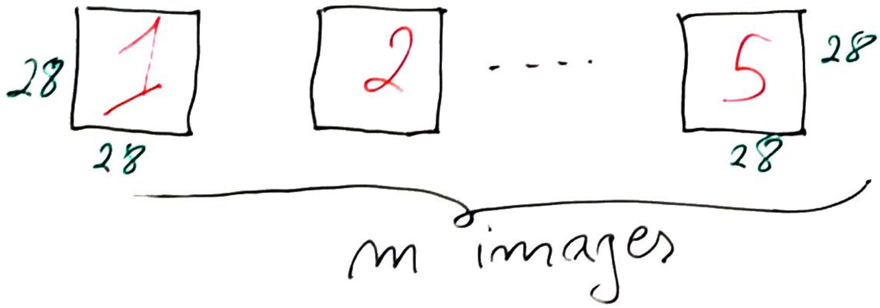
Multi-Layer perceptron from scratch:

Input X :



Output Y: 1 2 5

1- Reshaping the input:

$$X_{[m,28,28]} \Rightarrow X_{[m,784]} \Rightarrow X_{[N_x, m]}$$

N_x : # features
m : # samples

$$Y_{[m,]} \Rightarrow Y_{[1,m]}$$

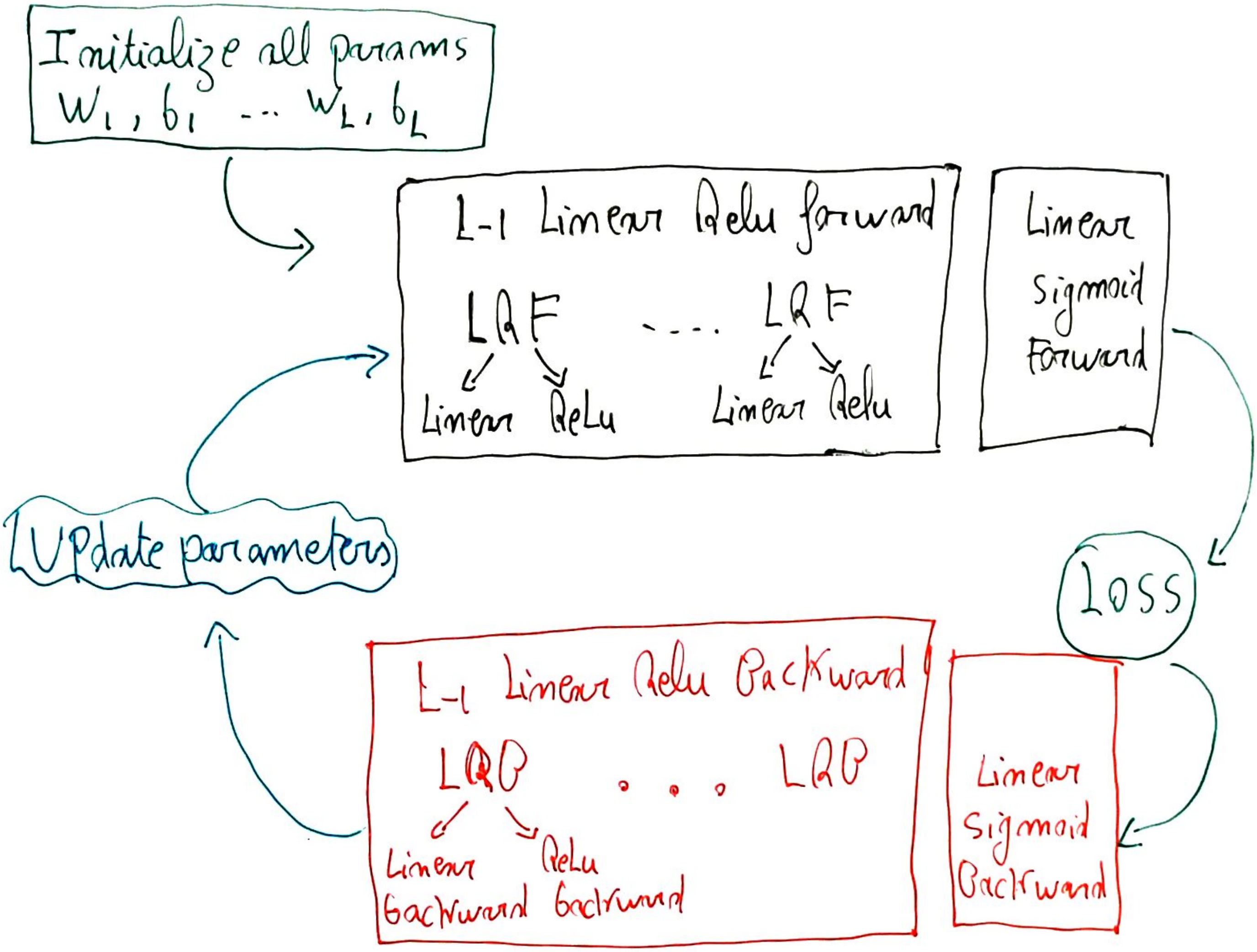
X:

	P_1	P_2	P_3	...	P_{784}
i_1	0	0	0	-	- - .
i_2	0	0	0	-	- -
⋮	0	255	0	-	- .
i_m	0	0	56	192 0	- ..

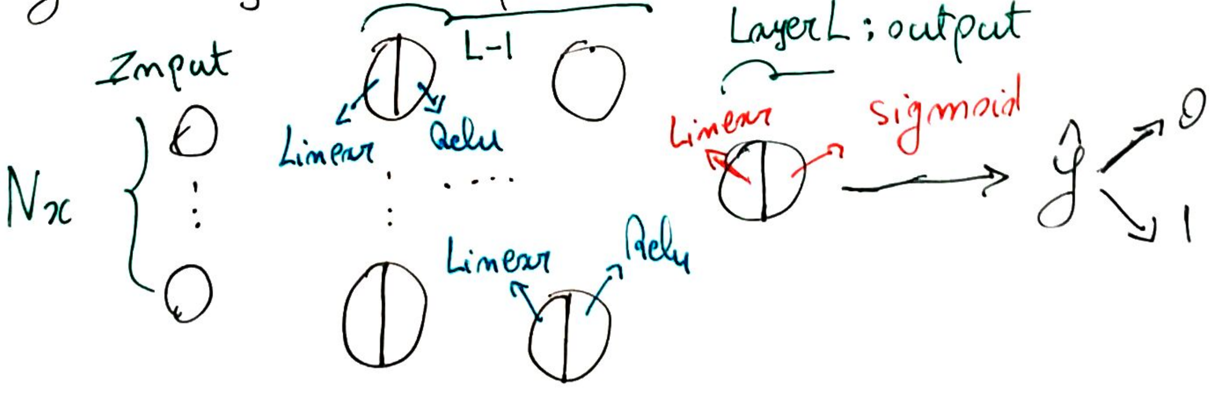
Y: $\underbrace{[1 \ 1 \ \cdots \cdots \ 5]}_{m}$

Objective : Building an L-layer neural network

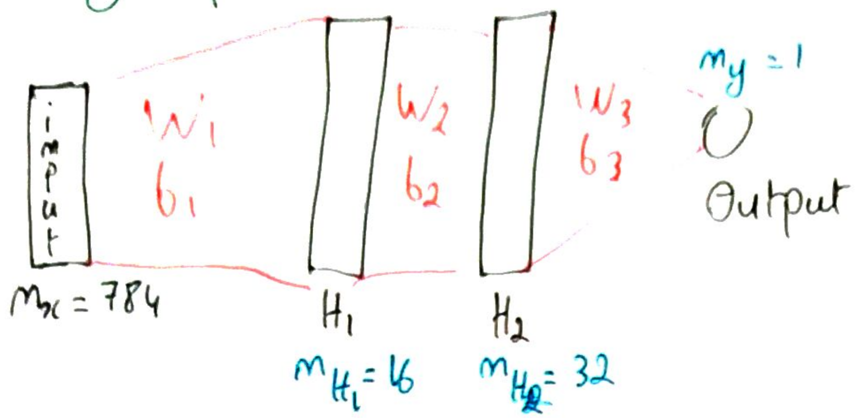
Strategy:
- initialize the parameters for L-layer NN
- Forward propagation
- Compute loss
- Backpropagation
- Update parameters.



Binary classification problem:



2- Initialize parameters



$W_1 \ (m_{H_1}, m_x)$

$b_1 \ (m_{H_1}, 1)$

$W_2 \ (m_{H_2}, m_{H_1})$

$b_2 \ (m_{H_2}, 1)$

$W_3 \ (m_y, m_{H_2})$

$b_3 \ (m_y, 1)$

3- Forward propagation:

Input $X \ (m_x, m)$

$\Downarrow \begin{matrix} W_1 \\ b_1 \end{matrix}$

Linear $Z_1 = W_1 X + b_1 \quad (m_{H_1}, m)$

Activation $A_1 = \sigma(Z_1) = \text{ReLu}(Z_1) = \max(0, Z_1)$

(m_{H_1}, m)

$\Downarrow \begin{matrix} W_2 \\ b_2 \end{matrix}$

Linear $Z_2 = W_2 A_1 + b_2 \quad (m_{H_2}, m)$

Activation $A_2 = \max(0, Z_2) \quad (m_{H_2}, m)$

$\Downarrow \begin{matrix} W_3 \\ b_3 \end{matrix}$

Linear $Z_3 = W_3 A_2 + b_3 \quad (\overset{1}{m_y}, m)$

Activation $A_3 = \sigma(Z_3) = \text{Sigmoid}(Z_3) = \frac{1}{1 + e^{-Z_3}}$

(m_y, m)

4- Compute cost

output A_L $(1, m)$

True labels Y $(1, m)$

binary classification

$\Downarrow$

binary crossentropy

x : images

label: number $= 3$, number $\neq 3$

what is the probability of an image being $= 3$ ideal $= 1$

Positive class : image $= 3$

Negative class : image $\neq 3$

Our model will predict a probability of an image being $= 3$. How good or bad are the predicted probabilities?

Loss function → Low values for good predictions

→ High values for bad predictions

$$J(W_1, b_1, \ldots, W_L, b_L) = -\frac{1}{m} \sum_{i=1}^{m} y_i \log(P(y_i)) + (1 - y_i) \log(1 - P(y_i))$$

$P(y_i)$ is the predicted probability of an image being $= 3$

Why it works?

Suppose: $y=1$

$P(y)=0.9$ ✓

$$y\log(P(y)) + \underbrace{(1-y)\log(1-P(y))}_{0}$$

$$1\log 0.9 = \boxed{-0.105}$$

$y=1$

$P(y)=0.2$ ✗

$$y\log(P(y)) + \underbrace{(1-y)\log(1-P(y))}_{0}$$

$$1\log 0.2 = \boxed{-1.609}$$

$y=0$

$P(y)=0.1$ ✓

$$\underbrace{y\log(P(y))}_{0} + (1-y)\log(1-P(y))$$

$$1\log(1-0.1)$$

$$1\log(0.9) = \boxed{-0.105}$$

$y=0$

$P(y)=0.8$ ✗

$$\underbrace{y\log(P(y))}_{0} + (1-y)\log(1-P(y))$$

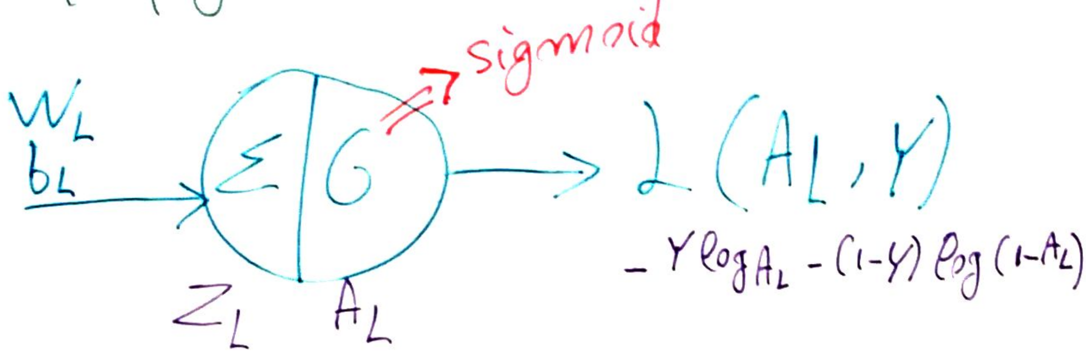
$$1\log(1-0.8)$$

$$1\log 0.2 = \boxed{-1.609}$$

---

we add the minus sign since the objective is to minimize error.

5- Backward propagation



chain rule: $$\frac{\partial L}{\partial W_L} = \frac{\partial L}{\partial A_L} \cdot \frac{\partial A_L}{\partial Z_L} \cdot \underbrace{\frac{\partial Z_L}{\partial W_L}}_{\hookrightarrow \frac{dZ_L A_{L-1}^T}{m}}$$

$$\frac{\partial L}{\partial A_L} = -\frac{Y}{A_L} + \frac{1-Y}{1-A_L}$$

$$\frac{\partial L}{\partial Z_L} = \frac{\partial L}{\partial A_L} \cdot \frac{d}{dZ_L} g(Z_L) = dA_L \cdot g'(Z_L)$$

Pseudo code:

$$dA[L] = -\left(\frac{Y}{A[L]}\right) + \frac{1-Y}{1-A[L]}$$

For each layer l:

input: $dA[l]$

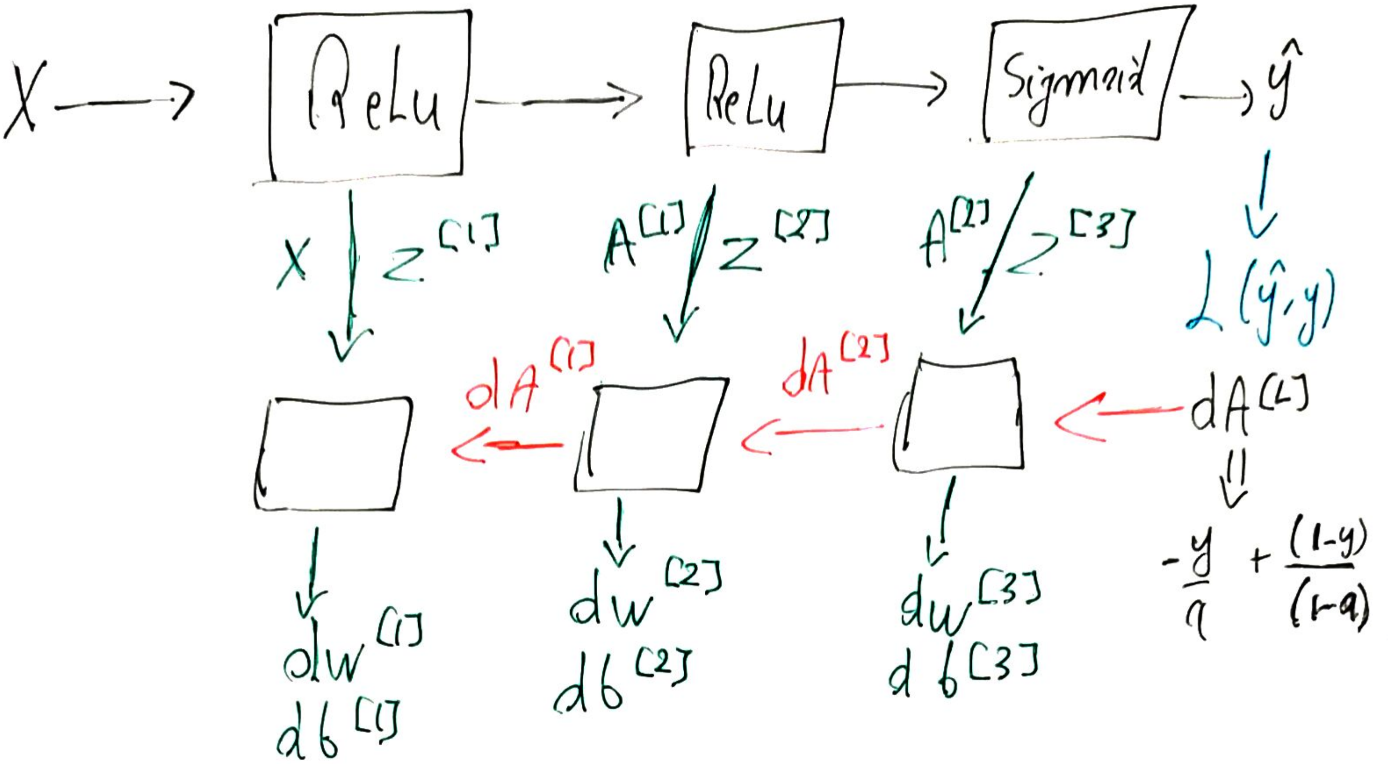
$dz[l] = dA[l]\, g'[l](Z[l])$

$dw[l] = \frac{1}{m}\, dz[l]\, A[l-1]^T$

$db[l] = \frac{1}{m} \sum dz[l]$

$dA[l-1] = w[l]^T\, dz[l]$

output $dA[l-1]$



6- Update parameters

for each layer l :

$$W_l = W_l - \alpha \, dW_l$$

$$b_l = b_l - \alpha \, db_l$$

α is the learning rate.

7 - predict