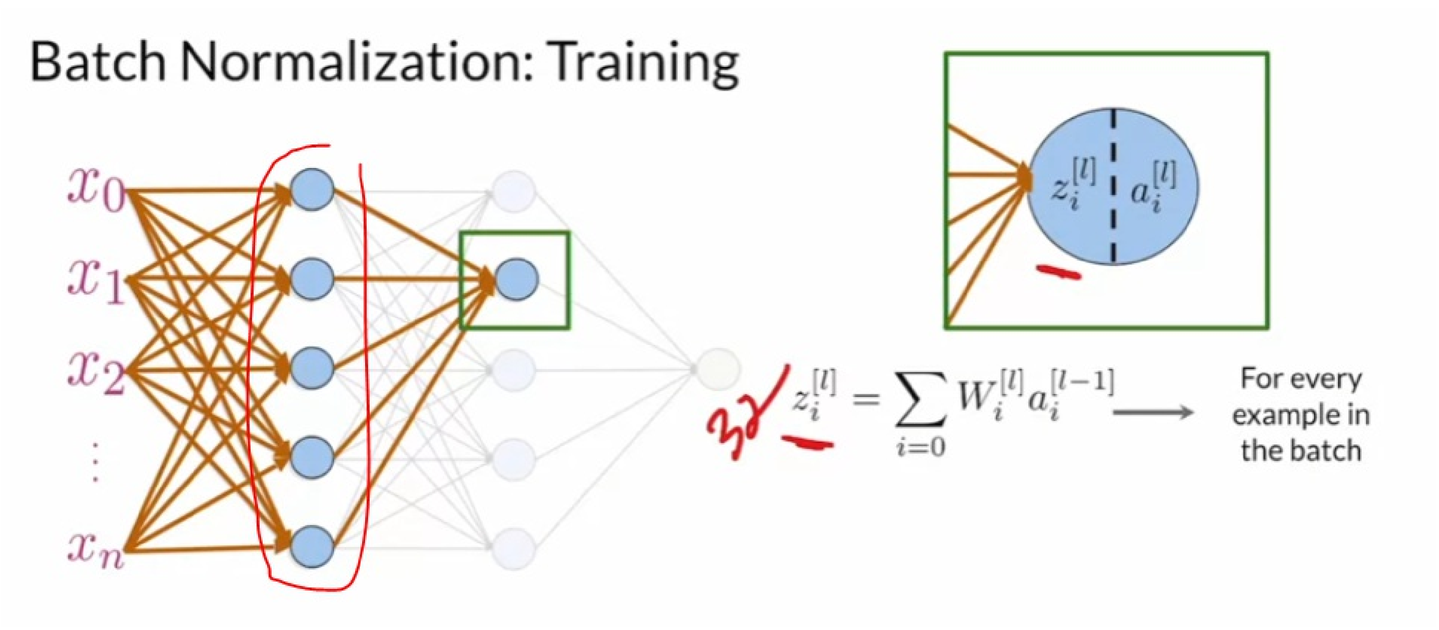
# 4, Batch Normalization (Procedure)

## □ Outline of this video

## □Notes


Batch Normalization: Training
$x_0$
$x_1$
$x_2$
$x_n$
$z_i^{[l]}$
$a_i^{[l]}$
32
$z_i^{[l]} = \sum_{i=0} W_i^{[l]} a_i^{[l-1]} \longrightarrow$
For every example in the batch

- case study (if it is batch size = 32, there will be 32 nodes in this layer)

  if it is batch size = 32, there will be 32 nodes in this layer, then zi[l]z_i^{[l]} zi[l] will have 32 zzz here

$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \mu_{z_i^{[l]}}}{\sqrt{\sigma^2_{z_i^{[l]}} + \epsilon}}$$

>> this equation is to normalized the zi[l]z_i^{[l]}zi[l] to have mean = 0 and std = 1

- zilz^{l}_izil

$$z_i^{[l]} = \sum_{i=0} W_i^{[l]} a_i^{[l-1]}$$

> Z is the outputs of the previous layer (ail−1a^{l-1}_iail−1), weighted by these values W, and this is typically called a linear layer.

>> bbb included in in this WWW

- typically is written like this (with bias (bbb))

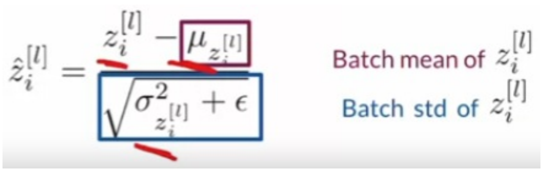
$$s = \sum_{i=1}^{n} w_i a_i + b$$

> > but since bbb can be included in www, so this equation for the node is not used.

- ▼ μzi[l]\mu_{z_i^{[l]}}μzi[l] > > mean of the batch (mean of all (32) the z )
- ▼ σzi[l]2\sigma_{z_i^{[l]}}^2σzi[l]2 > >variance of the batch (sigma square of the values of 32 개 z)
- ▼ ϵ\epsilonϵ > > add this, is to make sure the denominator isnt zero

▼ During training, different batch might have different mean and std > > this will cause unsteady prediction> > so, output of layer need to be normalized

- ▼ z^i[l]\hat{z}_i^{[l]}z^i[l] > > zi[l]z_i^{[l]}zi[l] after being normalized

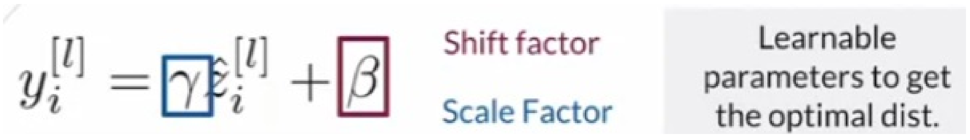


$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \boxed{\mu_{z_i^{[l]}}}}{\boxed{\sqrt{\sigma^2_{z_i^{[l]}} + \epsilon}}}$$

Batch mean of $z_i^{[l]}$

Batch std of $z_i^{[l]}$

- ▼ equation that can rescale the z_hat with the learned value (γ,β\gamma, \betaγ,β)



$$y_i^{[l]} = \boxed{\gamma}\hat{z}_i^{[l]} + \boxed{\beta}$$

Shift factor

Scale Factor

Learnable parameters to get the optimal dist.

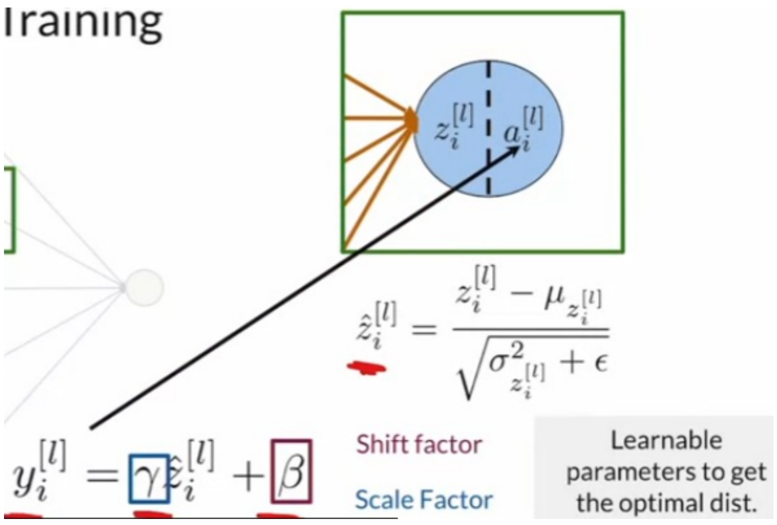
> > this equation include the learned parameter such as γ,β\gamma, \betaγ,β

β\betaβ > > shift factor

γ\gammaγ > >scale factor

> **These parameters (γ,β\gamma,\betaγ, β)are learned** during training to ensure that the distribution to which you're transforming z is the optimal one for your task.

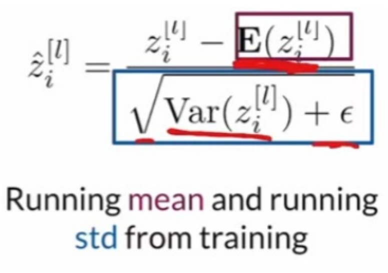
- ▼ yyy > >value of the value of normalized value, being rescaled (this yyy, is what then goes into the activation function (aila^{l}_iail))



Training
$z_i^{[l]}$
$a_i^{[l]}$
$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \mu_{z_i^{[l]}}}{\sqrt{\sigma^2_{z_i^{[l]}} + \epsilon}}$
$y_i^{[l]} = \boxed{\gamma}\hat{z}_i^{[l]} + \boxed{\beta}$
Shift factor
Scale Factor
Learnable parameters to get the optimal dist.

> > y is the value will goes into this activation function

▼ During testing, (similar equation)



$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \boxed{\mathbf{E}(z_i^{[l]})}}{\boxed{\sqrt{\mathrm{Var}(z_i^{[l]}) + \epsilon}}}$$

Running mean and running std from training

E(zi[l])E(z_i^{[l]})E(zi[l])>> expected value of those z values (mean)

- ▼ Var(zi[l])Var(z_i^{[l]})Var(zi[l])>>variance of those z values

  take square root, then the denominator will be the std (You still have that ϵ\epsilonϵ here to prevent that denominator from going to zero)

> dont worry about the framework or value of this (equation), framework like TensorFlow or Pytorch keep track of these statistics for you.

## □Vocabs

- ▼ terms that equavalent for `NOT TRAINING TIME`:
  - test time
  - inference time
  - eval mode
  - evaluation mode

## □ QOTD

## □Summary

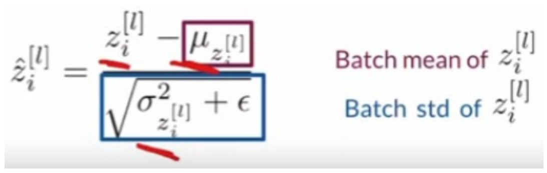
batch >> that whole layer (i.e hidden layer, and that layer for example, got 32 nodes)

- ▼ batch normalization step
  1. normalize the node, (to make the distribution of nodes in that batch, has mean of zero, and std of 1)
  2. then rescale the node

  >> (thru batch normalization, gives you control over what that distribution will look like moving forward in the neural network, and this final value after the shifting and scaling will be called y.)

- ▼ Batch norm introduces learnable shift and scale factors (γ,β\gamma, \betaγ,β)

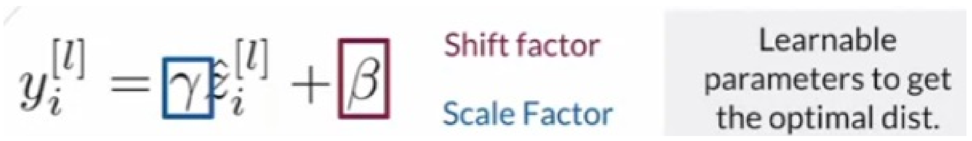
  >> the reason of having these factors, is because you don't force the target distribution to have a zero mean and a standard deviation of one.



$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \boxed{\mu_{z_i^{[l]}}}}{\boxed{\sqrt{\sigma^2_{z_i^{[l]}} + \epsilon}}}$$

Batch mean of $z_i^{[l]}$

Batch std of $z_i^{[l]}$



$$y_i^{[l]} = \boxed{\gamma}\hat{z}_i^{[l]} + \boxed{\beta}$$

Shift factor

Scale Factor

Learnable parameters to get the optimal dist.

- ▼ During test, the running statistics from training are used

> **Running statistics from training are applied to the entire data set**, which keeps

> predictions stable because the training values are independent and fixed.

$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \boxed{\mathbf{E}(z_i^{[l]})}}{\boxed{\sqrt{\mathrm{Var}(z_i^{[l]}) + \epsilon}}}$$

Running mean and running std from training

- Frameworks take care of the whole process (example for framework, TensorFlow Pytorch)