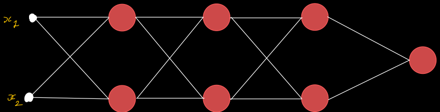
Batch Normalization

-> An algorithmic method which makes training of NN more faster & better.

-> It contains normalizing activation vectors from hidden layer using mean & variance of current batch.

-> This step is applied right before (or right after) the non-linear function.

-> In short we will normalize activation of each neuron.



-> Generally our inputs (x_1, x_2) are normalized ($\mu = 0, \sigma = 1$)

-> In Batch normalization, o/p of each node will be normalized. ($\mu = 0, \sigma = 1$).

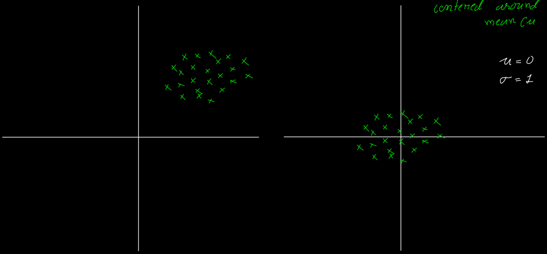
-> This process will be for each hidden layers. makes training fast & stable.

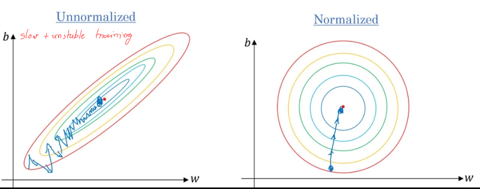
Why use batch normalization..?

-> In NN, it is advisable to normalize data before giving to model.

Example

Original Data   Normalized Data





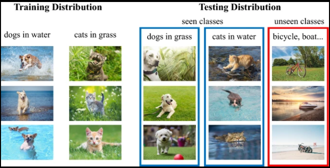
→ Due to unnormalized data, the cost function will look like first image, stretched in one direction.

→ in this scenario, we can't go with large learning-rate, because it may over-shoot in one direction.

→ So if normalized i/p can make training smoother & faster, what if we normalize the o/p of each activation in NN.

Covariate shift

→ A situation where the distribution of i/p changes between the training & testing data, while relationship between i/p & o/p remains same.

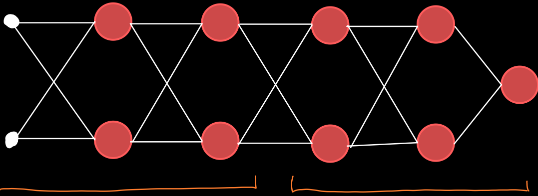


→ this image shows covariate shift
→ As image shows, the model is trained to detect dog or cat in grass or water.
→ But at testing time, model got some unseen objects located in grass or water.

In this situation we need to re-train our model.

Internal covariate shift

"We define internal covariate shift as change in the distribution of network activations due to the change in network parameters during training."



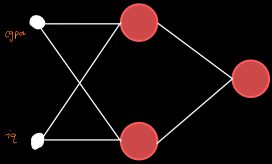
network-1 network-2

→ Here, i/p of network-2 is output of last layer of network-1.
→ The o/p of network-1 is depending on weights & bias of layers.
→ Since weights of network-1 are constantly changing the o/p is also constantly changing.
→ Due to this scenario, i/p distribution of network-2 is constantly changing due to this problem network-2 faces issue in training (unstable training)
→ This problem is called as Internal covariate shift.
→ In case of ICS without data normalization, learning rate should be

How batch normalization works...?

-> Works with mini-batch gradient descent.
-> Applies layer by layer, optional for each layer.

cgpa	iq	placed
6.2	100	1
6.2	89	0
9.1	97	0
7.7	76	1



How process will goes.....

$z_{11} = W_{11}\,cgpa + W_{21}\,iq + b_{11}$

Normalizing z_{11} before further process.

z_{11}^{N} $(u=0, \sigma=1)$

$\boxed{z_{11}^{N} = \frac{z_{11} - u}{\sigma}}$

$g(z_{11}^{N}) = a_{11}$

most popular method

$\boxed{a_{11}}$

OR

$z_{11} = W_{11}\,cgpa + W_{21}\,iq + b_{11}$

$g(z_{11}) = a_{11}$

Now we will normalize this a_{11}

a_{11}^{N}

$\boxed{a_{11}^{N}}$

for $z_{11}^{N} = \frac{z_{11} - u}{\sigma}$, 'u' will be mean of batch.

$$u_B = \frac{1}{m}\sum_{i=1}^{m} z_{11}^{i}$$

where

m = batch size

$$\sigma_B = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(z_{11}^{i} - u_B)}$$

where

m = batch size

$$z_{11}^{i} = \frac{z_{11}^{i} - \mu_B}{\sigma_B + \epsilon}$$

Where

ϵ = Error term

Now we have Z_{11}^{N}, still not done yet, we will do one-more operation here

$$Z_{11}^{BN} = \gamma \cdot Z_{11}^{N} + \beta$$

(in keras)

$$g(Z_{11}^{BN}) = a_{11}$$

Where:

γ and β are learnable parameters.

By default:

$\gamma = 1$

$\beta = 0$

-> This process, we will do for each neurons of a layer...

-> Each neuron will have their own γ and β parameter...

Q. Why we're using γ and β.

->Sometimes we don't need normalized data or our NN doesn't want data to be normalized.

->In this kinds of situations, using γ and β value we can change our distribution.

during Back-propagation, these 2 params will also get updated..

$$\gamma = \gamma - \alpha \frac{\partial L}{\partial \gamma} \qquad \beta = \beta - \alpha \frac{\partial L}{\partial \beta}$$

Advantages

- Fast & stable training
- Regularization effect.
- Reduces weight initialization impact.