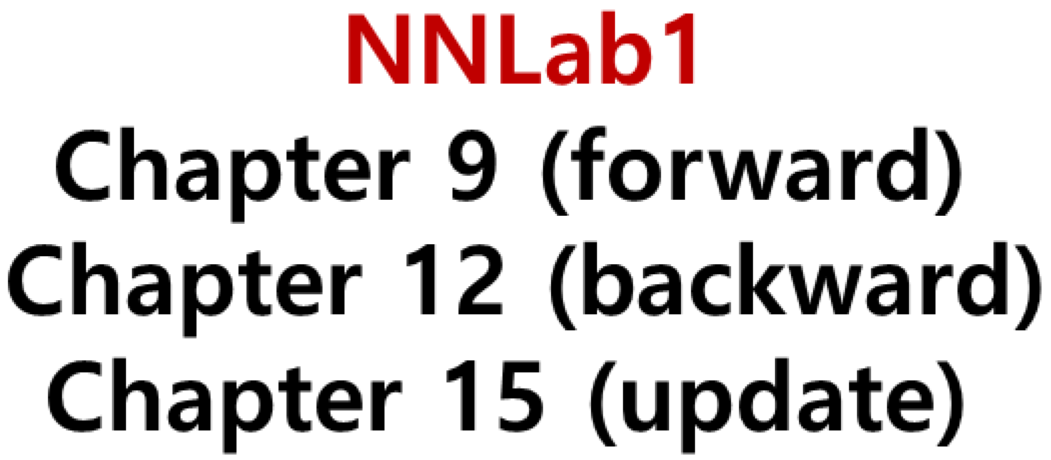
201600282 엄기산

```
In [1]: from IPython.core.interactiveshell import InteractiveShell
        InteractiveShell. ast_node_interactivity = "all"
```

NNLab1♂Chapter 9 (forward) ♂Chapter 12 (backward)♂Chapter 15 (update)



NNLab1

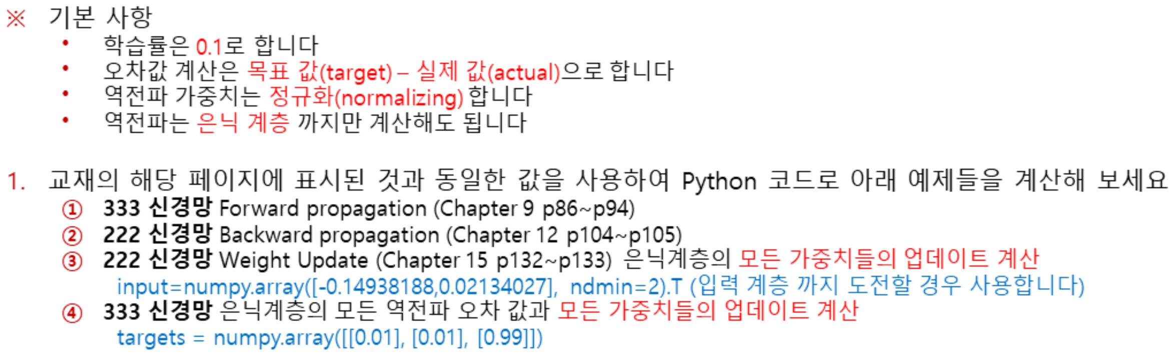
Chapter 9 (forward)

Chapter 12 (backward)

Chapter 15 (update)

기본 사항 학습률은 0.1로 합니다 오차값 계산은 목표 값(target) – 실제 값(actual)으로 합니다 역전파 가중치는 정규화(normalizing) 합니다 역전파는 은닉 계층 까지만 계산해도 됩니다

교재의 해당 페이지에 표시된 것과 동일한 값을 사용하여 Python 코드로 아래 예제들을 계산해 보세요 333 신경망 Forward propagation (Chapter 9 p86~p94) 222 신경망 Backward propagation (Chapter 12 p104~p105) 222 신경망 Weight Update (Chapter 15 p132~p133) 은닉 계층의 모든 가중치들의 업데이트 계산 input=numpy.array([-0.14938188,0.02134027], ndmin=2).T (입력 계층 까지 도전할 경우 사용합니다) 333 신경망 은닉계층의 모든 역전파 오차 값과 모든 가중치들의 업데이트 계산 targets = numpy.array([[0.01], [0.01], [0.99]])



※ 기본 사항

- 학습률은 0.1로 합니다
- 오차값 계산은 목표 값(target) – 실제 값(actual)으로 합니다
- 역전파 가중치는 정규화(normalizing) 합니다
- 역전파는 은닉 계층 까지만 계산해도 됩니다

1. 교재의 해당 페이지에 표시된 것과 동일한 값을 사용하여 Python 코드로 아래 예제들을 계산해 보세요
   ① **333 신경망** Forward propagation (Chapter 9 p86~p94)
   ② **222 신경망** Backward propagation (Chapter 12 p104~p105)
   ③ **222 신경망** Weight Update (Chapter 15 p132~p133) 은닉계층의 모든 가중치들의 업데이트 계산
   input=numpy.array([-0.14938188,0.02134027], ndmin=2).T (입력 계층 까지 도전할 경우 사용합니다)
   ④ **333 신경망** 은닉계층의 모든 역전파 오차 값과 모든 가중치들의 업데이트 계산
   targets = numpy.array([[0.01], [0.01], [0.99]])

1. 333신경망 Forward propagation

```
In [3]: import numpy as np
        import math
```

```
In [4]: I = np. array([[0.9],
        [0.1],
        [0.8]])
        Winput_hidden = np. array([[0.9, 0.3, 0.4],
        [0.2, 0.8, 0.2],
        [0.1, 0.5, 0.6]])
```

```
Whidden_output = np. array([[0.3, 0.7, 0.5],
[0.6, 0.5, 0.2],
[0.8, 0.1, 0.9]])
I
Winput_hidden
Whidden_output
```

```
Out[4]: array([[0.9],
               [0.1],
               [0.8]])

Out[4]: array([[0.9, 0.3, 0.4],
               [0.2, 0.8, 0.2],
               [0.1, 0.5, 0.6]])

Out[4]: array([[0.3, 0.7, 0.5],
               [0.6, 0.5, 0.2],
               [0.8, 0.1, 0.9]])
```

In [6]:

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

In [7]:

```
X_hidden = np. dot(Winput_hidden, I)
O_hidden = np. array([[sigmoid(X_hidden[0])],
[sigmoid(X_hidden[1])],
[sigmoid(X_hidden[2])]])
X_hidden
O_hidden
```

```
Out[7]: array([[1.16],
               [0.42],
               [0.62]])

Out[7]: array([[0.76133271],
               [0.60348325],
               [0.65021855]])
```

In [8]:

```
X_output = np. dot(Whidden_output, O_hidden)
O_output = np. array([[sigmoid(X_output[0])],
[sigmoid(X_output[1])],
[sigmoid(X_output[2])]])
X_output
O_output
```

```
Out[8]: array([[0.97594736],
               [0.88858496],
               [1.25461119]])

Out[8]: array([[0.72630335],
               [0.70859807],
               [0.77809706]])
```

2. 222 NN Backward Propagation

In [9]:

```
Winput_hidden = np. array([[3.0, 2.0],
[1.0, 7.0]])
Whidden_output = np. array([[2.0, 3.0],
[1.0, 4.0]])
Winput_hidden
Whidden_output
```

```
Out[9]: array([[3., 2.],
               [1., 7.]])

Out[9]: array([[2., 3.],
               [1., 4.]])
```

In [10]:

```
e_output = np. array([[0.8],
[0.5]])
e_hidden = np. dot(np. transpose(Whidden_output), e_output)
e_output
e_hidden
```

Out[10]:
```
array([[0.8],
       [0.5]])
```

Out[10]:
```
array([[2.1],
       [4.4]])
```

In [11]:
```
e_input = np. dot(np. transpose(Winput_hidden), e_hidden)
e_input
```

Out[11]:
```
array([[10.7],
       [35. ]])
```

3. 222 NN Weight Update

In [14]:
```
shidden_output_1 = sigmoid(2.0 * 0.4 + 3.0 * 0.5)
dedw_hidden_output_1 = - 0.8 * shidden_output_1 * (1 - shidden_output_1) * 0.4
dedw_hidden_output_1
```

Out[14]:
```
-0.02650226143703718
```

In [16]:
```
learning_rate = 0.1
new_w_hidden_output_1 = 2.0 - learning_rate * dedw_hidden_output_1
new_w_hidden_output_1
```

Out[16]:
```
2.002650226143704
```

4. 333 NN Backward Propagation & Weight Update

In [17]:
```
I = np. array([[0.9],
[0.1],
[0.8]])
Winput_hidden = np. array([[0.9, 0.3, 0.4],
[0.2, 0.8, 0.2],
[0.1, 0.5, 0.6]])
Whidden_output = np. array([[0.3, 0.7, 0.5],
[0.6, 0.5, 0.2],
[0.8, 0.1, 0.9]])
I
Winput_hidden
Whidden_output
```

Out[17]:
```
array([[0.9],
       [0.1],
       [0.8]])
```

Out[17]:
```
array([[0.9, 0.3, 0.4],
       [0.2, 0.8, 0.2],
       [0.1, 0.5, 0.6]])
```

Out[17]:
```
array([[0.3, 0.7, 0.5],
       [0.6, 0.5, 0.2],
       [0.8, 0.1, 0.9]])
```

In [19]:
```
target = np. array([[0.01],
[0.01],
[0.99]])
```

```
target
O_output
```

Out[19]:
```
array([[0.01],
       [0.01],
       [0.99]])
```

Out[19]:
```
array([[0.72630335],
       [0.70859807],
       [0.77809706]])
```

In [20]:
```
e_output = target - O_output
e_output
```

Out[20]:
```
array([[-0.71630335],
       [-0.69859807],
       [ 0.21190294]])
```

In [21]:
```
e_hidden = np. dot(np. transpose(Whidden_output), e_output)
e_hidden
```

Out[21]:
```
array([[-0.46452749],
       [-0.82952108],
       [-0.30715864]])
```

In [22]:
```
e_input = np. dot(np. transpose(Winput_hidden), e_hidden)
e_input
```

Out[22]:
```
array([[-0.61469483],
       [-0.95655444],
       [-0.5360104 ]])
```

In [25]:
```
e_output_1, e_output_2, e_output_3 = e_output
e_output_1, e_output_2, e_output_3
e_output

O_output_1, O_output_2, O_output_3 = O_output
O_output_1, O_output_2, O_output_3
O_output
```

Out[25]:
```
(array([-0.71630335]), array([-0.69859807]), array([0.21190294]))
```

Out[25]:
```
array([[-0.71630335],
       [-0.69859807],
       [ 0.21190294]])
```

Out[25]:
```
(array([0.72630335]), array([0.70859807]), array([0.77809706]))
```

Out[25]:
```
array([[0.72630335],
       [0.70859807],
       [0.77809706]])
```

In [28]:
```
Whidden_output

s_hidden_output_1 = sigmoid(np. sum(Whidden_output[0] * O_output_1))
s_hidden_output_2 = sigmoid(np. sum(Whidden_output[1] * O_output_2))
s_hidden_output_3 = sigmoid(np. sum(Whidden_output[2] * O_output_3))
s_hidden_output_1, s_hidden_output_2, s_hidden_output_3

dedw_hidden_output_1 = - e_output_1 * s_hidden_output_1 * (1 - s_hidden_output_1) * O
dedw_hidden_output_2 = - e_output_2 * s_hidden_output_2 * (1 - s_hidden_output_2) * O
dedw_hidden_output_3 = - e_output_3 * s_hidden_output_3 * (1 - s_hidden_output_3) * O
dedw_hidden_output_1, dedw_hidden_output_2, dedw_hidden_output_3

learning_rate = 0.1

Whidden_output[0] = Whidden_output[0] - learning_rate * dedw_hidden_output_1
Whidden_output[1] = Whidden_output[1] - learning_rate * dedw_hidden_output_2
```

```
Whidden_output[2] = Whidden_output[2] - learning_rate * dedw_hidden_output_3
Whidden_output
```

```
Out[28]: array([[0.3, 0.7, 0.5],
                [0.6, 0.5, 0.2],
                [0.8, 0.1, 0.9]])

Out[28]: (0.7482790839619047, 0.7152819674874038, 0.8022750700755457)

Out[28]: (array([0.09799365]), array([0.10081371]), array([-0.02615505]))

Out[28]: array([[0.29020064, 0.69020064, 0.49020064],
                [0.58991863, 0.48991863, 0.18991863],
                [0.8026155 , 0.1026155 , 0.9026155 ]])
```

```
In [29]: e_hidden_1, e_hidden_2, e_hidden_3 = e_hidden
         e_hidden_1, e_hidden_2, e_hidden_3
         e_hidden


         O_hidden_1, O_hidden_2, O_hidden_3= O_hidden
         O_hidden_1, O_hidden_2, O_hidden_3
         O_hidden
```

```
Out[29]: (array([-0.46452749]), array([-0.82952108]), array([-0.30715864]))

Out[29]: array([[-0.46452749],
                [-0.82952108],
                [-0.30715864]])

Out[29]: (array([0.76133271]), array([0.60348325]), array([0.65021855]))

Out[29]: array([[0.76133271],
                [0.60348325],
                [0.65021855]])
```

```
In [30]: Winput_hidden

         s_input_hidden_1 = sigmoid(np. sum(Winput_hidden[0] * O_hidden_1))
         s_input_hidden_2 = sigmoid(np. sum(Winput_hidden[1] * O_hidden_2))
         s_input_hidden_3 = sigmoid(np. sum(Winput_hidden[2] * O_hidden_3))
         s_input_hidden_1, s_input_hidden_2, s_input_hidden_3

         dedw_input_hidden_1 = - e_hidden_1 * s_input_hidden_1 * (1 - s_input_hidden_1) * O_hi
         dedw_input_hidden_2 = - e_hidden_2 * s_input_hidden_2 * (1 - s_input_hidden_2) * O_hi
         dedw_input_hidden_3 = - e_hidden_3 * s_input_hidden_3 * (1 - s_input_hidden_3) * O_hi
         dedw_input_hidden_1, dedw_input_hidden_2, dedw_input_hidden_3

          learning_rate = 0.1
         Winput_hidden[0] = Winput_hidden[0] - learning_rate * dedw_input_hidden_1
         Winput_hidden[1] = Winput_hidden[1] - learning_rate * dedw_input_hidden_2
         Winput_hidden[2] = Winput_hidden[2] - learning_rate * dedw_input_hidden_3
         Winput_hidden
```

```
Out[30]: array([[0.9, 0.3, 0.4],
                [0.2, 0.8, 0.2],
                [0.1, 0.5, 0.6]])

Out[30]: (0.77173470962324, 0.6735267945688745, 0.6857366338512985)

Out[30]: (array([0.06230083]), array([0.11007662]), array([0.04304009]))

Out[30]: array([[0.89376992, 0.29376992, 0.39376992],
                [0.18899234, 0.78899234, 0.18899234],
                [0.09569599, 0.49569599, 0.59569599]])
```

201600282 엄기산