# Building a Neural Network from Scratch.

Date:

1) Import the libraries and dataset.
   - → Numpy, Pandas, matPlotlib
   - → digit-recognizer train.csv.

About These dataset:-

- Train and Test csv → images from 0 to 9. → 28×28 image.
- Train.csv → 785 columns. → 1st Label: Digit by user, 784: Each pixel grayscale value from 0 to 255.
- The image is :-

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | .... | 25 | 26 | 27 |
| | 28 | | | | | | | 55 |
| | ⋮ | | | | | | | ⋮ |
| | 728 | | | | | | | 755 |
| | 756 | 757 | . . . . . | | | | 782 | 783 |

- Test.csv has no label.

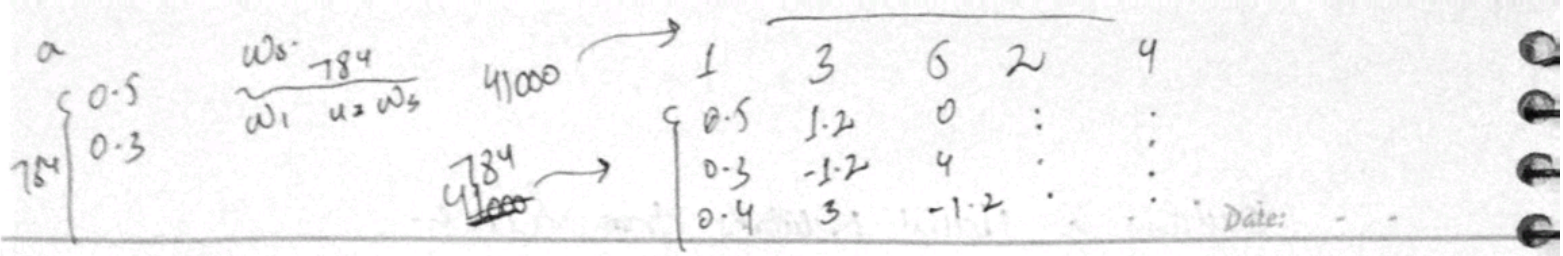
2) Convert it to numpy array and Split dataset. → To prevent overfitting.
   - ↳ linear algebra [we need].
   - → Keep a 1000 digits for validation.
   - → Shuffle the data by np.random.shuffle(data)
   - → Split dataset into val. and training sets. Transpose the datasets.

Why Transpose?

| Data → | Digit | P1 | P2 | . . . | P784. |
|---|---|---|---|---|---|
| We want → | D | | | | |
| | P1 | | | | |
| | ⋮ | | | | |
| | P784 | | | | |

→ This is way of N.N

→ Split each set into label and features.




3) Defining the neural network.

We want a N.N that is:-

N.N { Input Layer. H1 H2 :: O1 ..
784 ns. 16ns. 16ns. 10ns.

So W1 i.e weights for input to H1 will be 784 × 16.

we want a weights 1, array of (16 × 784).

For biases 1, array of (16 × 1).

Thus

W2 → (16, 16) and b2 → (16 × 1).

* weights and biases can be randomized using np.random.rand it gives value b/w 0 and 1. Subtract -0.5 to get +ive and -ive weight.

For Forward Propogation:-

This where we get the activation of K+1 layer by using activation of K^th layer and ws and bs and σ.

→ act. of first hidden layer.

z1 = W1·x + b1 → a1 = relu act.(z1).

z2 = W2·x + b2 → a2 = Softmax(z2).

For Backward Propogation:-

We want to calculate the lost function first. Our digits have specific pixel values and not output values. thus we need to make a function that gives output values of digits.

| | P1 | P2 | ... | P783 |
|---|---|---|---|---|
| r.n :- | 9 | 0 | 0 | 0.7 |
| we want | 9 | 1 | 2 | 3 ... 9 |
| | 0 | 0 | 0 | 1 |

Out_col:- { 10 rows , columns }

↳ value of digit ↳ no. of digits

Leader

How to do it?

Build a function that takes in y_train

Create a output matrix for 41000 digits, initialized by zeroes for all 10 output rows. → 41000 values. 10 values

For e.g 2 → 0 0 1 0 0 0 0 0 0 0

7 → 0 0 0 0 0 0 0 1 0 0

5 → 0 0 0 0 0 1 0 0 0 0

out_eol = np.zeroes( y_train.size , y.max() + 1 ) → we will transpose it later.

↓ ↳ will give 9 add 1 to make 10.

41000 { 0 0 0 0 0 0 0 0 0 0 ; 0 0 0 0 0 0 0 0 0 0 ; ⋮ ; 0 0 0 0 0 0 0 0 0 0 }

10 values.

Now we to populate this matrix based on digit value.

We need to go over each row and populate a 1 for correct digit.

out_eol [ np.arange( y_train.size ) , y_train ] = 1

↳ goes from 0 to 41000

↳ it goes from 0 to 41000 step by step ↳ takes in all digits

Our 2, 7, 5 example. Thus. it becomes. So if its 2, it goes to 2.

{ 0 0 1 0 0 0 0 0 0 0 ; 0 0 0 0 0 0 0 1 0 0 ; 0 0 0 0 0 1 0 0 0 0 }

Then Transpose it

out_eol = out_col.T.

Date:

Mathematics of loss function:-

We need to calculate ~~dw~~ dW and db for all the three layer connections. For that we need to calculate dz for all three too.

For dz3:-

$$L = -\frac{1}{m} \sum y_k \log A_k$$ } for all samples Cross Entropy loss.

By using chain rule

$$\frac{\partial L}{\partial z_3} = \frac{\partial L}{\partial A_3} \times \frac{\partial A_3}{\partial z_3}$$

For $\frac{\partial A_3}{\partial z_3}$, $A_3 = \frac{e^z}{\sum e^z}$

$= A_i(1 - A_i)$ } $i = j$, $e^z$

$= -A_i A_j$ } $i \neq j$

$= A_3(1 - A_3)$ since $A_3$ and $z_3$.

$\frac{\partial L}{\partial z_3} = A_3 - Y$ ← multiplying Jacobian by C.E Loss der.

For dW3:- → we know this

$dW_3 \rightarrow \frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial z_3} \times \frac{\partial z_3}{\partial W_3}$ → $z_3 = W_3(a_2) + b_3$

$= A_2$

$= \{dz_3 \times A_2^T\}/m$ → for all samples.

↳ transpose it

For db3:-

$db_3 \rightarrow \frac{\partial L}{\partial b_3} = \frac{\partial L}{\partial z_3} \times \frac{\partial z_3}{\partial b_3}$ → $z_3 = W_3(a_2) + b_3$

$= 1$.

$= \frac{1}{m} dz$ → but we need to sum the value for each column to give one value for each sample.

$= \frac{1}{m} \sum dz$.

Leader

Date:

For $dz_2$ :-

$$\frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2}$$

For $\frac{\partial L}{\partial A_2}$ :-

$$\frac{\partial L}{\partial A_2} = \frac{\partial L}{\partial z_3} \times \frac{\partial z_3}{\partial A_2}$$

$$= dz_3 \times w_3.$$

For $\frac{\partial A_2}{\partial z_2}$ :-

$$A_2 = \text{relu}(z_2)$$

$$= \text{relu}'(z_2)$$

Thus

$$dz_2 = dz_3 w_3 \times \text{relu}'(z_2).$$

→ transpose it

For $dw_2$ :-

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial w_2} \quad \frac{\partial L}{\partial z_2} \times \frac{\partial z_2}{\partial w_2}$$

→ already calculated.

$$= \frac{\partial z_2}{\partial w_2} = z_2 = w_2 a_1 + b_2.$$

$$dz_2 = a_1.$$

Thus

$$dw_2 = \frac{1}{m} dz_2 \times a_1.$$ → transpose it

For $db_2$ :-

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2} \times \frac{\partial z_2}{\partial b_2} = \frac{1}{m} dz_2 \times 1$$

$$= \frac{1}{m} \Sigma dz_2.$$

For first layer it's same as this one.

Leader

Update the weights and biases:-

$$W_k = W_k - \alpha \, dW_k$$

$$b_k = b_k - \alpha \, db_k.$$