National Taiwan University **Introduction to Machine Learning and Deep Learning**

Department of Civil Engineering Instructor: C.-S. CHEN

Homework 4 Report

Essay and Programming, Due 21:00, Wednesday, November 5, 2025

Student ID: B12501001
Name: 蔣尚誠

---

1. Print the weights for each epoch. The output format should be displayed as shown in the `HW4_1.ipynb`**.**

```
epoch = 1, weights = -0.00 0.40 0.40 0.40
epoch = 2, weights = -0.40 0.20 0.40 0.20
epoch = 3, weights = -0.40 0.40 0.40 0.20
epoch = 4, weights = -0.40 0.40 0.40 0.20
epoch = 5, weights = -0.40 0.40 0.40 0.20
epoch = 6, weights = -0.40 0.40 0.40 0.20
epoch = 7, weights = -0.40 0.40 0.40 0.20
epoch = 8, weights = -0.40 0.40 0.40 0.20
epoch = 9, weights = -0.40 0.40 0.40 0.20
epoch = 10, weights = -0.40 0.40 0.40 0.20
epoch = 11, weights = -0.40 0.40 0.40 0.20
epoch = 12, weights = -0.40 0.40 0.40 0.20
epoch = 13, weights = -0.40 0.40 0.40 0.20
epoch = 14, weights = -0.40 0.40 0.40 0.20
epoch = 15, weights = -0.40 0.40 0.40 0.20
epoch = 16, weights = -0.40 0.40 0.40 0.20
epoch = 17, weights = -0.40 0.40 0.40 0.20
epoch = 18, weights = -0.40 0.40 0.40 0.20
epoch = 19, weights = -0.40 0.40 0.40 0.20
epoch = 20, weights = -0.40 0.40 0.40 0.20
```

2. Compare the convergence behaviors (number of epochs to achieve zero error) with three cases. Discuss how learning rate and initialization influence stability and speed.

For epoch = 20, **learning rate = 0.01, initial weight = 0.60**, the output should be displayed as following：

epoch = 1, weights = 0.52 0.58 0.58 0.58
epoch = 2, weights = 0.44 0.56 0.56 0.56
epoch = 3, weights = 0.36 0.54 0.54 0.54
epoch = 4, weights = 0.28 0.52 0.52 0.52
epoch = 5, weights = 0.20 0.50 0.50 0.50
epoch = 6, weights = 0.12 0.48 0.48 0.48
epoch = 7, weights = 0.04 0.46 0.46 0.46
epoch = 8, weights = -0.02 0.44 0.44 0.44
epoch = 9, weights = -0.08 0.42 0.42 0.42
epoch = 10, weights = -0.14 0.40 0.40 0.40
epoch = 11, weights = -0.20 0.38 0.38 0.38
epoch = 12, weights = -0.26 0.36 0.36 0.36
epoch = 13, weights = -0.32 0.34 0.34 0.34
epoch = 14, weights = -0.34 0.32 0.34 0.34
epoch = 15, weights = -0.34 0.32 0.34 0.34
epoch = 16, weights = -0.34 0.32 0.34 0.34
epoch = 17, weights = -0.34 0.32 0.34 0.34
epoch = 18, weights = -0.34 0.32 0.34 0.34
epoch = 19, weights = -0.34 0.32 0.34 0.34
epoch = 20, weights = -0.34 0.32 0.34 0.34

===== **epoch = 14** 時收斂

For epoch = 20, **learning rate = 0.10, initial weight = 0.25**, the output should be displayed as following

epoch = 1, weights = -0.15 0.05 0.25 0.25
epoch = 2, weights = -0.15 0.05 0.45 0.25
epoch = 3, weights = -0.35 0.05 0.45 0.25
epoch = 4, weights = -0.35 0.25 0.45 0.45
epoch = 5, weights = -0.55 0.25 0.45 0.25
epoch = 6, weights = -0.55 0.45 0.45 0.25
epoch = 7, weights = -0.55 0.45 0.45 0.25
epoch = 8, weights = -0.55 0.45 0.45 0.25

epoch = 9, weights = -0.55 0.45 0.45 0.25
epoch = 10, weights = -0.55 0.45 0.45 0.25
epoch = 11, weights = -0.55 0.45 0.45 0.25
epoch = 12, weights = -0.55 0.45 0.45 0.25
epoch = 13, weights = -0.55 0.45 0.45 0.25
epoch = 14, weights = -0.55 0.45 0.45 0.25
epoch = 15, weights = -0.55 0.45 0.45 0.25
epoch = 16, weights = -0.55 0.45 0.45 0.25
epoch = 17, weights = -0.55 0.45 0.45 0.25
epoch = 18, weights = -0.55 0.45 0.45 0.25
epoch = 19, weights = -0.55 0.45 0.45 0.25
epoch = 20, weights = -0.55 0.45 0.45 0.25

===== **epoch = 6 時收斂**

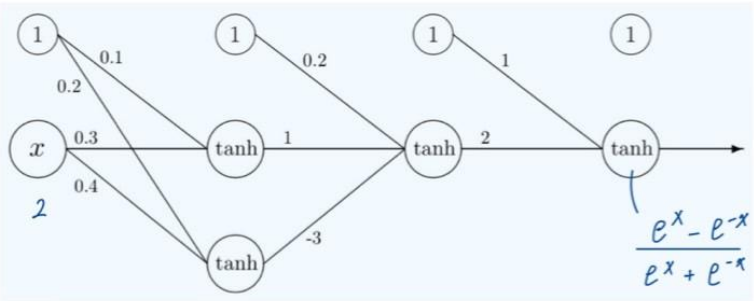
第一題為 epoch = 3 時收斂 (learning rate = 0.10, initial weight = 0.60)，可得知當 learning rate (影響收斂速度) 越大，代表每次改變權重的幅度越多，越快到達權重的收斂位置；當 Initial weights (影響早期錯誤數) 越接近最後的收斂位置時，所需的修正就較少，越快到達權重的收斂位置。

即使資料是可以被正確分類的，只要一開始設定的 learning rate 或 initial weight 不一樣，神經網路在訓練時走的路徑就會不同。當模型第一次成功把資料全部分對時，它就會停下來，不再繼續往更好的解微調，因此每次訓練最後收斂的位置都可能不一樣。

3.

(a) Using the square root error as our error function, derive and compute $\delta^{(3)}$, $\delta^{(2)}$, $\delta^{(1)}$.

$\delta^{(3)}$	$\delta^{(2)}$	$\delta^{(1)}$
[[-0.56]]	**[[-0.21]]**	**[[-0.13] [0.26]]**



Forward:

$$\begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix}^T \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.7 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0.2 \\ 1 \\ -3 \end{bmatrix}^T \begin{bmatrix} 1 \\ 0.6 \\ 0.76 \end{bmatrix} = [-1.5]$$

$$\Rightarrow \begin{bmatrix} 1 \\ 2 \end{bmatrix}^T \begin{bmatrix} 1 \\ 0.9 \end{bmatrix} = [-0.8] \Rightarrow -0.66$$

Backprop:

square root error: $s^L = \theta = x^L$, $e = \sqrt{(x^L - y)^2}$

$$\delta^L = \frac{\partial e}{\partial s^L} = \boxed{\frac{\partial e}{\partial x^L}} \cdot \boxed{\frac{\partial x^L}{\partial s^L}} \longrightarrow = \theta'(s^L) = 1 - (x^L)^2, \quad \theta(s) = \tanh(s)$$

$$\rightarrow \frac{1}{2}\left[(x^L - y)^2\right]^{-\frac{1}{2}} \cdot 2(x^L - y) = \frac{x^L - y}{\sqrt{(x^L - y)^2}} = \frac{x^L - y}{|x^L - y|} = \text{sign}(x^L - y)$$

$$\Rightarrow \delta^L = \text{sign}(x^L - y) \cdot (1 - (x^L)^2)$$

$$\delta^3 = \text{sign}(-0.66 - 1) \cdot [1 - (-0.66)^2] = -0.56 \ *$$

$$\Rightarrow \delta^l = \theta'(s^l) \otimes \left[w^{(l+1)} \delta^{(l+1)} \right]_1^{d^{(l)}}$$

$$\delta^2 = (1 - (-0.9)^2) \cdot (2 \cdot (-0.56)) = -0.21 \ \#$$

$$\delta^1 = \begin{bmatrix} (1 - (0.6)^2) \cdot (1 \cdot (-0.21)) \\ (1 - (0.76)^2) \cdot (-3 \cdot (-0.21)) \end{bmatrix} = \begin{bmatrix} -0.13 \\ 0.26 \end{bmatrix} \#$$

(b) Compute $\frac{\partial e}{\partial W^{(1)}}$, $\frac{\partial e}{\partial W^{(2)}}$, $\frac{\partial e}{\partial W^{(3)}}$.

$\frac{\partial e}{\partial W^{(1)}}$	$\frac{\partial e}{\partial W^{(2)}}$	$\frac{\partial e}{\partial W^{(3)}}$
[[-0.13 0.26] [-0.26 0.52]]	[[-0.21] [-0.13] [-0.16]]	[[-0.56] [0.50]]

$$s^L = W^L \cdot x^{L-1}$$

$$\frac{\partial e}{\partial W^L} = \underbrace{\frac{\partial e}{\partial s^L}}_{\delta^L} \cdot \underbrace{\frac{\partial s^L}{\partial W^L}}_{x^{L-1}} \Rightarrow x^{L-1} \cdot (\delta^L)^T$$

$$\frac{\partial e}{\partial w} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} -0.13 \\ 0.26 \end{bmatrix}^T = \begin{bmatrix} -0.13 & 0.26 \\ -0.26 & 0.52 \end{bmatrix}$$

$$\frac{\partial e}{\partial w^2} = \begin{bmatrix} 1 \\ 0.6 \\ 0.76 \end{bmatrix} \begin{bmatrix} -0.21 \end{bmatrix}^T = \begin{bmatrix} -0.21 \\ -0.13 \\ -0.16 \end{bmatrix}$$

$$\frac{\partial e}{\partial w^3} = \begin{bmatrix} 1 \\ -0.9 \end{bmatrix} \begin{bmatrix} -0.56 \end{bmatrix}^T = \begin{bmatrix} -0.56 \\ 0.50 \end{bmatrix}$$

(c) Update the weight matrices using this single datapoint with a learning rate $\eta = 0.5$, repeat the forward propagation.

$W^{(3)}(t)$	$W^{(2)}(t)$	$W^{(1)}(t)$
[[1.28] [1.75]]	[[0.305] [1.065] [-2.92]]	[[0.165 0.07] [0.43 0.14]]

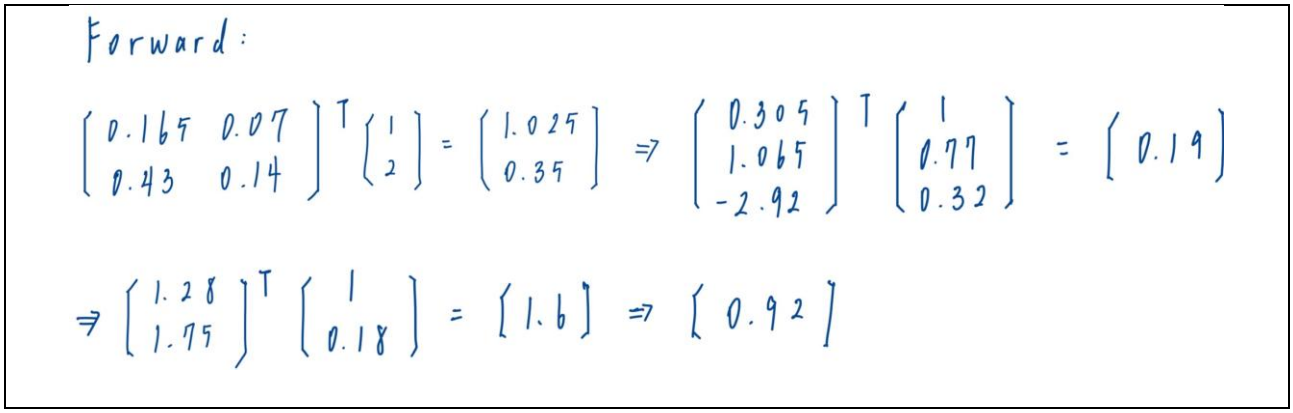
$$W^1(t) = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix} - 0.5 \begin{bmatrix} -0.13 & 0.26 \\ -0.26 & 0.52 \end{bmatrix} = \begin{bmatrix} 0.165 & 0.07 \\ 0.43 & 0.14 \end{bmatrix}$$

$$W^2(t) = \begin{bmatrix} 0.2 \\ 1 \\ -3 \end{bmatrix} - 0.5 \begin{bmatrix} -0.21 \\ -0.13 \\ -0.16 \end{bmatrix} = \begin{bmatrix} 0.305 \\ 1.065 \\ -2.92 \end{bmatrix}$$

$$W^3(t) = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - 0.5 \begin{bmatrix} -0.56 \\ 0.50 \end{bmatrix} = \begin{bmatrix} 1.28 \\ 1.75 \end{bmatrix}$$

Compute $s^{(1)}$, $x^{(1)}$, $s^{(2)}$, $x^{(2)}$, and $x^{(3)}$.

$s^{(1)}$	$x^{(1)}$	$s^{(2)}$	$x^{(2)}$	$x^{(3)}$
[[1.025] [0.35]]	[[1.00] [0.77] [0.32]]	[[0.19]]	[[1.00] [0.18]]	[[0.92]]



4.

(a) Derive and compute $s^{(1)}$, $x^{(1)}$, $s^{(2)}$, $x^{(2)}$, and $x^{(3)}$.

$s^{(1)}$	$x^{(1)}$	$s^{(2)}$	$x^{(2)}$	$x^{(3)}$
[[0.700000] [1.000000]]	[[1.000000] [0.604368] [0.761594]]	[[-1.480415] [2.367119]]	[[1.000000] [-0.901546] [0.914285]]	[[1.379970]]

```python
# ===== 基本函數 =====
def tanh(x): return np.tanh(x)
def tanh_p_from_a(a): return 1.0 - a**2
def sigmoid(x): return 1.0 / (1.0 + np.exp(-x))
def sigmoid_p_from_a(a): return a * (1.0 - a)

# GELU（近似）與導數
_k, _c = np.sqrt(2.0/np.pi), 0.044715
def gelu(z):
    g = _k * (z + _c * z**3)
    return 0.5 * z * (1.0 + np.tanh(g))
def gelu_p(z):
    g = _k * (z + _c * z**3)
    t = np.tanh(g)
    gp = _k * (1.0 + 3.0 * _c * z**2)
    return 0.5 * (1.0 + t) + 0.5 * z * (1.0 - t**2) * gp

# 小工具
def add_bias(v): return np.concatenate([[1.0], v.reshape(-1)])
def pvec(v): return np.array2string(v, precision=6, floatmode="fixed")
def pmat(M): return np.array2string(M, precision=6, floatmode="fixed")

# ===== 題目權重（依你給的版）=====
# L0->[tanh,tanh]，輸入 [1,x] @ W1  -> s^(1)  (shape: (2,))
W1 = np.array([[0.1, 0.2],
               [0.3, 0.4]], dtype=float)

# L1->[tanh, sigmoid]，輸入 [1, a1_0, a1_1] @ W2 -> s^(2) (shape: (2,))
#   第一列為 bias 權重；第一個神經元 tanh，第二個神經元 sigmoid
W2 = np.array([[ 0.2,  0.3],
               [ 1.0,  0.9],
               [-3.0,  2.0]], dtype=float)

# L2->輸出 (GELU)，輸入 [1, tanh, sigmoid] @ W3 -> s^(3) (scalar)
W3 = np.array([[1.0],
               [2.0],
               [2.5]], dtype=float)
```

```python
# ===== (a) Forward =====
x0a = add_bias(x0)          # [1, x]
s1  = x0a @ W1              # pre-activation of layer-1 (2,)
a1  = tanh(s1)              # activations of layer-1 (2,)
x1a = add_bias(a1)          # [1, a1_0, a1_1]

s2  = x1a @ W2              # (2,)
a2  = np.array([tanh(s2[0]), sigmoid(s2[1])])  # [tanh, sigmoid]
x2a = add_bias(a2)          # [1, tanh, sigm]

s3  = float(x2a @ W3)       # scalar
a3  = np.array([gelu(s3)])  # scalar in array
E   = 0.5 * (a3 - y)**2
```

(b) Using the sum square as our error function, derive and compute $\delta^{(3)}$, $\delta^{(2)}$, $\delta^{(1)}$.

$\delta^{(3)}$	$\delta^{(2)}$	$\delta^{(1)}$
[[0.857356]]	[[0.321021] [0.167972]]	[[0.299721] [-0.263373]]

```
# ===== (b) Backprop (δ) =====
# 輸出層
dE_da3 = 2 * (a3 - y)                # d(1/2*(a3 - y)^2)/da3
delta3  = float(dE_da3 * gelu_p(s3))  # scalar

# 第2層 ([tanh, sigmoid])
W3_nb  = W3[1:, :].flatten()                # 去掉 bias 的兩個權重
phi2p  = np.array([tanh_p_from_a(a2[0]), sigmoid_p_from_a(a2[1])])
delta2 = phi2p * (W3_nb * delta3)           # 逐元

# 第1層 (兩個 tanh)
W2_nb  = W2[1:, :]                          # 去掉 bias 列
phi1p  = tanh_p_from_a(a1)                  # 對 s^(1) 的導數，用 a1 計
delta1 = phi1p * (W2_nb @ delta2)

print("== Backprop deltas ==")
print(f"delta^(3) = {pvec(np.array([delta3]))}")
print(f"delta^(2) = {pvec(delta2)}    # [for tanh, for sigmoid]")
print(f"delta^(1) = {pvec(delta1)}\n")
```

(c) Compute $\frac{\partial e}{\partial W^{(1)}}$, $\frac{\partial e}{\partial W^{(2)}}$, $\frac{\partial e}{\partial W^{(3)}}$.

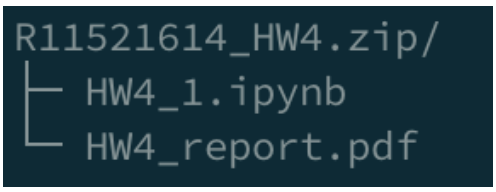
$\frac{\partial e}{\partial W^{(1)}}$	$\frac{\partial e}{\partial W^{(2)}}$	$\frac{\partial e}{\partial W^{(3)}}$
[[0.299721 -0.263373] [0.599443 -0.526746]]	[[0.321021 0.167972] [0.194015 0.101517] [0.244487 0.127927]]	[[0.857356] [-0.772946] [0.783868]]

```
# ===== (c) Gradients (外積) =====
gW3 = x2a.reshape(-1,1) * delta3
gW2 = np.outer(x1a, delta2)
gW1 = np.outer(x0a, delta1)

print("== Gradients ==")
print(f"dE/dW^(3) =\n{pmat(gW3)}\n")
print(f"dE/dW^(2) =\n{pmat(gW2)}\n")
print(f"dE/dW^(1) =\n{pmat(gW1)}\n")
```

Submission Format

Convert `HW4_report_template.docx` to `HW4_report.pdf`, then place `HW4_report.pdf` and `HW4_1.ipynb` into a folder named {yourStudentID}_HW4 and compress it into a ZIP file for upload to NTU COOL. Below are the file formats for upload.


R11521614_HW4.zip/
HW4_1.ipynb
HW4_report.pdf