cross entropy loss.

y : 10 labels. $\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & & & \\ \vdots & & & \\ y_{10,1} & & & y_{10,n} \end{bmatrix}$ $10 \times n$ ← n examples

$\hat{y} = \begin{pmatrix} \hat{y}_{11} & \cdots & \hat{y}_{1n} \\ \vdots & & \vdots \\ \hat{y}_{10,1} & \cdots & \hat{y}_{10,n} \end{pmatrix}$ $10 \times n$

$loss = \begin{pmatrix} -\sum y_{r,1} \log(\hat{y}_{r,1}) & \cdots & -\sum y_{r,n} \log(\hat{y}_{r,n}) \end{pmatrix}$ $1 \times n$

$y_{r,1} \rightarrow r^{th}$ row, 1^{st} col.

$-\sum y_{r,1} \log(\hat{y}_{r,1}) \rightarrow -(y_{11} \log(\hat{y}_{11}) + y_{21} \log(\hat{y}_{21}) + \cdots)$

↓

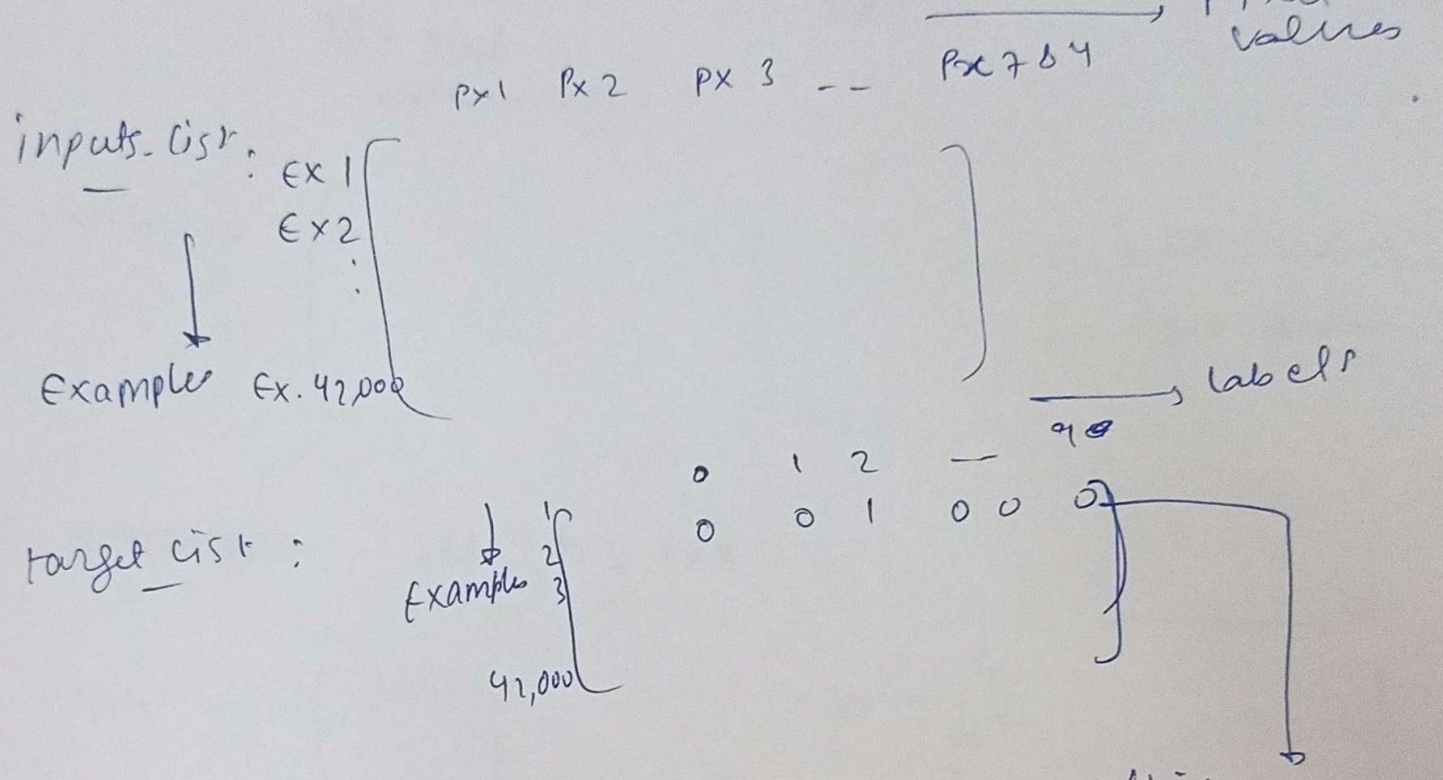
multi cross entropy loss.

if we sum up the "loss" matrix's all columns then we get the cost of model.

def. forward Pass (input_list)

* inputs_list shape : (42,000, 784)

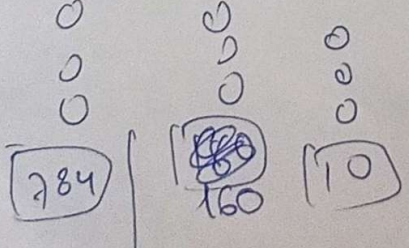
targets_list. shape : (42,000, 10)



Lets take hidden neuro = 160

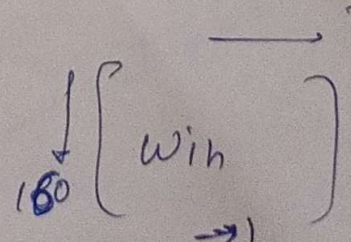
input neur = 784

output " = 10

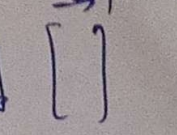


weight matrix

win = (160, 784)

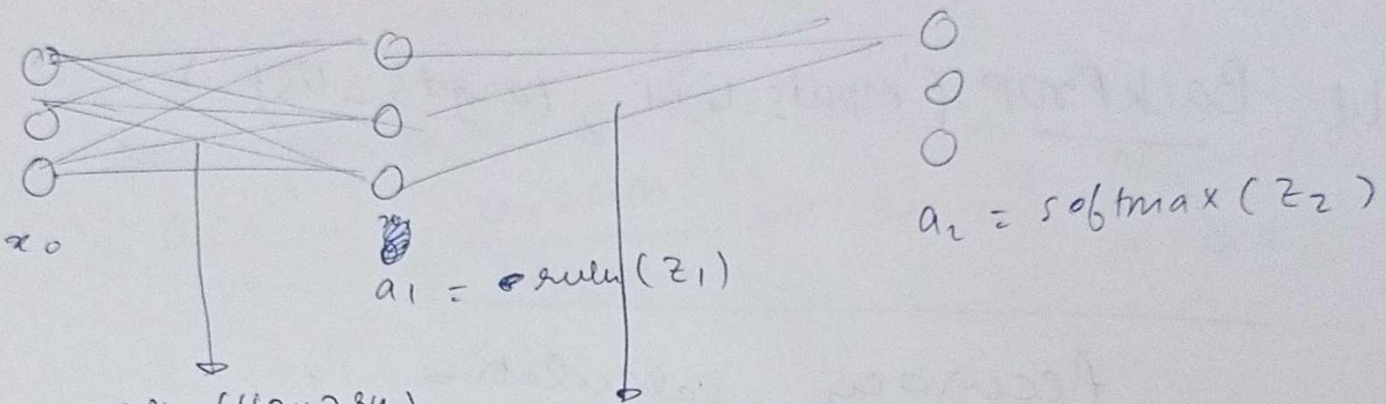


bin = (160, 1)



inputs = (Input . List)^T

inputs = (784, 42000) = x_0



w_{in} (160 x 784)
b_{in} (160 x 1)

w_{ho} (10 x 160)
b_{ho} (10 x 1)

$$z_1 = w_{ho} \cdot x_0 + b_{ho}$$

$$= \underbrace{\begin{bmatrix} 160 \times 784 \end{bmatrix}}_{w_{in}} \underbrace{\begin{bmatrix} 784 \times 42000 \end{bmatrix}}_{x_0} + \underbrace{\begin{bmatrix} \\ b_{in} \end{bmatrix}}_{160 \times 1}$$

$$= \begin{bmatrix} 160 \times 42000 \end{bmatrix} + \begin{bmatrix} 160 \times 1 \end{bmatrix}$$

↓ broadcast to (160, 42000)

$$= \begin{bmatrix} 160, 42000 \end{bmatrix}$$

a_1 = relu z_1 = 160, 42000

$z_2 = w_{ho} \cdot a_1 + b_{ho}$

$= (10 \times 160) \cdot (160 \cdot 42000) + (10 \times 1)$

$a_2 = (10, 42000)$

Backprop.

① input_list
② target_lis

① return loss
② and update params.

$L = self.CEL(y, \hat{y})$

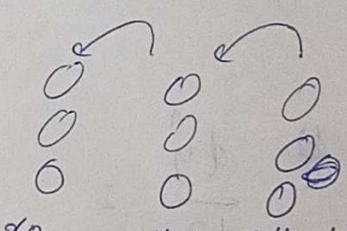
comes from forward pass.

$\hat{y} = forward(inputs\_list)$

$L = \checkmark$

now for param update,

$a_2 = \hat{y}$



$\frac{dL}{dw_{ho}} = \frac{dL}{da_2} \times \frac{da_2}{d\hat{y}} \times \frac{d\hat{y}}{dw_{ho}}$

$L = -\Sigma(y * \log(\hat{y}))$

$\hat{y} = softmax(z_2)$

$z_2 = w_{ho} \cdot a_1 + b_{ho}$

$a_1 = relu(z_1)$

$z_1 = w_{ih} \cdot x_0 + b_{ih}$

Terminologies

- x_0 = input
- $z_1 = z_h$
- $a_1 = z_h$
- $a_2 = z_0$
- $z_2 = z_0$
- $a_2 = \hat{y}$

$\frac{dL}{dw_{ho}} = \frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dz_2} \times \frac{dz_2}{dw_{ho}}$, $\frac{dL}{dw_{ih}} = \frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dz_2} \times \frac{dz_2}{da_1} \times \frac{da_1}{dz_1} \times \frac{dz_1}{dw_{ih}}$

↳ same for b_{ho}

↳ same for b_{ho}

$$\frac{dL}{d\hat{y}} = \text{cross\_entropy\_loss\_derivative}$$

$$= -\Sigma \frac{d(y_i \log \hat{y}_i)}{d\hat{y}_i} = -\Sigma y \frac{1}{\hat{y}}$$

$\rightarrow$ i is the i-th label

$$\frac{d\hat{y}}{dz_2} = \frac{d\,\text{softmax}(z_2)}{dz_2} = \frac{d}{dz_2}\left(\frac{e^{z_i}}{\sum e^{z_i}}\right)$$

on solving, $\frac{dL}{dz_2} = \frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dz_2} = \hat{y} - y$

$$\frac{dL}{dw_{ho}} = (\hat{y} - y)\frac{dz_2}{dw_{ho}} = (\hat{y} - y) \cdot a_1^T$$

$(10, 42000)$ $\rightarrow$ $(160, 42000)^T$

$= (42,000, 160)$

$= (10, 160)$

$$\frac{dL}{dw_{ih}} = \frac{dL}{dz_2} \times \frac{dz_2}{da_1} \times \frac{da_1}{dz_1} \times \frac{dz_1}{dw_{ih}} = (\hat{y} - y) \times d\,\text{relu}(z_2)$$

d

$$\frac{dL}{dw_{in}} = (\hat{y} - y) \times w_{ho} \times \frac{d(\text{relu } z_1)}{dz_1} \times x_0$$

$$= (\hat{y} - y) \times w_{ho} \times [\text{illegible}] \times x_0$$

$$rd(z_1)$$

$$\swarrow$$

$$\text{relu\_derivative}(z_1)$$

$$= (w_{ho}^T \cdot (\hat{y} - y)) \cdot x_0^T \times rd(z_1)$$

$$(10 \times 160)^T \quad (10, 42{,}000) \quad \rightarrow (784, 42{,}000)^T$$

$$= (160 \times 784)$$

same for

$\frac{dL}{db_{in}}$ and $\frac{dL}{db_{ho}}$

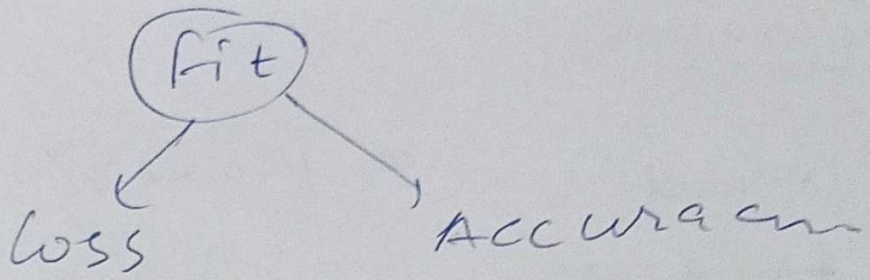
Fit

(X_train) inputs_list = (42,000, 784) } → train

(y_train) target_list = (42,000, 10)

X(_val) validation_labels = (12000, 784) } → val

(y_val) valid^n_data = (12,000, 10)



① loss.

tloss = self.backpro(inputs_list, targets_list)

Or

tloss = np.mean(CEL (tr_a , tr_p))

tr_a = targets_list . T

tr_p = forward (inputs_list)

vloss = np.mean (CEL (v_a, v_p))

v_a = val. label . T

v_p = forward (val_data)

② Accuracy

tr_pred = forward(X_train).T

~~tr_a =~~

---

tr_pred = np.argmax(tr_pred, axis=1)

tr_a = np.argmax(tr_a, axis=1)

same for val.

Momentum.

$\beta = 0.9$.

$w = w - \frac{dL}{dw} \times l.r$ → Before momentum.

$w = w - V_w \times l.r$ → after momentum.

$V_w = \beta \times w + (1-\beta) \times \frac{dL}{dw}$

& $V_b = \beta \times b + (1-\beta) \times \frac{dL}{db}$

if $\beta = 0$

$V_w = \frac{dL}{dw}$, $V_b = \frac{dL}{db}$.