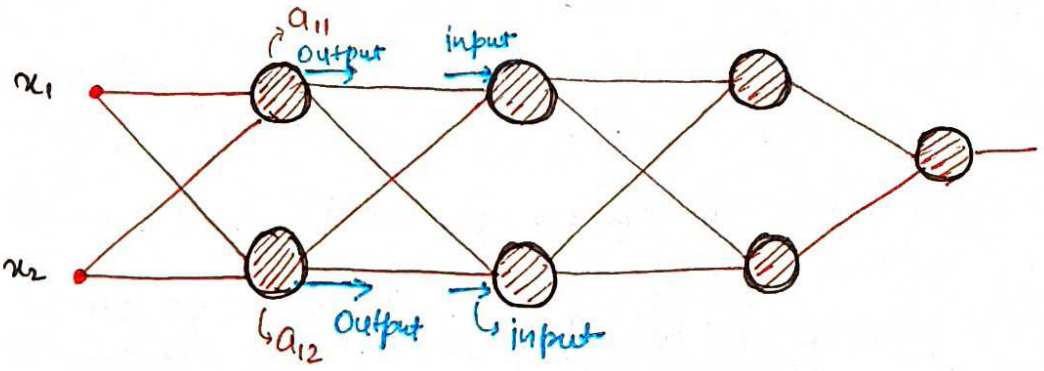
Batch Normalization

What is Batch Norm?

Batch Normalization (BN) is an algorithmic method which makes the training of Deep Neural Network (DNN) faster and more stable.

It consists of normalizing activation vectors from hidden layers using the mean and variance of the current batch. This normalization step is applied right before (or right after) the non linear function.



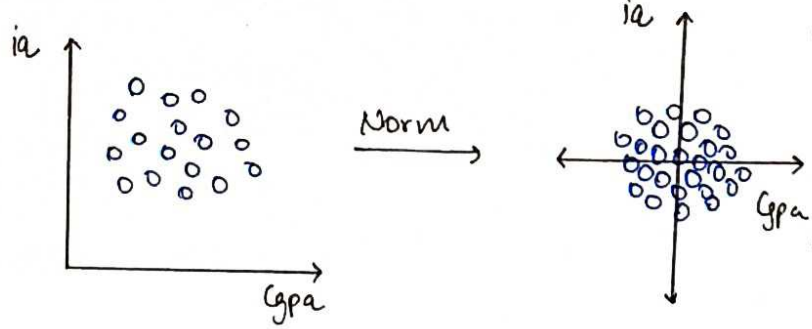
Batch Norm → Normalize the output of the activation fn.

mean = 0 Std = 1

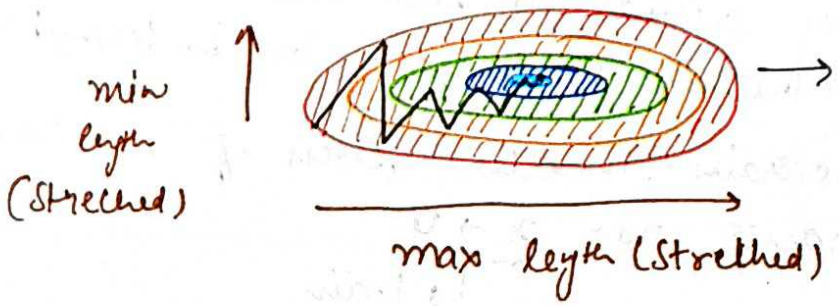
Why use Batch Norm?

Advice → 1. Scaling data
2. Normalization

Cgpa	iq	placed
7	70	1
8	80	0
9	90	1
6	60	0



Contour plot of unnormalized data.

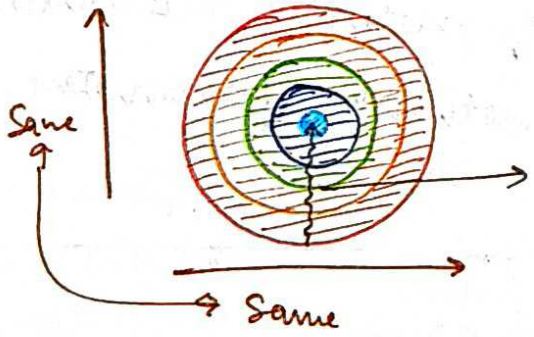


We cannot approach with higher learning rate. Highly chance to overshoot in min length direction.

That's why we use small learning rate and training would be slow.

Un-normalized data → training would be slow.

Contour plot after Normalize the data



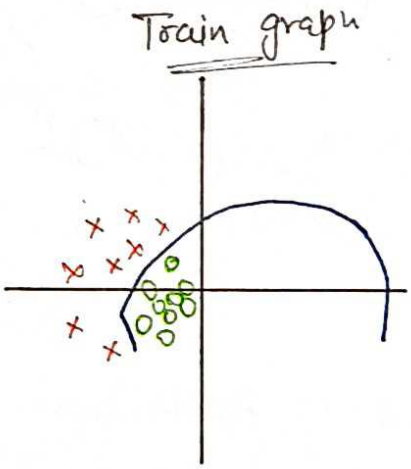
Training is faster and stable.

Covariate Shift

Train data ↴

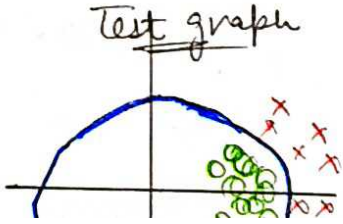
Red colour → Rose Image ⇒ Rose (Y=1)

different flower image ⇒ Not Rose Y=0



Test Data ↴

different color → Rose Image ⇒ Rose (Y=1)



In input colm of test data → distribution changed

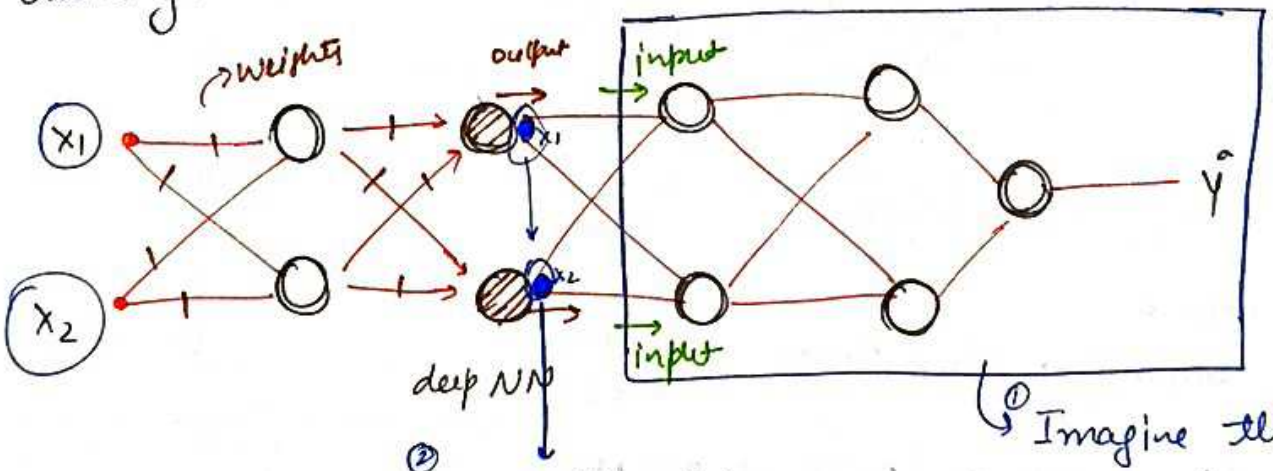
Same decision can divide between red and green points but distribution changed. ~~Bad in testing person~~

we have to retrain model even if relation both $x \to y$ is same as $x \to y$
↳ test ↳ train

or relation is same $x \to y$ still model not perform well. This is called covariate shift.

Internal Covariate

We define Internal covariate shift as the change in the distribution of network activation due to the change in network parameters during training.

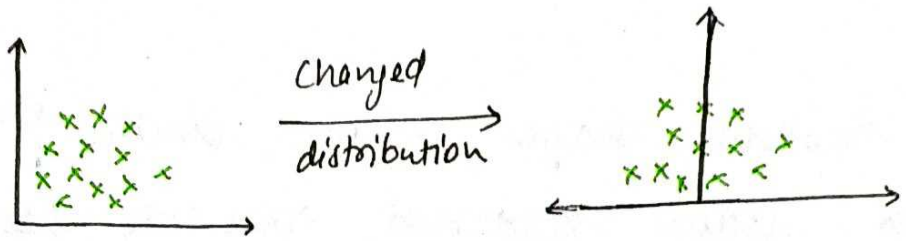


① Imagine this is independent NN.

② Imagine these point are working as x_1 and x_2. But actually these are not input → these are output of Hidden layer.

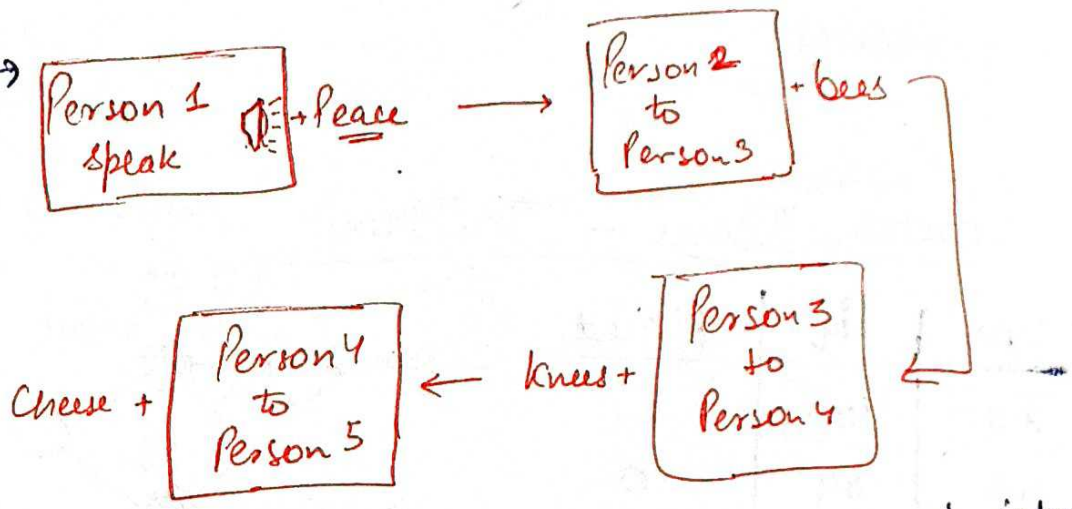
③ Output of Hidden layer depend on previous layer of weights. And weights are constantly changed

distribution ... changing ...



So, Independence NN face difficulties to train and training are unstable.

for example →



In this example first person speak peace and information travel and changed. Last person got information which is Cheese.

Same in Neural Network, Input $(x_1, x_2 \ldots)$ pass the correct information but again and again change in weight and correct information are also changed (distribution changed) and wrong information to $\hat{y}$. Our model not trained correctly. Model not stable.

Here's Batch normalization play important role.

Batch Norm ensure after every activation function (output of activation function) is normaly distributed.

$\mu = 0$, $std = 1$
(σ)

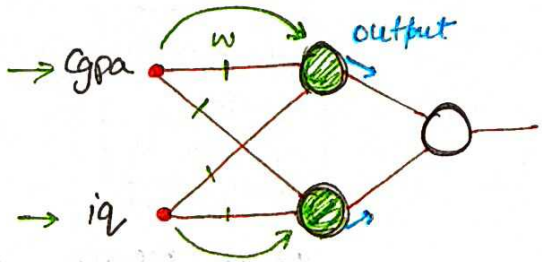
With the help of Batch Norm, Next layer will get stable ground to do their work. And training also improve.

Output of every hidden layer convert into Gaussian so this help to reduce Internal Covariate Shift.

→ If you dont use Batch norm then use small learning rate with Internal covariate shift.

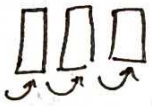
Batch Norm — The How

cgpa	iq	placed
8.9	100	1
6.2	89	0
9.1	91	0
7.7	76	1
⋮	⋮	⋮
6.7	91	0



Bullet points for Batch Norm

→ Mini-batch Gradient descent

→ Layer by layer 

$z_{11} = w_1 \text{cgpa} + w_2 \text{iq} + b$

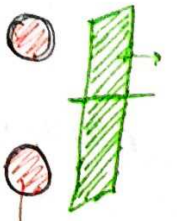
$g(z_{11}) = a_{11}$

Batch_Norm: Normalize the output of hidden layer.

How? ↓ first method ↓

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

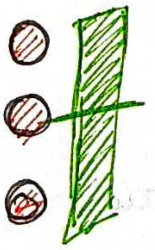
first find the z_{11} then Normalize the z_{11} value of every node in hidden layer.



for every neuron that you have store 4 parameters

- learnable
  - γ
  - β
- Non-learnable
  - μ
  - σ

→ let assume, we have 3 neuron in Hidden layer

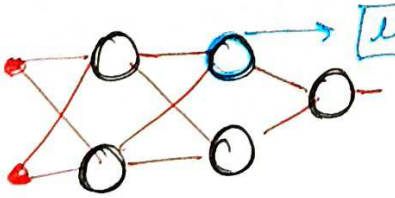


Total Parameter is $3 \times 4 = 12$

- Learnable (6)
- Non-learnable (6)

Advantages

1. Batch Norm makes training more stable.
   - mean → when you use hyperparameter, you have "wider range of values" that you can choose.
2. Training become faster → learning rate value can be set to higher value.
3. Batch Norm act as Regularization (helps in overfitting) ← But no much powerful like Regularization method.

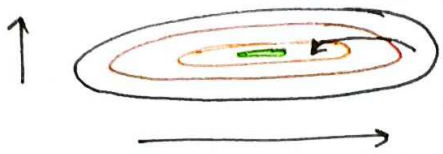


μ, σ → depend on Batch

If batch value change then μ, σ change. And if μ, σ change then activation change and randomness add → reduce overfitting.

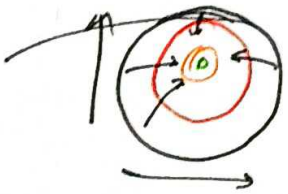
4) Weight init impact reduce

Without Batch Norm



face difficulties to reach optimal solution.

Without Batch Norm



use diff weight still easily you can reach optimal solution

Keras Implementation

```
model = Sequential()
model.add(Dense(3, activation='relu', input_dim=2)),
model.add(BatchNormalization()),
model.add(Dense(2, activation='relu')),
model.add(BatchNormalization())
model.add(Dense(1, activation='sigmoid'))
```

Second Method :

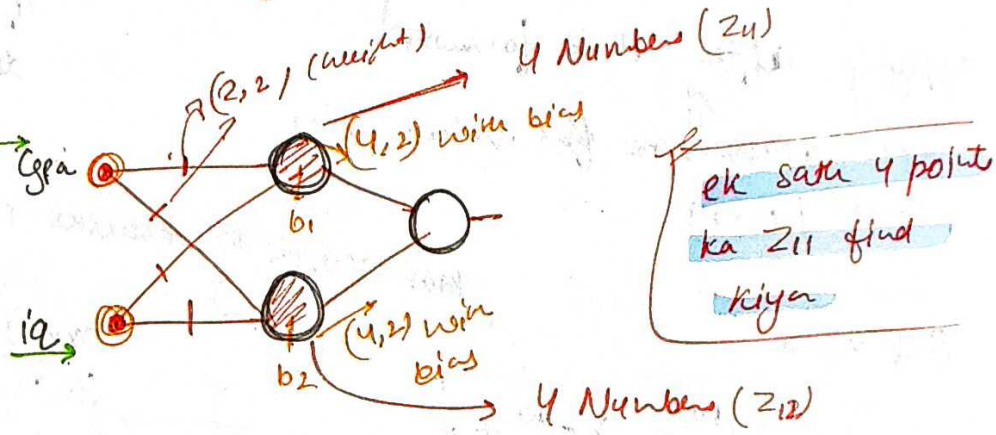
$z_{11} \rightarrow g(z_{11}) = a_{11} \rightarrow a_{11}^N$

first, find the z_{11} of every node of layer and put into $g(z_{11})$ and find the a_{11} value of every node of hidden layer apply normalization.

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

→ how to find μ?

Cgpa	iq	placed
6.2 (1)	89	0
9.1 (2)	91	0
7.7 (3)	76	1
8.9 (4)	100	1
6.7	91	0



Let, batch-size = 4

Take 4 points and Normalize it. Send to NN (ek sath 4 points send kiye)

Input shape :

→ (4×2) and Multiply with (2×2)

$(4\times2) * (2\times2) = (4\times2)$

→ $(4,2) + (1,2)$ (bias) $= (4\times2)$

after addiy bias we get (4,2) of every node.

Now calculate μ,

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

where $m = 4$ ie of batch size.

[ek sath 4 z_{11} find kiye the wahi use karke karna hai]

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

we have to find μ_B and σ_B for two times because NN have 2 node in hidden layer.

Step 1 :-

So, we have 4 z_{11} as per neuron (node).

Now we have to calculate as per activation of per node.

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

↳ add this error term if std is 0 then denominator not get 0.

4 z_{11} the, yaha individual z_{11}^N calculate hoga

applying after this formula all z_{11}^N value lies betn (0-1)

Step 2 :-

har neuron ka khudka γ, β parameter hoga

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

learnable parameter

correct value of γ and β find during training, but initial values is $\gamma = 1$, $\beta = 0$ in keras. after back propagation values update changed.

Step 3 = $g(z_{11}^{BN}) = a_{11}$

Step 4 = a_{111} → activation func^n

Why we use γ and β?

Because in Neural Network some dataset not need to Normalization. So, these value (γ, β) help to reverse the Normalization.

Eg:- Let assume

$\gamma = \sigma + \epsilon$ $\quad \beta = \mu$ $\Longrightarrow$ $Z_{il} \Rightarrow Z_{il}^{N} \rightarrow Z_{il}^{BN}$

these value come

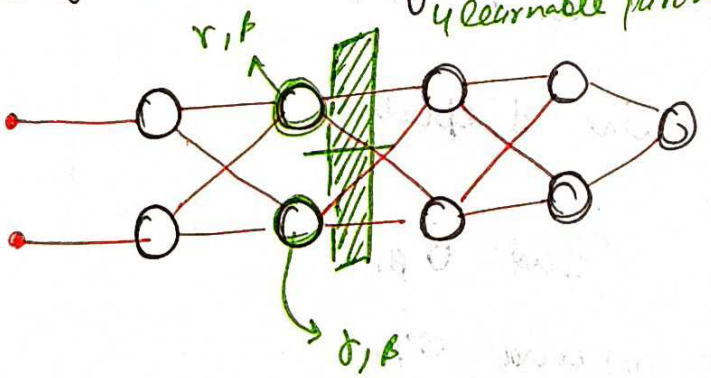
these value come when dataset not need normalization.

both are same

So, reverse the Normalization and back to Z_{il} value.

Keras consider layer → Batch Norm

4 learnable parameter



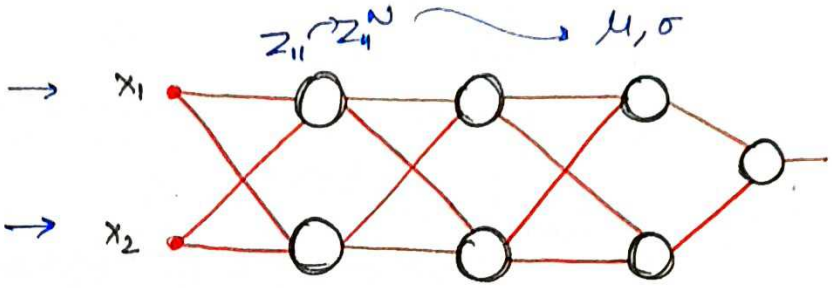
during learning, we can update γ using Gradient descent.

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

Batch Norm During test

cgpa	iq	placed
8	80	1
7	70	0
6	60	1
⋮	⋮	⋮
9	90	1

100 students



In training, we use batch or mini batch like $b=3$ then train first 3 datapoint together. for Normalizing process we find μ and σ with the help of 3 datapoints.

But in test, we have only 1 query point. So, we use Exp weighted (Exponential) moving avg.

let assume,

data → 100 students and $b=4$

So, 25 times data run in 1 epoch

⇒ At batch 1 → find → μ_{B_1} and σ_{B_1}

⇒ At batch 2 → find → μ_{B_2} and σ_{B_2}

⋮

⇒ A batch$_{25}$ → find → $\mu_{B_{25}}$ and $\sigma_{B_{25}}$

EWMA Maintain → μ and β

→ After ~~test~~ training complete, last updated μ and β use in test.