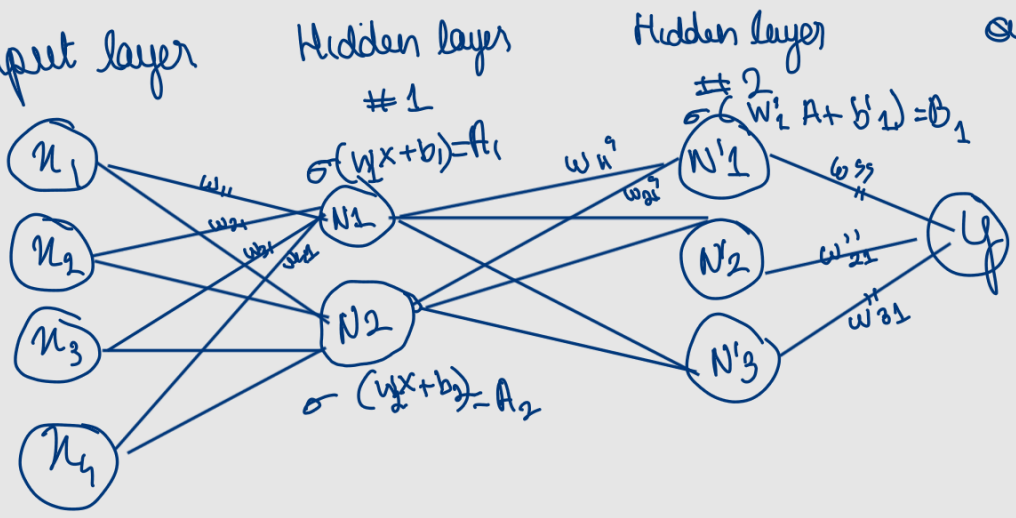
Approach A adds a false sense of distance between the classes of digits. For eg. If the desired output is say 2 , it will think the output 3 is closer to 2 than the output 5 because of the definition of error function. And it will train the weights accordingly. while in reality output 5 is just as wrong as output 3. Hence learning is misaligned with the ultimate goal.
A linear activation frame simply returns a weighted sum of inputs without any non linear transformation. The given problem is that of classification, we require a probabilistic output.
Approach B uses one hot encoding which is used to represent categories as vectors. Here we use a vector of length 10 to represent each of the 10 possible digit class(0-9):
For eg if digit=2 y=[0,0,1,0,0,0,0,0,0,0]. This clearly indicates which class is correct without creating a false sense of distance between classes.
Softmax converts raw network outputs (called logits) into probabilities.
Example output $[0.01, 0.03, 0.92, 0.01 \ldots]$

For outputs $z_0, z_1 \ldots z_9$:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=0}^{9} e^{z_j}}$$

this approach treats all classes equally, produces probability output for each digit and final uses that to predict the most probable digit.

2) a) Network architecture diagram



b) Let input vector $X = [x_1, x_2, x_3, x_4]$

weights are represented in matrix format

Hidden layer #1

	N_1	N_2
x_1	w_{11}	w_{12}
x_2	w_{21}	w_{22}
x_3	w_{31}	w_{32}
x_4	w_{41}	w_{42}

Let this be column vector X

this be row vector W_1

this be W_2

$\therefore$ output of neuron 1 $= \sigma(z_1) = A_1$ (at N_1)

where $z_1 = x_1 w_{11} + x_2 w_{21} + x_3 w_{31} + x_4 w_{41} + b_1$

$= W_1 \times X + b_1$

output of neuron 2 $= \sigma(z_2) = A_2$ (at N_1)

$z_2 = x_1 w_{12} + x_2 w_{22} + x_3 w_{32} + x_4 w_{42} + b_2$

$= W_1 \times X + b_2$

and $\sigma(z) = \max(0, z) \rightarrow$ ReLU fn

output of Hidden layer 1 = input of H2

A = vector (a_1, a_2)

Hidden layer #2

	N'_1	N'_2	N'_3
A_1	w'_{11}	w'_{12}	w'_{13}
A_2	w'_{21}	w'_{22}	w'_{23}
A	W'_1	W'_2	W'_3

output of $N'_1 = \sigma(z_1) = B_1$

$z_1 = w'_{11} A_1 + w'_{21} A_2 + b'_1$

$= W'_1 \times A + b'_1$

output of $N'_2 = \sigma(z_2) = B_2$

$z_2 = w'_{12} A_1 + w'_{22} A_2 + b'_2$

$= W'_2 \times A + b'_2$

output of $N'_3 = \sigma(z_3) = B_3$

$z_3 = w'_{13} A_1 + w'_{23} A_2 + b'_3$

output vector = $[B_1, B_2, B_3]$

Output layer (single neuron)

$y = w''_{11} B_1 + w''_{21} B_2 + w''_{31} B_3 + b''_1$

c) Let $X = (-1, 0, 1, 2)$

Let $W_1 = (1, 0, -1, 2)$, $b_1 = 0$

$z_1 = W_1 X + b_1$

$= -1 + 0 + -1 + 4 + 0$

$= 2$

$A_1 = \sigma(z_1) = 2$

Let $W_2 = (0, 1, 1, -1)$ $b_2 = 1$

$z_2 = 0 + 0 + 1 + -2 + 1$

$= 0$

$A_2 = \sigma(z_2) = 0$

Let $W'_1 = (1, 1)$ $b'_1 = 1$

$z'_1 = -2 + 0 + 1$

$= -1$

$B_1 = \sigma(z'_1) = 0$

Let $W'_2 = (0, 1)$ $b'_2 = 0$

$z'_2 = 0 + 0 + 0$

$= 0$

$B_2 = \sigma(z'_2) = 0$

Let $W'_3 = (-1, 0)$ $b'_3 = 1$

$z'_3 = -2 + 0 + 1$

$= -1$

$B_3 = \sigma(z'_3) = 0$

Let $W'''_1 = (1, 2, 3)$ $b'''_1 = 0$

$y = 0 + 0 + 0 + 0 = 0$

d) parameters

Layer 1: $4 \times 2 + 2 = 10$

Layer 2: $2 \times 3 + 3 = 9$

Layer 3: $3 + 1 = 4$

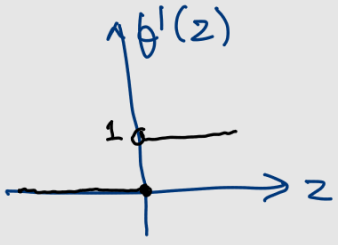
23

3) a) Derivative of $f(z)$ wrt z

$$f(z) = \max(0, z) \quad \therefore \quad = \begin{cases} 0 & ; \ z \leq 0 \\ z & ; \ z > 0 \end{cases}$$

$$f'(z) = \begin{cases} 0 & ; \ z < 0 \\ \text{undefined} & ; \ z = 0 \\ 1 & ; \ z > 0 \end{cases}$$

* most libraries define $f'(z) = 0$ at $z = 0$



b) given pre-activation sum $z < 0$ for all inputs

$\therefore f(z) = 0 \quad \therefore f' = 0$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial f} \cdot \frac{\partial f}{\partial w_i} = \frac{\partial L}{\partial f} \cdot \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial w_i} = 0$$

c) No, there is no change in Loss f^n due to weights or bias

bc $\frac{\partial L}{\partial w_i}$ & $\frac{\partial L}{\partial b} = 0$ bc of the ReLu function.

during training using gradient descent

$$w_f = w_i - (l \cdot r) \frac{\partial L}{\partial w_i} = w_i \qquad b_f = b_i - (l \cdot r) \frac{\partial L}{\partial b} = b_i$$

$\therefore$ there is no change in weights or bias during training.

$\therefore$ we can say that bc of ReLu f^n, any neuron that produces a negative z for all its inputs, doesn't affect the loss f^n and hence its weights and bias will never get trained i.e in a way neuron becomes dead.

4) a) $J_{total}(w) = J_{data}(w) + \frac{\lambda}{2} w^2$

↓ total loss

↓ (MSE, cross entropy loss)

$\lambda \geq 0$ regularisation strength

w = single weight parameter

$$w_{new} = w_{old} - \eta \left(\frac{\partial J_{total}(w)}{\partial w} \right)$$

$$\frac{\partial J_{tot}}{\partial w} = \frac{\partial J_{data}}{\partial w} + \frac{\lambda}{2} 2w$$

$$\therefore w_{new} = w_{old} - \eta \left(\frac{\partial J_{data}}{\partial w} + \lambda w_{old} \right)$$

b) $w_{new} \simeq w_{old} \underbrace{(1 - \eta\lambda)}_{\text{factor slightly} < 1} - \eta \frac{\partial J_{data}}{\partial w}$

$\because \eta > 0 \quad \lambda > 0$

c) L2 regularisation is called weight decay because at every gradient descent step the weights are multiplied by a shrinking factor, even before applying the data gradient.

Even if the data gradient is 0, $w_{new} = (1 - \eta\lambda) w_{old}$

the weight still decays towards 0.

This discourages large weights and reduces overfitting

$\nabla \left(\frac{\lambda}{2} w^2 \right) = \lambda w$ is points away from origin

since its $-\lambda w$ points towards origin

$\therefore$ the shrink step pulls the point radially inward towards origin by contracting its weight vector

and the data gradient moves in the direction that reduces data error.