04 Oct 2024

WEIGHT INITIALIZATION

Why weight initialization is important?

(i) initialize the parameters
(ii) Choose an optimization algorithm
(iii) Repeat these steps.

- Forward propagate an input
- Compute the cost function.
- Compute the gradients of the cost with respect to parameters using backpropagation
- update each parameter using the gradients, according to the optimization algorithm.

problems if wrong initialization of weights:

- vanishing gradient problem
- Exploding gradient problem
- slow convergence.

what not to do in weight initialization

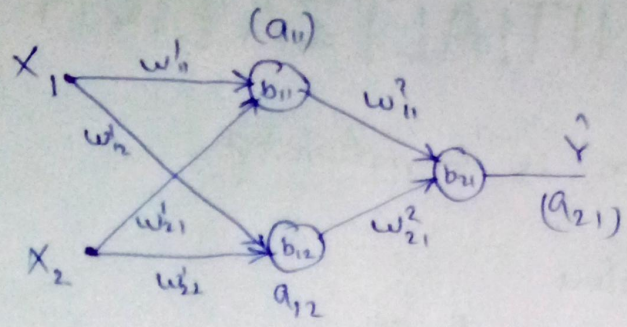
case-1 → zero initialization.

Questions: why we not set the intial value of weight of zero ('0') ?

let us take an example, regression example

assume. $w_0 = 0$ and $b = 0$

(i) ReLU in hidden layer:



Here $a_{11}, a_{12} = a_{21}$ is activation function

$a_{11}, a_{12} \rightarrow$ ReLU

$a_{21} \rightarrow$ linear.

$$a_{11} = \max(0, z_{11})$$

$$z_{11} = w'_{11} x_1 + w'_{21} x_2 + b_{11}$$

$$a_{12} = \max(0, z_{12})$$

$$z_{12} = w'_{12} x_1 + w'_{22} x_2 + b_{12}$$

because $w = 0$ and $b = 0$

Hence,

$a_{11} = 0$, $z_{11} = 0$ and $a_{12} = 0$, $z_{12} = 0$

Hence, $a_{11} = a_{12} = 0$

for weight update

$$w'^{new}_{11} = w'^{old}_{11} - \eta \frac{\partial L}{\partial w'_{11}} \rightarrow 0$$

$$\therefore w'^{new}_{11} = w'^{old}_{11}$$

Here no update

thus there is no training will take place

(ii) Tanh in hidden layers:

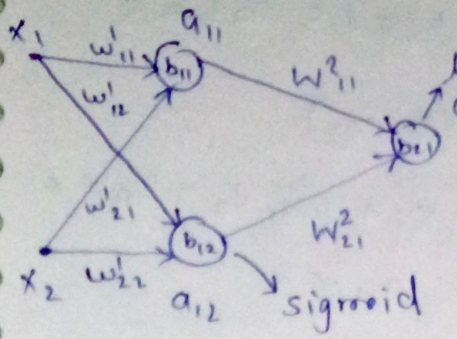
$$a_{11} = \frac{e^{z_{11}} - e^{-z_{11}}}{e^{z_{11}} + e^{-z_{11}}}$$

$\because z_{11} = 0$

$\therefore a_{11} = 0$ and $a_{12} = 0$ also

similar case as in previous one.

(iii) Sigmoid in hidden Layer:



$a_{11} = \sigma(z_{11}) = 0.5$

similarly $a_{12} = 0.5$

but $a_{11} = a_{12}$ also $z_{11} = z_{12}$

weight updation.

$$w = w - \eta \frac{\partial L}{\partial w}$$

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{11}} \cdot \frac{\partial a_{11}}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial w'_{11}} \rightarrow x_1$$

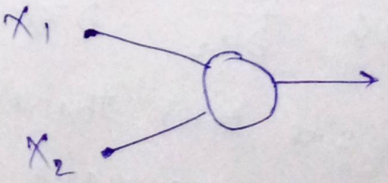
$$\frac{\partial L}{\partial w'_{12}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{12}} \cdot \frac{\partial a_{12}}{\partial z_{12}} \cdot \frac{\partial z_{12}}{\partial w'_{12}} \rightarrow x_1$$

$$\frac{\partial L}{\partial w'_{21}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{11}} \cdot \frac{\partial a_{11}}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial w'_{21}} \rightarrow x_2$$

$$\frac{\partial L}{\partial w'_{22}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{12}} \cdot \frac{\partial a_{12}}{\partial z_{12}} \cdot \frac{\partial z_{12}}{\partial w'_{22}} \rightarrow x_2$$

Hence $\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial w'_{12}}$ and $\frac{\partial L}{\partial w'_{21}} = \frac{\partial L}{\partial w'_{22}}$

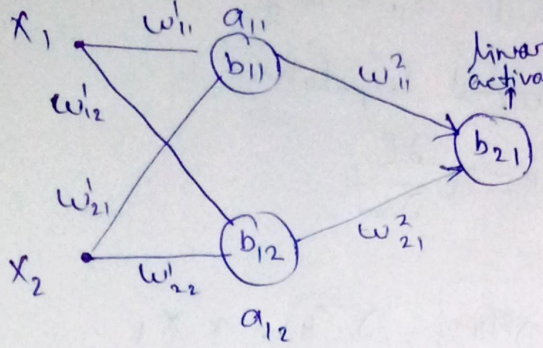
→ The neural network becomes



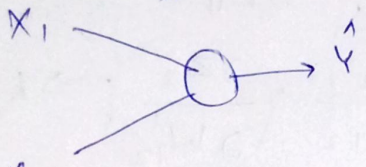
not capture non-linearity because.
work as a perceptron

Case-2 → Non-zero initialization (constant value)

Let us assume $W=0.5$ $b=0.5$



a_{12}



$a_{11} = \max(0, z_{11})$

$z_{11} = w'_{11} x_1 + w'_{21} x_2 + b_{11} \neq 0$

hence $a_{11} \neq 0$

$a_{12} = \max(0, z_{12})$

$z_{12} = w'_{12} x_1 + w'_{22} x_2 + b_{12} \neq 0$

hence $a_{12} \neq 0$

$z_{12} = z_{11} \rightarrow a_{11} = a_{12}$

due to this.

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{11}} \cdot \frac{\partial a_{11}}{\partial z_{11}} x_1$$

$$\frac{\partial L}{\partial w'_{12}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{12}} \cdot \frac{\partial a_{12}}{\partial z_{12}} x_1$$

$$\frac{\partial L}{\partial w'_{21}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{11}} \cdot \frac{\partial a_{11}}{\partial z_{11}} x_2$$

$$\frac{\partial L}{\partial w'_{22}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{12}} \cdot \frac{\partial a_{12}}{\partial z_{12}} \cdot x_2$$

here we see that there is similare case is occur as previous one. (zero initialization of weigh and bias with hidden layer activation function is sigmoid)

case-3 → Random Initialization

Small Random value

np.random.randn(500,500)*0.01

bias(b) = 0

- vanishing GP

large Random value.

range [0,1]

for sigmoid and tanh:
- slow training
- vanishing GP
- saturating values of weights

for ReLU:
- unstable gradient occur.

- vanishing GP (sigmoid, tanh), exploding GP (ReLU)

Weight initialisation Techniques:

weight initialisation (Heuristics approach) Technique

Xavier / Glorot initialisation

Normal
* use when we work with tanh or sigmoid

uniform

He initialisation

Normal
* use when you work with ReLU

uniform.

- till now we know the weight is initialize with Random value but we don't know what it's range. because small random value and large random value can be face a problem.

- small value initialization example
np.random.randn(250,250) * 0.01

- large value initialization example
np.random.randn(250,250) * 1

correct way to initialization example
np.random.randn(250,250) * $\sqrt{\frac{1}{250}}$

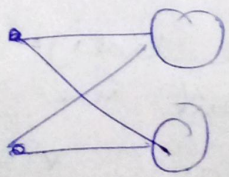
standard deviation
250 is number of input for given particular neuron.

(i) Xavier/Gibrat Initialisation:

→ Normal Xavier/Gibrat initialisation:

$\sqrt{\frac{1}{fan\_in}}$ Here fan_in is number of inputs coming to the node

example.
np.random.randn(2,2) * $\frac{1}{2}$



Or we can also use $\sqrt{\frac{2}{fan\_in + fan\_out}}$

It is used in the case of tanh and sigmoid.

→ uniform distribution Xavier/Glorot initialisation:

$$[-\text{limit}, \text{limit}]$$

$$\text{limit} = \sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}$$

(ii) He initialisation:

→ Normal He initialisation:

$$\sqrt{\frac{2}{\text{fan\_in}}}$$

→ uniform distributed He initialisation:

$$[-\text{limit}, \text{limit}]$$

Here, $$\text{limit} = \sqrt{\frac{6}{\text{fan\_in}}}$$

* all these things are done in the code as a parameter set using kernel_initializer = 'initialize_type', but in the default case.

kernel_initializer = 'glorot_uniform'