Homework 11/14 (4-4 Backpropagation)

This is an homework for a neural network course offered by the master's class of the Department of IEM at the NCUT in the first semester of the 2024 academic year (113-1).

Submitted by: 4B315021 詹家緯

The codes can be viewed on GitHub: https://github.com/chankai1016/113-1_Neural_Network/

Problem 1

證明$\boldsymbol{f'(x) = f(x)[1 - f(x)]}$

將方程整理為：

$$
\begin{aligned}
& f'(x) = f(x)[1 - f(x)] \\
\Leftrightarrow & \frac{df}{dx} = f(x)[1 - f(x)] \\
\Leftrightarrow & \frac{1}{f(x)[1 - f(x)]} \times df = dx \\
\Leftrightarrow & \left(\frac{1}{f(x)} + \frac{1}{1 - f(x)}\right) df = dx \ (\because \text{部分分式分解})
\end{aligned}
$$

對兩邊積分：

$$
\begin{aligned}
& \int \frac{1}{f(x)} df + \int \frac{1}{1 - f(x)} df = \int dx \\
\Leftrightarrow & ln|f(x)| - ln|1 - f(x)| = x + C \\
\Leftrightarrow & ln\left|\frac{f(x)}{1 - f(x)}\right| = x + C \\
\Leftrightarrow & \frac{f(x)}{1 - f(x)} = Ce^x
\end{aligned}
$$

解出$f(x)$：

$$
\begin{aligned}
& \frac{f(x)}{1 - f(x)} = Ce^x \\
\Leftrightarrow & f(x) = Ce^x(1 - f(x)) \\
\Leftrightarrow & f(x) = Ce^x - Ce^x f(x) \\
\Leftrightarrow & f(x) + Ce^x f(x) = Ce^x \\
\Leftrightarrow & (1 + Ce^x) f(x) = Ce^x \\
\Leftrightarrow & f(x) = \frac{Ce^x}{1 + Ce^x}
\end{aligned}
$$

驗證解：

對$f(x) = Ce^x/(1 + Ce^x)$求導

$$
f'(x) = \frac{Ce^x \times (1 + Ce^x) - Ce^x \times Ce^x}{(1 + Ce^x)^2} = \frac{Ce^x}{(1 + Ce^x)^2}
$$

計算$f(x)[1 - f(x)]$

$$
f(x)[1 - f(x)] = \frac{Ce^x}{1 + Ce^x} \times \left(1 - \frac{Ce^x}{1 + Ce^x}\right) = \frac{Ce^x}{1 + Ce^x} \times \frac{1}{1 + Ce^x} = \frac{Ce^x}{(1 + Ce^x)^2}
$$

$$
\therefore f'(x) = f(x)[1 - f(x)]
$$

■

Problem 2

Numerical example of the backpropagation procedure

Write a program to perform the following computational problem of the backpropagation algorithm below:

> The initial biases and weights generated by computer·for network 2-4-4 (Fig 1) are listed in Tables 1 and 2. When the first pattern, (x_1, t_1), is presented, where x_1 = (0.017322, 1.480488)^t and t_1 = (0.494200, 0.495051, 0.494171, 0.501720)^t.

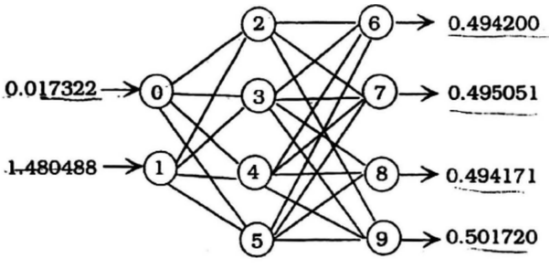

Figure 1 The architecture of network 2-4-4

Table 1 The initial random biases of network 2-4-4

i	b_i	i	b_i
2	-0.444700	6	0.094012
3	0.410733	7	-0.058550
4	0.358089	8	-0.055376
5	-0.005783	9	-0.158925

Table 2 The initial random weights of network 2-4-4

$j-i$	w_{ij}	$j-i$	w_{ij}
0—2	0.121845	2—7	0.326563
1—2	-0.384945	3—7	0.360866
0—3	0.474700	4—7	0.046312
1—3	0.131458	5—7	0.323694
0—4	0.194113	2—8	-0.106006
1—4	0.187948	3—8	0.264275
0—5	-0.318567	4—8	-0.455687
1—5	0.117237	5—8	0.128620
2—6	0.069170	2—9	-0.189261
3—6	-0.088916	3—9	-0.165883
4—6	-0.432951	4—9	-0.068896
5—6	-0.270775	5—9	-0.490692

Code: main.py

```
import numpy as np

def backpropagation(input_v, target, bias, weight):
    print("\n#### 1. Forward Pass")
    print("\n##### A. Hidden Unit")
    print("\n| i |  net(i)  |   a(i)   |")
    print("| :--: | ---: | ---: |")
    net = [0] * 10
    a = [0] * 10
    for i in range(len(input_v)):
        a[i] = input_v[i]
    for i in range(2, 5 + 1):  # 2, 3, 4, 5 : hidden num
        for j in range(0, 1 + 1):  # 0, 1 : input num
            # net_i = a_0*a_0i + a_1*a_1i + b_i
            net[i] += input_v[j] * weight[j][i]
        net[i] += bias[i]
        a[i] = (1 + np.exp(-1 * net[i])) ** -1
        print("| **{}** | {:.6f} | {:.6f} |".format(i, net[i], a[i]))
    print("\n##### B. Output Unit")
    print("\n| i |  net(i)  |   a(i)   |")
    print("| :--: | ---: | ---: |")
    for i in range(6, 9 + 1):  # 6, 7, 8, 9 : output num
        for j in range(2, 5 + 1):  # 2, 3, 4, 5 : hidden num
            # net_i = a_2*a_2i + a_3*a_3i + a_4*a_4i + a_5*a_5i + b_i
            net[i] += a[j] * weight[j][i]
        net[i] += bias[i]
        a[i] = (1 + np.exp(-1 * net[i])) ** -1
        print("| **{}** | {:.6f} | {:.6f} |".format(i, net[i], a[i]))

    print("\n#### 2. Backward Pass")
```

```
    delta = [0] * 10
    print("\n##### A. Output Unit")
    print("\n| i |   δ(i)   |")
    print("| :--: | ---: |")
    for i in range(6, 9 + 1):  # 6, 7, 8, 9 : output num
        delta[i] = (target[i - 6] - a[i]) * ( a[i] * (1 - a[i] ))  # 1-6 = 0, 1, 2, 3 :
target num
        print("| **{}** | {:.6f} |".format(i, delta[i]))
    print("\n##### B. Hidden Unit")
    print("\n| i |   δ(i)   |")
    print("| :--: | ---: |")
    for i in range(2, 5 + 1):  # 2, 3, 4, 5 : hidden num
        for j in range(6, 9 + 1):  # 6, 7, 8, 9 : output num
            delta[i] += delta[j] * weight[i][j]
        delta[i] *= a[i] * (1 - a[i])
        print("| **{}** | {:.6f} |".format(i, delta[i]))

    print("\n#### 3. Change of Weights and Biases")
    print("\n##### A. Weights")
    print("\n| i - j |   ΔW(i)   |  W^new(i) | i - j |   ΔW(i)   |  W^new(i) | i - j |
ΔW(i)   |  W^new(i) | i - j |   ΔW(i)   |  W^new(i) |")
    print("| :--: | ---: | ---: | :--: | ---: | ---: | :--: | ---: | ---: | :--: | ---: | --
-: |")
    d_weight = [[0] * 10] * 6
    weight_new = [[0] * 10] * 6
    for i in range(0, 1+1): # 0, 1 : input num
        for j in range(2, 5+1): # 2, 3, 4, 5 : hidden num
            d_weight[i][j] = 0.20 * delta[j] * a[i]
            weight_new[i][j] = weight[i][j] + d_weight[i][j]
            print("| **{} - {}** | {:.6f} | {:.6f} ".format(i, j, d_weight[i][j],
weight_new[i][j]), end = "")
        print("|")
    for i in range(2, 5+1): # 2, 3, 4, 5 : hidden num
        for j in range(6, 9+1): # 6, 7, 8, 9 : output num
            d_weight[i][j] = 0.20 * delta[j] * a[i]
            weight_new[i][j] = weight[i][j] + d_weight[i][j]
            print("| **{} - {}** | {:.6f} | {:.6f} ".format(i, j, d_weight[i][j],
weight_new[i][j]), end = "")
        print("|")
    print("\n##### B. Biases")
    print("\n| i |   Δb(i)   |  b^new(i) | i |   Δb(i)   |  b^new(i) |")
    print("| :--: | ---: | ---: | :--: | ---: | ---: |")
    d_bias = [0] * 10
    bias_new = [0] * 10
    for i in range(2, 9+1): # 2 , ..., 9 : biases num
        d_bias[i] = 0.20 * delta[i]
        bias_new[i] = bias[i] + d_bias[i]
    # for output table
    for i in range(2, 5+1):
        print("| **{}** | {:.6f} | {:.6f} | **{}** | {:.6f} | {:.6f} |".format(i, d_bias[i],
bias_new[i], i+4, d_bias[i+4], bias_new[i+4]))
    return 0
```

Input

```
input_v = (0.017322, 1.480488)
target = (0.494200, 0.495051, 0.494171, 0.501720)
# b_2 = bias[2] etc...
```

```
bias = (0, 0, -0.444700, 0.410733, 0.358089, -0.005783, 0.094012, -0.058550, -0.055376,
-0.158925)
# w_0-2 = weight[0][2] etc...
weight = [
    [0, 0, 0.121845, 0.474700, 0.194113, 0.318567, 0, 0, 0, 0],
    [0, 0, -0.384945, 0.131458, 0.187948, 0.117237, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0.069170, 0.326563, -0.106006, -0.189261],
    [0, 0, 0, 0, 0, 0, -0.088916, 0.360866, 0.264275, -0.165883],
    [0, 0, 0, 0, 0, 0, -0.432951, 0.046312, -0.455687, -0.068896],
    [0, 0, 0, 0, 0, 0, -0.270775, 0.323694, 0.128620, -0.490692],
]
backpropagation(input_v, target, bias, weight)
```

Output

以markdown形式輸出

1. Forward Pass

A. Hidden Unit

i	net(i)	a(i)
2	-1.012496	0.266492
3	0.613578	0.648757
4	0.639706	0.654687
5	0.173303	0.543218

B. Output Unit

i	net(i)	a(i)
6	-0.375777	0.407146
7	0.468747	0.615087
8	-0.140639	0.464898
9	-0.628637	0.347820

2. Backward Pass

A. Output Unit

i	δ(i)
6	0.021013
7	-0.028419
8	0.007282
9	0.034911

B. Hidden Unit

i	δ(i)
2	-0.002972
3	-0.003644
4	-0.003648
5	-0.007713

3. Change of Weights and Biases

A. Weights

i - j	ΔW(i)	W^new(i)	i - j	ΔW(i)	W^new(i)	i - j	ΔW(i)	W^new(i)	i - j	ΔW(i)	W^new(i)
0 - 2	-0.000010	0.121835	**0 - 3**	-0.000013	0.474687	**0 - 4**	-0.000013	0.194100	**0 - 5**	-0.000027	0.318540
1 - 2	-0.000880	-0.385825	**1 - 3**	-0.001079	0.130379	**1 - 4**	-0.001080	0.186868	**1 - 5**	-0.002284	0.114953
2 - 6	0.001120	0.070290	**2 - 7**	-0.001515	0.325048	**2 - 8**	0.000388	-0.105618	**2 - 9**	0.001861	-0.187400
3 - 6	0.002726	-0.086190	**3 - 7**	-0.003687	0.357179	**3 - 8**	0.000945	0.265220	**3 - 9**	0.004530	-0.161353
4 - 6	0.002751	-0.430200	**4 - 7**	-0.003721	0.042591	**4 - 8**	0.000954	-0.454733	**4 - 9**	0.004571	-0.064325
5 - 6	0.002283	-0.268492	**5 - 7**	-0.003088	0.320606	**5 - 8**	0.000791	0.129411	**5 - 9**	0.003793	-0.486899

B. Biases

i	Δb(i)	b^new(i)	i	Δb(i)	b^new(i)
2	-0.000594	-0.445294	**6**	0.004203	0.098215
3	-0.000729	0.410004	**7**	-0.005684	-0.064234
4	-0.000730	0.357359	**8**	0.001456	-0.053920
5	-0.001543	-0.007326	**9**	0.006982	-0.151943