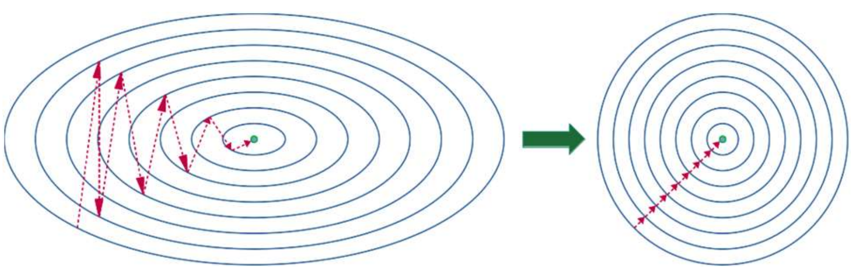
BATCH NORMALIZATION

Introduction to Machine Learning, Artificial Intelligence and Deep Learning
Session 14
Pramod Sharma
pramod.sharma@prasami.com

2

Batch Normalization

- It definitely helps to normalize input data
- Gradient converges faster

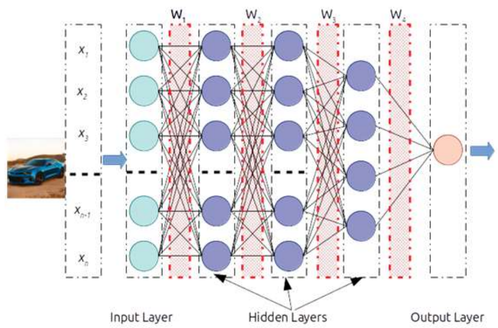


7/10/2023

3

Batch Normalization

- What about hidden layer?
- After all activations from previous layer are inputs for current layer…



- Will it help if we normalize the hidden layers too?

7/10/2023

4

Batch Normalization

- Batch normalization (also known as batch norm) [by Sergey Loffe and Christian Szegedy in 2015]
  - Make artificial neural networks faster
  - More stable through normalization of the input layer by re-centering and re-scaling
  - Wider choices of hyper- parameter…
- In theory, its normalizing activation values of the respective layers
- In practice, it works better if we normalize ‘z’
  - Look at the documentation for details

7/10/2023

5

Batch Normalization

- In General, any Z^i can be normalized

  mean $\mu = \frac{\sum Z^i}{m}$

  std $\sigma^2 = \frac{1}{m}\Sigma(Z^i - \mu)^2$

6

Batch Normalization

- In General, any Z^i can be normalized

  mean $\mu = \frac{\sum Z^i}{m}$

  std $\sigma^2 = \frac{1}{m}\Sigma(Z^i - \mu)^2$

  $z^i_{Norm} = \frac{z^i - \mu}{\sqrt{\sigma^2}}$

  $\hat{z} = \gamma \, . \, z^i_{Norm} + \beta$

- where γ and β are parameters, we can **Train**

Instead of using $z^i_{Norm,}$ researchers realized that its better to derive $\hat{z}$ with two trainable parameters.

Intuition is that by normalizing z, we are introducing bias in the system. Hence it makes sense to train these parameters

7

Batch Normalization

- In General, any Z^i can be normalized

  mean $\mu = \frac{\sum Z^i}{m}$

  std $\sigma^2 = \frac{1}{m}\Sigma(Z^i - \mu)^2$

  $z^i_{Norm} = \frac{z^i - \mu}{\sqrt{\sigma^2 + \varepsilon}}$

  $\hat{z} = \gamma \, . \, z^i_{Norm} + \beta$

- where γ and β are parameters, we can train

- if $\gamma = \frac{1}{\sqrt{\sigma^2 + \varepsilon}}$ and $\beta = \frac{\mu}{\sqrt{\sigma^2 + \varepsilon}}$; $z^i_{Norm} = \hat{z}^i$

Lets add a small ε to prevent zero divide error…

7/10/2023

8

Batch Normalization

- In General, any Z^i can be normalized

  mean $\mu = \frac{\sum Z^i}{m}$

  std $\sigma^2 = \frac{1}{m}\Sigma(Z^i - \mu)^2$

$z^i_{Norm} = \frac{z^i - \mu}{\sqrt{\sigma^2 + \varepsilon}}$

$\hat{z} = \gamma \, . \, z^i_{Norm} + \beta$

- where γ and β are parameters, we can train

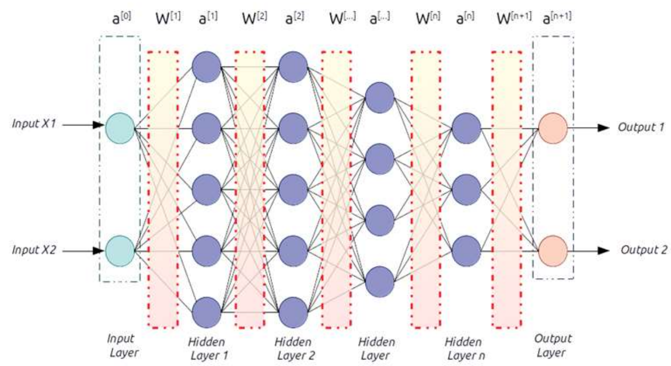
- if $\gamma = \frac{1}{\sqrt{\sigma^2 + \varepsilon}}$ and $\beta = \frac{\mu}{\sqrt{\sigma^2 + \varepsilon}}$; $z^i_{Norm} = \hat{z}^i$

7/10/2023

9

Batch Normalization

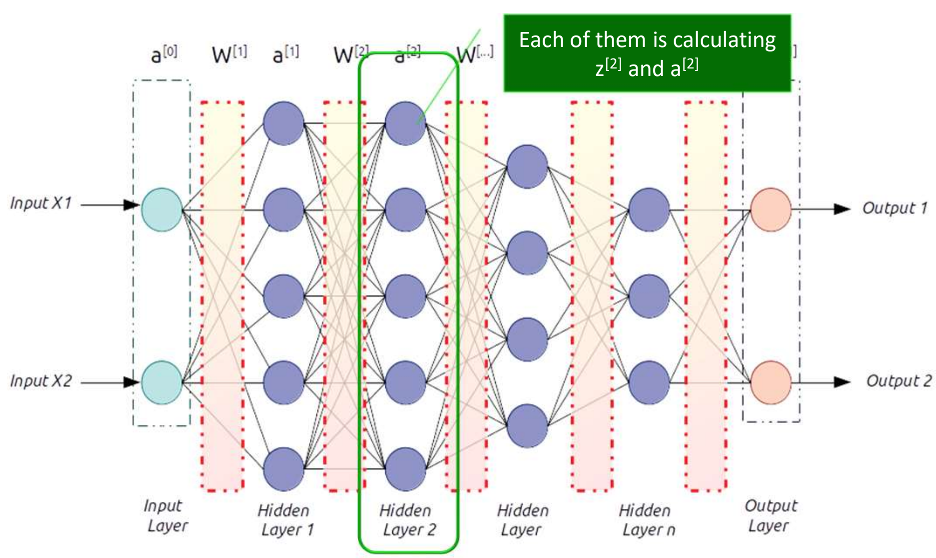
- Notes:
  - Batch norm is used along with mini batches
  - Batch norm is applied to the batch under consideration only irrespective of other mini batches
- Where does it fit in overall scheme?



7/10/2023

10

Batch Normalization

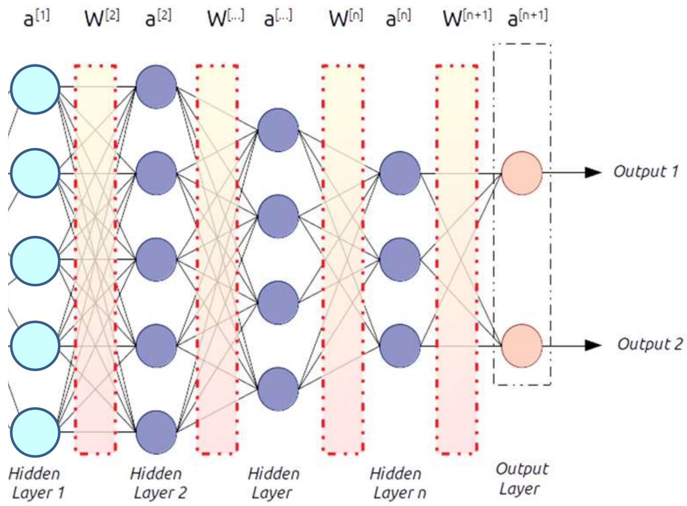


7/10/2023

11

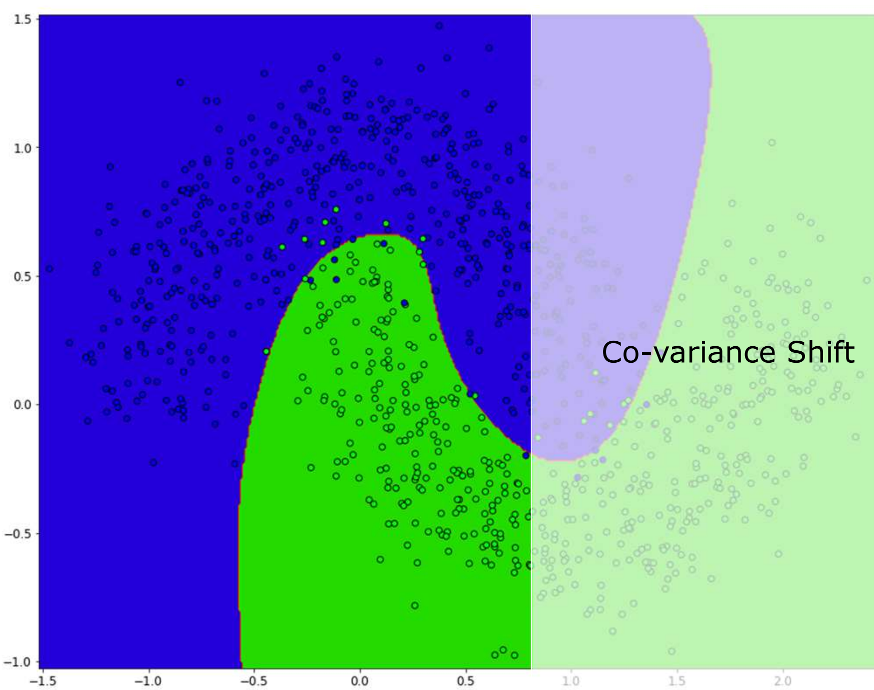
Batch Normalization

For $a^{[2]}$ all $a^{[1]}$ are acting as input features



7/10/2023

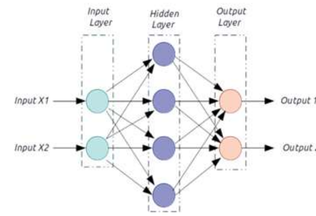
12



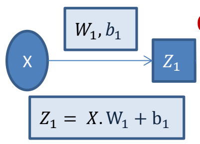
7/10/2023

13

Batch Normalization



- Forward and back propagation with batch norm:

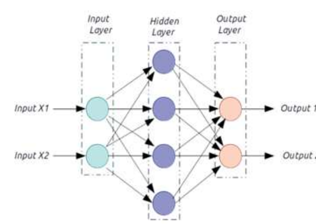


Our standard equation to calculate z_1.

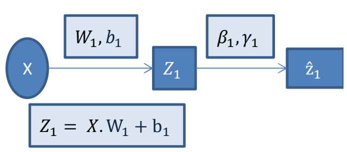
7/10/2023

14

Batch Normalization



- Forward and back propagation with batch norm:



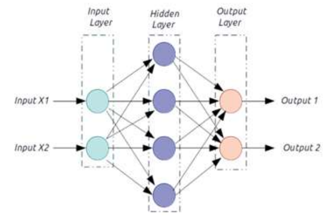
Calculate $\hat{z}_1$, based on β_1, γ_1

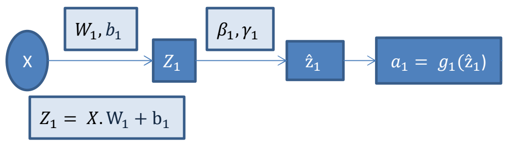
7/10/2023

Batch Normalization

15

- Forward and back propagation with batch norm:





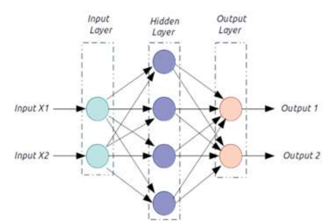
Apply activation function $g_1(\hat{z}_1)$

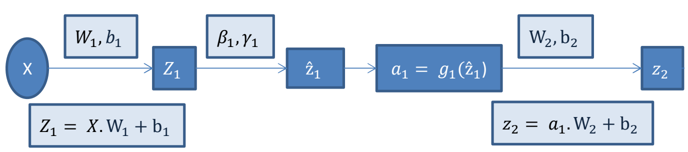
7/10/2023

Batch Normalization

16

- Forward and back propagation with batch norm:





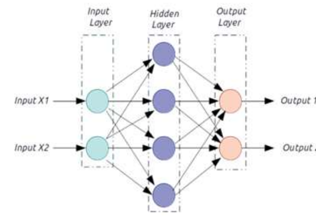
Calculate z_2 as usual

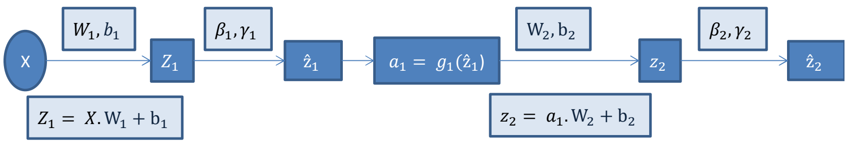
7/10/2023

17

Batch Normalization

- Forward and back propagation with batch norm:





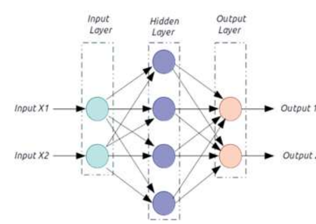
We know how to calculate $\hat{z}_2$

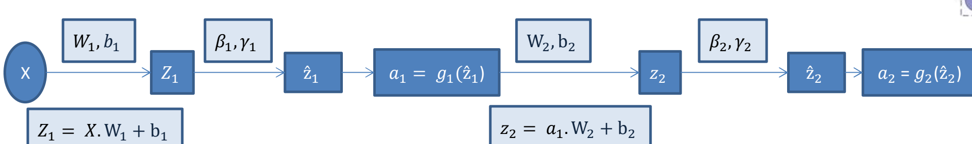
7/10/2023

18

Batch Normalization

- Forward and back propagation with batch norm:





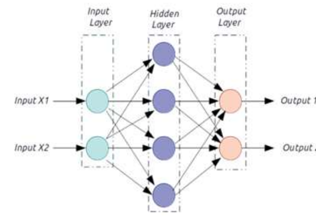
We also know how to calculate a_2

7/10/2023

19

Batch Normalization

- Forward and back propagation with batch norm:



W_1, b_1 β_1, γ_1 W_2, b_2 β_2, γ_2

X → Z_1 → $\hat{z}_1$ → $a_1 = g_1(\hat{z}_1)$ → z_2 → $\hat{z}_2$ → $a_2 = g_2(\hat{z}_2)$

$Z_1 = X.W_1 + b_1$

$z_2 = a_1.W_2 + b_2$

$\beta_1 = \beta_1 - \alpha . \partial \beta_1$
$\gamma_1 = \gamma_1 - \alpha . \partial \gamma_1$

Using the gradient descent, update β's, γ's along with W's and b's

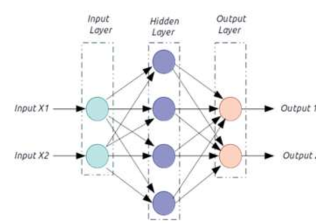
$\beta_2 = \beta_2 - \alpha . \partial\beta_2$
$\gamma_2 = \gamma_2 - \alpha . \partial\gamma_2$

7/10/2023

20

Batch Normalization

- Forward and back propagation with batch norm:



$W_1, \cancel{b_1}$ β_1, γ_1 $W_2, \cancel{b_2}$ β_2, γ_2

X → Z_1 → $\hat{z}_1$ → $a_1 = g_1(\hat{z}_1)$ → z_2 → $\hat{z}_2$ → $a_2 = g_2(\hat{z}_2)$

$Z_1 = X.W_1 \cancel{+ b_1}$

$z_2 = a_1.W_2 \cancel{+ b_2}$

$\beta_1 = \beta_1 - \alpha . \partial \beta_1$
$\gamma_1 = \gamma_1 - \alpha . \partial \gamma_1$

$\beta_2 = \beta_2 - \alpha . \partial\beta_2$
$\gamma_2 = \gamma_2 - \alpha . \partial\gamma_2$

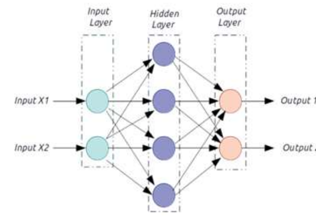
One more thing, since we are normalizing our Z's, keeping b's in the equation does not make any sense now. Being the constant it will get eliminated!!

7/10/2023

21

Batch Normalization

- Forward and back propagation with batch norm:



X → Z_1 → $\hat{z}_1$ → $a_1 = g_1(\hat{z}_1)$ → z_2 → $\hat{z}_2$ → $a_2 = g_2(\hat{z}_2)$

$W_1, \cancel{b_1}$ $\quad$ β_1, γ_1 $\quad$ $W_2, \cancel{b_2}$ $\quad$ β_2, γ_2

$Z_1 = X.W_1 \cancel{+ b_1}$

$z_2 = a_1.W_2 \cancel{+ b_2}$

$\beta_1 = \beta_1 - \alpha . \partial \beta_1$
$\gamma_1 = \gamma_1 - \alpha . \partial \gamma_1$

$\beta_2 = \beta_2 - \alpha . \partial\beta_2$
$\gamma_2 = \gamma_2 - \alpha . \partial\gamma_2$

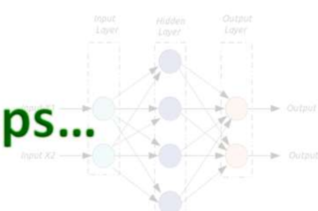
And at test/validation time using a exponentially weighted average!
So while training do not forget to save exponentially weighted values or simply running average!!

7/10/2023

22

Batch Normalization

- Forward and back propagation with batch norm:



Too many calculation steps…

Don't worry…

Most frameworks have one line code to do it.

7/10/2023

23

Batch Normalization – Code Sample

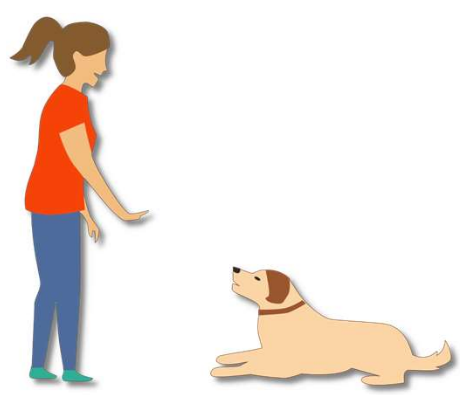
```
model = tf.keras.models.Sequential(
      [
        tf.keras.layers.RNN( keras.layers.LSTMCell(units), input_shape=(None, input_dim)  ),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dense(output_size),
      ]
   )

class Net(nn.Module):
        def __init__(self):
        super(Net, self).__init__()
        self.dense1 = nn.Linear(in_features=320, out_features=50)
        self.dense1_bn = nn.BatchNorm1d(50)
        self.dense2 = nn.Linear(50, 10)
```

- And it is applied to mini batches only….
- Batch Norm can be updated using any of the optimization functions…

7/10/2023

24

Batch Normalization



Remember β, γ *are parameters you train*!

7/10/2023

25

7/10/2023