는 가중치가 전부 작을 때와

전부 클 때 그리고 hidden 레이어와 output 레이어의 가중치의 순서를 바꿔서 실행해보았다.

double Winput\_hidden[3][3] = {{0.9, 0.8, 0.9}, {0.1, 0.2, 0.3}, {0.8, 0.1, 0.9}};

double Whidden\_output[3][3] = {{0.8, 0.7, 0.9}, {0.1, 0.2, 0.1}, {0.1, 0.5, 0.6}};

각 Layer 에서 첫번째에 해당하는 노드에는 높은 가중치를 두번째에 해당하는 노드에는 낮은   
가중치를 세번째에 해당하는 노드의 가중치는 hidden 과 output 에서의 가중치 순서를 바꿨다.

**텍스트이(가) 표시된 사진

자동 생성된 설명** 텍스트이(가) 표시된 사진

자동 생성된 설명

왼쪽이 원래 결과값이고 오른쪽이 가중치를 바꾼 결과값이다.

예상했던 것처럼 가중치를 높이게 될 경우 시그모이드 함수의 결과값이 커지기 때문에

출력 값이 전보다 커지게 되고 가중치를 작게 줄 경우 결과값이 전보다 작아진 것을 알 수 있다.

세번째의 경우 가중치의 순서를 바꾸게 된다고 해서 Layer 의 출력 결과 값이 서로 바뀌는 것이

아닌 Layer들을 거치면서 수행하는 수식의 순서가 완전히 달라지므로 이전과는 다른 결과값이

나오게 된다.

**파이썬 코드**

import math

input = [0.9, 0.1, 0.8]

Winput\_hidden = [[0.9, 0.3, 0.4], [0.2, 0.8, 0.2], [0.1, 0.5, 0.6]]

Whidden\_output = [[0.3, 0.7, 0.5], [0.6, 0.5, 0.2], [0.8, 0.1, 0.9]]

Xhidden = []

Ohidden = []

Xoutput = []

Ooutput = []

# Weight input to hidden

for i in range(3):

    temp = 0

    for j in range(3):

        temp += input[j] \* Winput\_hidden[i][j]

    Xhidden.append(temp)

    Ohidden.append(1 / (1 + math.exp(-Xhidden[i])))

# Weight hidden to output

for i in range(3):

    temp = 0

    for j in range(3):

        temp += Ohidden[j] \* Whidden\_output[i][j]

    Xoutput.append(temp)

    Ooutput.append(1 / (1 + math.exp(-Xoutput[i])))

print(" input    Ohidden    Ooutput")

for i in range(3):

    print(f"{input[i] : 0.1f} --> {Ohidden[i] : 0.3f} --> {Ooutput[i] : 0.3f}")