202001555 지은미

In [84]:
```
# 1번 (Forward propagation)
import numpy as np

def sigmoid(x):
    return 1/(1+np.exp(-x))


w_input = np.array([[0.9,0.3,0.4],[0.2,0.8,0.2],[0.1,0.5,0.6]]) # input layer -> hidde
w_out=np.array([[0.3,0.7,0.5],[0.6,0.5,0.2],[0.8,0.1,0.9]])  # hidden layer -> out lay
i_input=np.array([0.9,0.1,0.8], ndmin=2).T  # input 전치행렬
print("i_input")
print(i_input)

x_hidden_input=np.dot(w_input,i_input)  # x_hidden = w_input * input 내적 -> 은닉계층
print("x_hidden")
print(x_hidden_input)

o_hidden_output = sigmoid(x_hidden_input) # 은닉계층의 결과값
print("o_hidden_output")
print(o_hidden_output)

x_out = np.dot(w_out,o_hidden_output)
print("x_out")
print(x_out)

o_out = sigmoid(x_out)
print("o_out")
print(o_out)
```

```
i_input
[[0.9]
 [0.1]
 [0.8]]
x_hidden
[[1.16]
 [0.42]
 [0.62]]
o_hidden_output
[[0.76133271]
 [0.60348325]
 [0.65021855]]
x_out
[[0.97594736]
 [0.88858496]
 [1.25461119]]
o_out
[[0.72630335]
 [0.70859807]
 [0.77809706]]
```

In [85]:
```
# 2번 (Backward propagation)
e=np.array([0.8,0.5],ndmin=2) # 오차
w_hidden = np.array([[3.0,2.0],[1.0,7.0]], ndmin=2) # layer1,2 가중치
print("w_hidden")
print(w_hidden)
w_output = np.array([[2.0,3.0],[1.0,4.0]], ndmin=2) # layer2,3 가중치
print("w_output")
print(w_output)

# hidden
w_output_sum=w_output.sum(axis=1, dtype="float") # e_hidden 구하기 : 분모의 합
e_output_d=w_output.T/w_output_sum
e_output=np.dot(e_output_d,e.T)
```

```
print("e_output")
print(e_output)

#output
w_hidden_sum=w_hidden.sum(axis=1, dtype="float") # e_output 구하기 :분모의 합
e_hidden_d=w_hidden.T/w_hidden_sum
e_hidden=np.dot(e_hidden_d,e_output)
print("e_hidden")
print(e_hidden)
```

```
w_hidden
[[3. 2.]
 [1. 7.]]
w_output
[[2. 3.]
 [1. 4.]]
e_output
[[0.42]
 [0.88]]
e_hidden
[[0.362]
 [0.938]]
```

In [86]:

```
# 3번 (Weight Update)
study = 0.1
e_output=np.array([0.8,0.5],ndmin=2).T
e_input=np.array([0.42,0.88], ndmin=2).T
w_output = np.array([[2.0,3.0],[1.0,4.0]], ndmin=2)
w_hidden = np.array([[3.0,2.0],[1.0,7.0]], ndmin=2)
o_output=np.array([0.4,0.5],ndmin=2).T
o_input=np.array([-0.14938188,0.02134027], ndmin=2).T

#은닉계층과 출력계층 사이의 가중치 업데이트
weightoutput_o_sum = np.dot(w_output,o_output)
print("weightoutput_o_sum")
print(weightoutput_o_sum)
sig_output=sigmoid(weightoutput_o_sum)*(1-sigmoid(weightoutput_o_sum))
print("sig_output")
print(sig_output)
m_output=np.dot(-e_output*sig_output,o_output.T)
print("m_output")
print(m_output)
study_output = study*m_output
print("study_output")
print(study_output)
weightoutput_update = w_output - study_output
print("weightoutput_update")
print(weightoutput_update)


#입력계층과 은닉계층 사이의 가중치 업데이트
weightinput_o_sum = np.dot(w_hidden,o_input)
print("weightinput_o_sum")
print(weightinput_o_sum)
sig_input=sigmoid(weightinput_o_sum)*(1-sigmoid(weightinput_o_sum))
print("sig_input")
print(sig_input)
m_input=np.dot(-e_input*sig_input,o_input.T)
print("m_input")
print(m_input)
study_input = study*m_input
print("study_input")
print(study_input)
weightinput_update = w_hidden - study_input
print("weightinput_update")
print(weightinput_update)
```

```
weightoutput_o_sum
[[2.3]
 [2.4]]
sig_output
[[0.08281957]
 [0.076255  ]]
m_output
[[-0.02650226 -0.03312783]
 [-0.015251   -0.01906375]]
study_output
[[-0.00265023 -0.00331278]
 [-0.0015251  -0.00190637]]
weightoutput_update
[[2.00265023 3.00331278]
 [1.0015251  4.00190637]]
weightinput_o_sum
[[-4.054651e-01]
 [ 1.000000e-08]]
sig_input
[[0.24]
 [0.25]]
m_input
[[ 0.01505769 -0.0021511 ]
 [ 0.03286401 -0.00469486]]
study_input
[[ 0.00150577 -0.00021511]
 [ 0.0032864  -0.00046949]]
weightinput_update
[[2.99849423 2.00021511]
 [0.9967136  7.00046949]]
```

In [88]:

```
# 4번(역전파 오차 값과 가중치 업데이트 계산)
def forward_propagation(i,w):
    x=np.dot(w,i)
    o=sigmoid(x)
    return o

def back_propagation(w,e):
    w_sum=w.sum(axis=1, dtype="float")
    e_divide=w.T/w_sum
    e_result=np.dot(e_divide,e.T)
    return e_result

def weight_update(w,o,e,study):
    weight_sum=np.dot(w,o)
    sig=sigmoid(weight_sum)*(1-sigmoid(weight_sum))
    m=np.dot(-e*sig,o.T)
    result=w-(study*m)
    return result

targets=np.array([[0.01],[0.01],[0.99]]) # 목표값
w_input = np.array([[0.9,0.3,0.4],[0.2,0.8,0.2],[0.1,0.5,0.6]]) # input layer -> hidde
w_out=np.array([[0.3,0.7,0.5],[0.6,0.5,0.2],[0.8,0.1,0.9]])  # hidden layer -> out lay
i_input=np.array([0.9,0.1,0.8], ndmin=2).T  # input
o_input=np.array([0.4,0.5,0.6], ndmin=2).T
o_ouput=np.array([0.1,0.2,0.3],ndmin=2).T
study=0.1 #학습률

# forwardpropergation
hidden_result=forward_propagation(i_input,w_input) # input layer -> hidden layer
print("hidden_result")
print(hidden_result)
output_result=forward_propagation(hidden_result,w_out) # hidden layer -> out layer
print("output_result")
print(output_result)

e_output=targets - output_result  #오차구하기
```

```
print("e_output")
print(e_output)

#backpropagation
error_out=back_propagation(w_out,e_output.T) #은닉계층과 출력계층 사이의 오차값
print("error_out")
print(error_out)

error_input=back_propagation(w_input,error_out.T) #입력계층과 은닉계층 사이의 오차값
print("error_input")
print(error_input)

#weight update
update_out=weight_update(w_out,o_ouput,e_output,study) # 은닉계층과 출력계층 사이의 기
print("update_out")
print(update_out)
```

```
hidden_result
[[0.76133271]
 [0.60348325]
 [0.65021855]]
output_result
[[0.72630335]
 [0.70859807]
 [0.77809706]]
e_output
[[-0.71630335]
 [-0.69859807]
 [ 0.21190294]]
error_out
[[-0.37151146]
 [-0.59119408]
 [-0.24029294]]
error_input
[[-0.32753196]
 [-0.56390984]
 [-0.31155668]]
update_out
[[0.29825431 0.69650863 0.49476294]
 [0.59827447 0.49654894 0.1948234 ]
 [0.80051203 0.10102406 0.9015361 ]]
```

202001555 지은미