$$\begin{bmatrix} [a & b] \\ [c & d] \\ [e & f] \end{bmatrix} = a$$

$$a^T = \begin{bmatrix} [a & c & e] \\ [b & d & f] \end{bmatrix}$$

Features (2)

Samples (3)

2 × 3

layers = [3, 5, 4]

W matrix in each layer

$$\text{Input} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

2×1

No of nodes will determine the no of rows.

No of features → No. of Columns.

$$W = \begin{bmatrix} w_{00} & w_{01} \\ w_{10} & w_{11} \\ w_{20} & w_{21} \end{bmatrix}$$

3×2

$$y_{pred} = Wx + b$$

No of dimensions of bias is $(1, \text{no of neurons})$

$$= \begin{bmatrix} w_{02} & w_{01} \\ w_{10} & w_{11} \\ w_{20} & w_{21} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

1 Sample

$3 \times 2 \quad 2 \times 1 \quad 3 \times 1$

$$= \begin{bmatrix} b_1 & w_{00} & w_{01} \\ b_2 & w_{10} & w_{11} \\ b_3 & w_{20} & w_{21} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

$3 \times 3 \quad 3 \times 1$

bias term is already added to the weight

So we need to convert each sample from

$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ to $\begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$

this can be acheived by vstacking of concatenating

the array of 1 with

the $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Example Test Case

Total 110 Samples

100 Samples for training and 10 for test

X_train = 4 x 100

4 features each 100

Samples.

Y_train = 1 x 100

1 output each for 100 Samples

X_test = 4 x 10

4 features each 10 Samples

Y_test = 1 x 10

1 output each for 10 Samples.

layers = [3, 5, 4]

1st layer = 3 neurons

2nd layer = 5 neurons

3rd layer = 4 neurons

W in 1st layer

= 3 x 4 + 1

W in 2nd layer

$= 5 \times 3 + 1$

W in 3rd layer

$= 4 \times 5 + 1$

$W x$

1st W x Samples

$3 \times 5 \cdot 4 \times 1$

not possible
after vstacking
3x5 . 5x1

3X1 Total

ie, 3 outputs per
Sample. 1 Sampl
3 outputs.

- 3 sets of weight
matrix are there due

to 3 set of layers.

- Looping through layers to find the output to the next layer.

1st weight X_train

3 x 5 5 x 1

Y = 3 x 1 (1st layer output)

Applying sigmoid(Y)

returns 3 x 1 matrix

Activation in one layer might determine the activations in next layer.

Output of 1st layer

3 x 1

It should get multiplied with the weight + bias of the next layer ie

5 x 4

$W x$

5 x 4 . 3 x 1

Since we added an extra element bias in the weight matrix we need to apply ones (3 x 1) to make it (4 x 1)

$\Rightarrow$ 5 x 1 outputs from 2nd layer.

ie, 5 outputs per sample

For 100 sample 500 outputs

When Comes to 3rd layer. $W = 4 \times 6$

But $O_{2nd} = 5 \times 1$

Applying ons (5×1)

$\Rightarrow 6 \times 1$

Thus $W x$

$4 \times 6 \cdot 6 \times 1$

$\Rightarrow$ 4×1 (output from 3rd layer)

ie, 4 outputs per sample

1 Samples 4 outputs.

yhat = 4×1

Now need to find MSE

wrt to yhat and

y_train

mse(yhat − y_train)

✓

Now updating weight starts.

1) Taking the first weight

$\frac{\partial MSE}{\partial W_{00}}$ — (can be found by computational Graph.

↑

First element

of the 1st weight in the

1st layer

$f(x+h) \cdot f(x-h) / 2h$

f is MSE()

x - the weight that we want to update

ie MSE(wo + h)

Let the MSE of the 1st Time be 25

To find MSE(w00 + h)

if $w_{00} = 1.6$, $h = .2$

add $1.6 + .2$

put this w_{00} in the 1st weight matrix while keeping everything else same and calculate yhat.

when yhat changes

MSE changes. (28.5)

Now find MSE(w00 - h)

ie MSE(1.6 - .2)

Keep the new value in the network while keeping all the values same.

we get new yhat and new MSE ie, 24

Then $$\frac{f(x+h) - f(x-h)}{2h}$$

$$\Rightarrow \frac{MSE(w_{00}+h) - MSE(w_{00}-h)}{2h}$$

$$\frac{28.5 - 24}{2 \times .2} = 11.25$$

$$\Rightarrow \frac{\partial MSE}{\partial w_{00}} = 11.25$$

$$W_{00_{new}} = W_{00_{old}} - \alpha \frac{\partial MSE}{\partial W_{00}}$$

Let $\alpha = 0.5$

$$\Rightarrow W_{00_{new}} = 1.6 - 0.5 \times 11.25$$

$$= -4.025$$

When we are find W_{01} dont update W_{00} new in the place of W_{00} old store it separately.

After everything we will have 3 temporary weight matrix for 3 layers.

Change the original weight matrix will the new temp weights and do it for the 2nd sample

$$W_{new}^{(m)} = [np.copy(;)$$

for in Temp]

Doing Deep Copy otherwise it won't Copy.

When all the 100 samples are passed we complete one epoch.

After 1 epoch use X_test — No more adjusting weights $\left(W_{new}^{(m)}\right)$

1st Sample goes in and we get an MSE

2nd Sample another MSE

Likewise to MSE for 10 samples.

Take mean of 10 MSE and add this to MSE=[] list. This the MSE of One epoch.

Remember to seed the weight matrix when you are initializing it.

After all the epochs are over. We need to find the Output with the final weights of the trained Neural

Network. This is the output of the neural Network.

Use X_test as a matrix itself here to get the Output.

Training a Neural Network is generally minimizing the loss.

- Anything that we train there will be a parameter where we minimize or maximize something.

- Nelson Joseph