Exercise for back propagation

Figure 9.5 shows a multilayer feed-forward neural network. Let the learning rate be 0.9. The initial weight and bias values of the network are given in Table 9.1, along with the first training tuple, X = (1, 0, 1), with a class label of 1.

This shows the calculations for backpropagation, given the first training tuple, X. The tuple is fed into the network, and the net input and output of each unit are computed

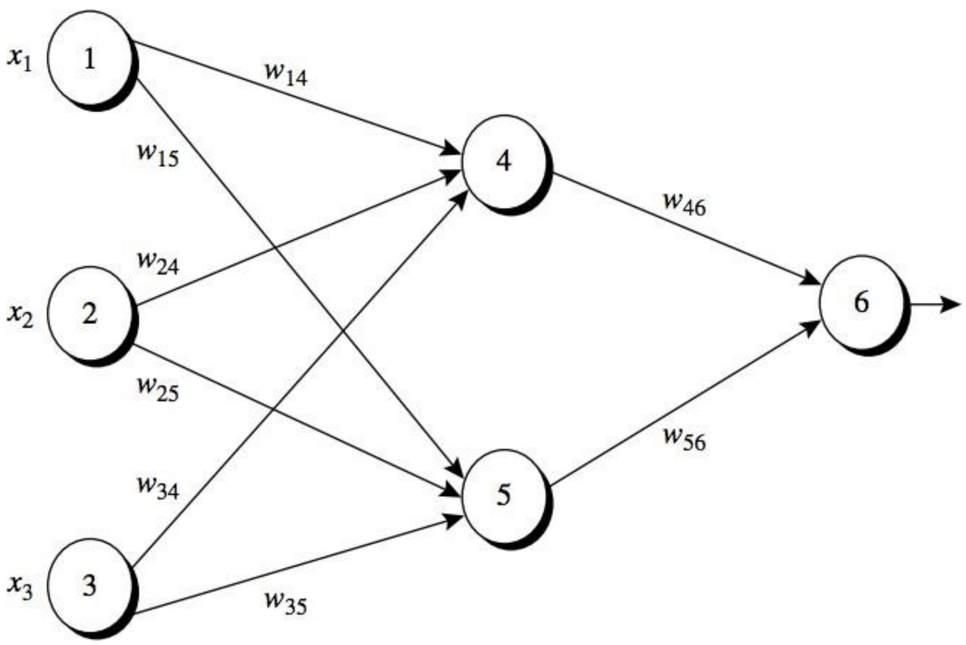

Figure 9.5 Example of a multilayer feed-forward neural network

Table 9.1 Initial Input, Weight, and Bias Values

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

1. Please calculate the Net Input and Output   Activation function: $S(x) = \frac{1}{1+e^{-x}}.$

Table 9.2 Net Input and Output Calculations

Unit, j	*Net Input, I_j*	*Output, O_j*
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1+e^{0.7}) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1+e^{-0.1}) = 0.525$
6	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1+e^{0.105}) = 0.474$

2. Calculation of the Error at Each Node

Unit, j	Err_j
6	$(0.474)(1-0.474)(1-0.474)=0.1311$
5	$(0.525)(1-0.525)(0.1311)(-0.2)=-0.0065$
4	$(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087$

3. Please Calculate for each Weight and Bias Updating

Weight or Bias	*New Value*
w_{46}	$-0.3+(0.9)(0.1311)(0.332)=-0.261$
w_{56}	$-0.2+(0.9)(0.1311)(0.525)=-0.138$
w_{14}	$0.2+(0.9)(-0.0087)(1)=0.192$
w_{15}	$-0.3+(0.9)(-0.0065)(1)=-0.306$
w_{24}	$0.4+(0.9)(-0.0087)(0)=0.4$
w_{25}	$0.1+(0.9)(-0.0065)(0)=0.1$
w_{34}	$-0.5+(0.9)(-0.0087)(1)=-0.508$
w_{35}	$0.2+(0.9)(-0.0065)(1)=0.194$
θ_6	$0.1+(0.9)(0.1311)=0.218$
θ_5	$0.2+(0.9)(-0.0065)=0.194$
θ_4	$-0.4+(0.9)(-0.0087)=-0.408$

(参考公式)

The variable l is the learning rate

Backpropagate the error: The error is propagated backward by updating the weights and biases to reflect the error of the network's prediction. For a unit j in the output layer, the error Err_j is computed by

$$Err_j = O_j(1 - O_j)(T_j - O_j), \tag{9.6}$$

where O_j is the actual output of unit j, and T_j is the known target value of the given training tuple. Note that $O_j(1 - O_j)$ is the derivative of the logistic function.

To compute the error of a hidden layer unit j, the weighted sum of the errors of the units connected to unit j in the next layer are considered. The error of a hidden layer unit j is

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}, \tag{9.7}$$

where w_{jk} is the weight of the connection from unit j to a unit k in the next higher layer, and Err_k is the error of unit k.

The weights and biases are updated to reflect the propagated errors. Weights are updated by the following equations, where Δw_{ij} is the change in weight w_{ij}:

$$\Delta w_{ij} = (l) Err_j O_i. \tag{9.8}$$

$$w_{ij} = w_{ij} + \Delta w_{ij}. \tag{9.9}$$

第一題

```
import math
dct={"n1":{"x1":1,"w14":0.2,"w15":-0.3},
   "n2":{"x2":0,"w24":0.4,"w25":0.1},
   "n3":{"x3":1,"w34":-0.5,"w35":0.2},
   "n4":{"b4":-0.4,"w46":-0.3},
   "n5":{"b5":0.2,"w56":-0.2},
   "n6":{"b6":0.1}}

input_n4 = [
 dct["n1"]["x1"]*dct["n1"]["w14"],
 dct["n2"]["x2"]*dct["n2"]["w24"],
 dct["n3"]["x3"]*dct["n3"]["w34"],
  dct["n4"]["b4"]
]

input_n5 = [
 dct["n1"]["x1"]*dct["n1"]["w15"],
 dct["n2"]["x2"]*dct["n2"]["w25"],
 dct["n3"]["x3"]*dct["n3"]["w35"],
  dct["n5"]["b5"]
]


def get_input(arr):
  agg=0
  for i in range(len(arr)):
    agg+=arr[i]
  return agg

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

x4=get_input(input_n4)
x5=get_input(input_n5)
print("output-4:",get_output(x4))
print("output-5:",get_output(x5))


input_n6 = [
 dct["n4"]["w46"]*get_output(x4),
 dct["n5"]["w56"]*get_output(x5),
  dct["n6"]["b6"]
]

x6=get_input(input_n6)
print("output-6:",get_output(x6))
```

第二題

```
arr_output=[get_output(x4),get_output(x5),get_output(x6)]
arr_err4=[dct["n4"]["w46"]*get_output(x4)*(1-get_output(x4))]
arr_err5=[dct["n5"]["w56"]*get_output(x5)*(1-get_output(x5))]

def get_error6(arr_output):
  err6=(get_output(x6))*(1-(get_output(x6)))*(1-(get_output(x6)))
  return err6

print('Err6:',get_error6(arr_output))

def get_error(arr):
  err=0
  for i in range(len(arr)):
    err=get_error6(arr_output)*arr[i]
    return err


print('Err5:',get_error(arr_err5))
print('Err4:',get_error(arr_err4))
```

第三題

```
LR=0.9
a=[0,dct['n1']['x1'],dct['n2']['x2'],dct['n3']['x3'],get_output(x4)
,get_output(x5),get_output(x6)]
Err6=get_error6(arr_output)
Err5=get_error(arr_err5)
Err4=get_error(arr_err4)

print("w14:",dct['n1']['w14']+Err4*a[1]*LR)
print("w15:",dct['n1']['w15']+Err5*a[1]*LR)
print("w24:",dct['n2']['w24']+Err4*a[2]*LR)
print("w25:",dct['n2']['w25']+Err5*a[2]*LR)
print("w34:",dct['n3']['w34']+Err4*a[3]*LR)
print("w35:",dct['n3']['w35']+Err5*a[3]*LR)
print("w46:",dct['n4']['w46']+Err6*a[4]*LR)
print("w56:",dct['n5']['w56']+Err6*a[5]*LR)
print("Θ4:",dct['n4']['b4']+Err4*LR)
print("Θ5:",dct['n5']['b5']+Err5*LR)
print("Θ6:",dct['n6']['b6']+Err6*LR)
```