UE21CS343BB2
Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD),
Centre for Data Sciences & Applied Machine
Learning (CSSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack: Anirudh Chandrasekar,

Teaching Assistant

Topics in Deep Learning

Batch Normalization

Tuning Hyperparameters

There are multiple hyperparameters to look at when it comes to deep networks such as:

- number of layers
- number of hidden units
- learning rate decay
- mini-batch size
- dropout rate etc

It is hard to decide which hyperparameter is the most important in a problem as it depends a lot on your problem.

- Try random values

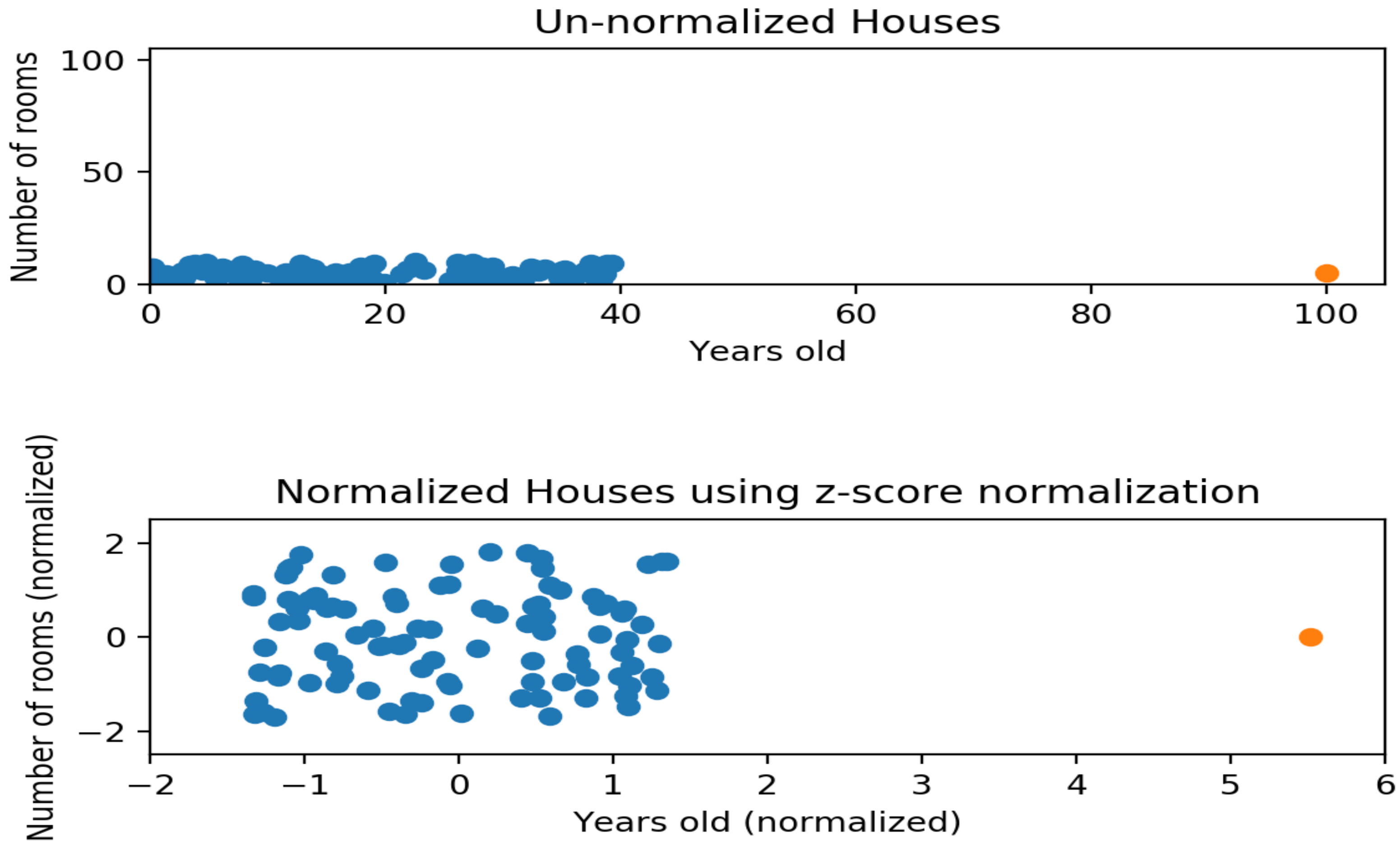
Why Normalization

- If the features are on different scale (1,1000) and (0,1),then the weights will end up taking different values.
- More steps may be needed to reach optimal value and the learning can be slow.

Vanishing and Exploding Gradients

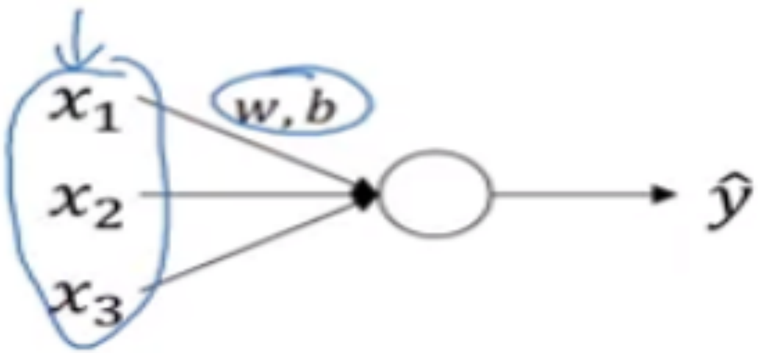
- The Vanishing and Exploding gradients occurs when your derivatives become very small or very big. So it important how we initialize our weights.

Why Normalization

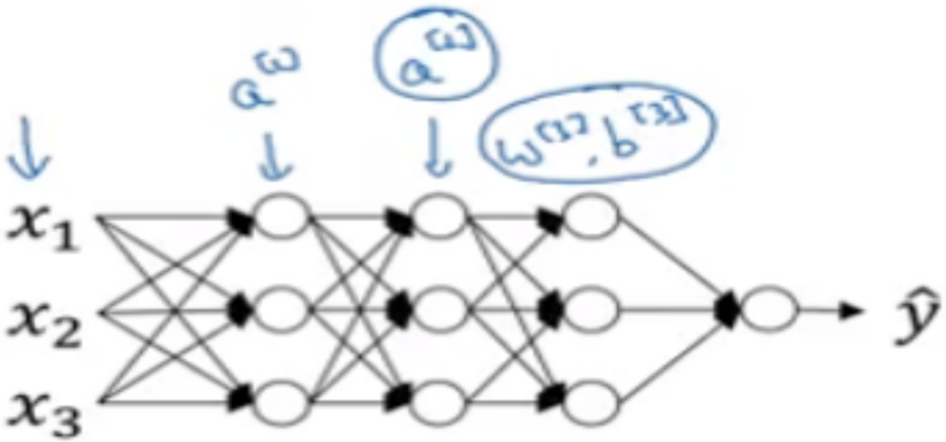


Normalizing Activations in a Network

Normalizing inputs is done to speed up learning



This is done by computing the mean and variance of the input and then subtracting the mean and normalizing the data by the variance.



Considering a deep network, Can we normalize $a^{[2]}$ (or any hidden layer) so as to train $w^{[3]}$, $b^{[3]}$ faster?

- This is what batch normalization does.
- But we normalize $z^{[2]}$ instead of $a^{[2]}$ i.e. before applying the activation function.

Implementing Batch Normalization

Given some intermediate values of the NN $z^{(1)},z^{(2)},z^{(3)},....z^{(i)}$ for some hidden layer l.

$$\mu = \frac{1}{m} \sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m} \sum_i (z_i - \mu)^2$$

$$z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

where

- $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\sigma}^2$ is the variance.
- $\boldsymbol{\varepsilon}$ is added to ensure mathematical stability. (incase variance turns out to be 0).

Implementing Batch Normalization

PES UNIVERSITY

Now we have normalized the values of z such that they have mean 0 and standard unit variance.

But we do not want this to be the case always and might want them to have a different distribution.

So we compute,

$$\tilde{Z}^{(i)} = \boldsymbol{\gamma}\, Z^{(i)}_{norm} + \boldsymbol{\beta}$$

where **γ** and **β** are learnable parameters of the model.

Question: For what values of **γ** and **β** will $\tilde{Z}^{(i)} = Z^{(i)}$?

Ans: $\boldsymbol{\gamma} = \sqrt{\boldsymbol{\sigma}^2 + \boldsymbol{\varepsilon}}$ and $\boldsymbol{\beta} = \boldsymbol{\mu}$

Adding Batch Normalization to a Network

Using Batch Norm in 3 hidden layers NN:

```
  W[1],b[1]         beta[1], alpha[1]                   W[2],b[2]        beta[2], alpha[2]
X ==========> Z[1] ===================> Z_n[1] ====> A[1] ==========> Z[2] ===================> Z_n[2]
                     batch norm.                                          batch norm.
```

Our NN parameters will be:

`W[1]`, `b[1]`, ..., `W[L]`, `b[L]`, `beta[1]`, `gamma[1]`, ..., `beta[L]`, `gamma[L]`

`beta[1]`, `gamma[1]`, ..., `beta[L]`, `gamma[L]` are updated using any optimization algorithms (like GD, RMSprop, Adam)

If you are using a deep learning framework, you won't have to implement batch norm yourself:

- Ex. in Tensorflow you can add this line: tf.nn.batch-normalization()

Why Batch Normalization works?

- While updating weights in a multilayered Neural network, we update a given layers weights under the assumption that the weights of the prior layers have a given distribution.
- This distribution is likely changed after the weights of the prior layer are updated.
- The authors of the paper introducing batch normalization refer to change in the distribution of inputs during training as "**internal covariate shift.**"
- We define **Internal Covariate Shift** as the change in the distribution of network activations due to the change in network parameters during training.

Why Batch Normalization works?

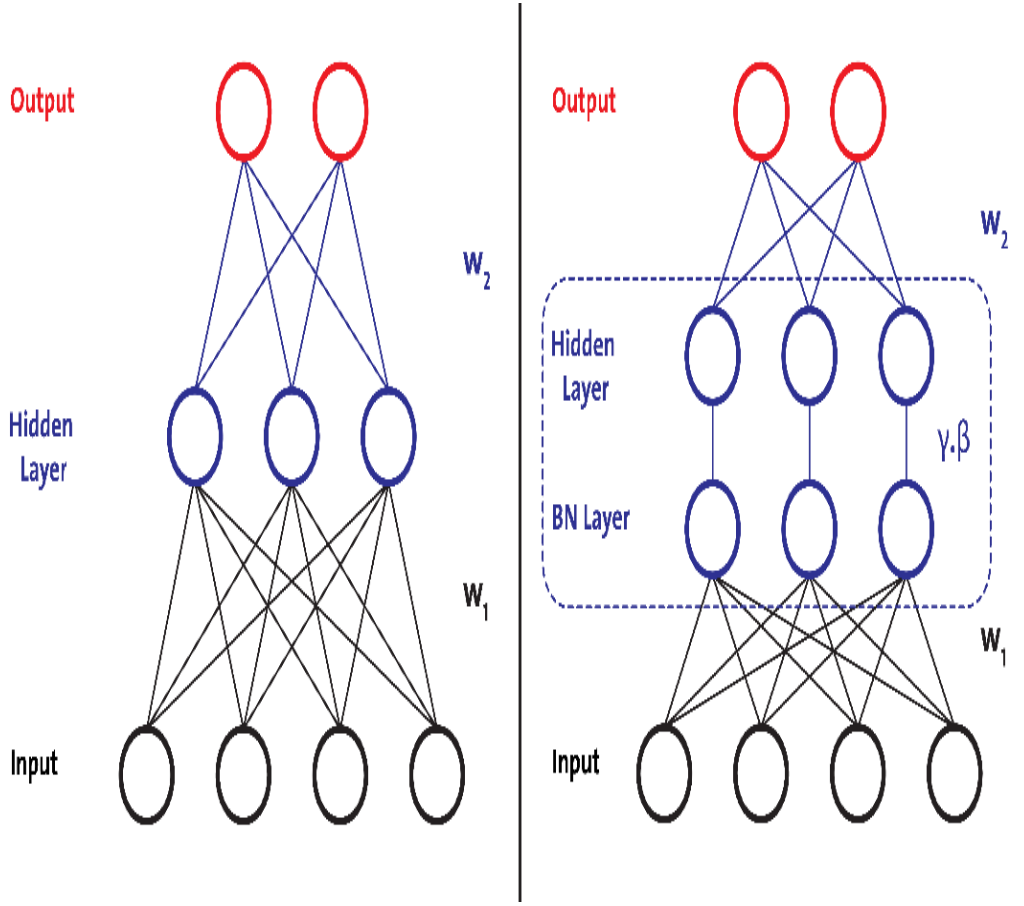
Therefore, we normalize the outputs every layer, intuitively, we are forcing them to have uniform distribution throughout training.

- Standardizing the activations of the prior layer essentially means that the assumptions the subsequent layer makes about the spread and distribution of inputs during the weight update will not change, at least not dramatically.
- This has the effect of stabilizing and speeding-up the training process of deep neural network.

PES UNIVERSITY

Advantages of Batch Normalization

- Networks train faster.
- Allows for higher learning rates.
- Makes weights easier to initialise.
- Provides some regularization.



Acknowledgements & References

- https://deeplearning.ai
- https://medium.com/@rishavsapahia/5-min-recap-for-andrew-ng-deep-learning-specialization-course-2-8a59fd58ca0d

UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD),
Centre for Data Sciences & Applied Machine Learning (CSSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack: Anirudh Chandrasekar,

Teaching Assistant