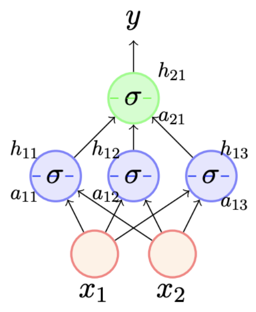
Initialisation Strategies

Badrinath Singhal

March 2021

Here we will see what happens to neural network when we initialise weights with different strategies before training.

Let's consider a neural network with 1 hidden layer with 3 neurons (it will work with any number of layer and neuron). Also let's assume that we are using sigmoid non linearity and use gradient descent to parameter update. Below is the image which describes the network.



$$
\begin{gathered}
a_{11} = w_{11}x_1 + w_{12}x_2 \\
a_{12} = w_{21}x_1 + w_{22}x_2 \\
a_{13} = w_{31}x_1 + w_{32}x_2 \\
h_{11} = \sigma(a_{11}) \\
h_{12} = \sigma(a_{12}) \\
h_{13} = \sigma(a_{13}) \\
\nabla w_{11} = \frac{\partial L(\mathbf{W})}{\partial y} \cdot \frac{\partial y}{\partial h_{11}} \cdot \frac{\partial h_{11}}{\partial a_{11}} . x_1 \\
\nabla w_{12} = \frac{\partial L(\mathbf{W})}{\partial y} \cdot \frac{\partial y}{\partial h_{12}} \cdot \frac{\partial h_{12}}{\partial a_{12}} . x_1
\end{gathered}
$$

We will consider different cases of initialisation neural networks weights.

Case 1: All weights to 0

What will happen if all weights are initialised to 0? All the neurons will get initialised to same weights.

$$
\begin{gathered}
a_{11} = 0 \\
a_{12} = 0 \\
a_{13} = 0 \\
\text{Which means } h_{11} = h_{12} = h_{13} \\
\nabla w_{11} = \nabla w_{12}
\end{gathered}
$$

This means that both weights will get the same gradient and thus step taken by them will be equal resulting in same weights.

This symmetry will never break during training and is true for for all weights.

This will also happen if we initialise weights to any other value as long as all the weights are equal.

Case 2: All weights to a small number

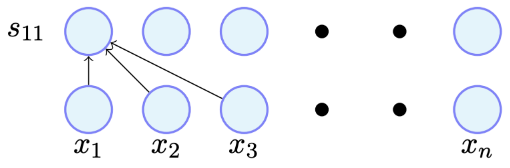
Let's initialise all weights close to zero (here weights can be different from each other). Now lets see what happens in back propagation

We saw from relation defined above that ∇w_1 is proportional to the activation passing through it. So if all the activations are close to 0 then the gradient of the weights connected to this layer will also be close to 0 which will create vanishing gradient problem.

Case 3: All weights to a large number

Here if we initialise weights to large numbers, more activations will get saturated thus the gradients will all be close to 0 which will again lead to vanishing gradient problem.

Case 4: Principled way of initialising weights



Consider above image, we have made few assumptions here. We assume that input and weights are 0 mean. We also assume that $Var(x_i) = Var(x) forall x_i$

$$s_{11} = \sum_{i=1}^{n} w_{1i}x_i$$

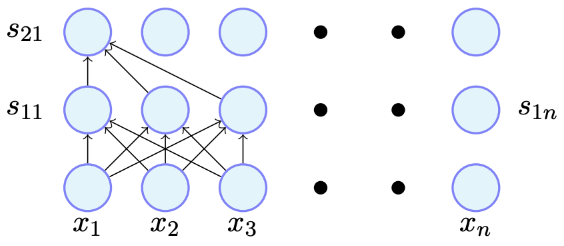
$$Var(s_{11}) = Var(\sum_{i=1}^{n} w_{1i}x_i) = \sum_{i=1}^{n} Var(w_{1i}x_i)$$

$$Var(s_{11}) = \sum_{i=1}^{n}[(E[w_{1i}])^2 Var(x_i) + (E[x_i])^2 Var(w_{1i}) + Var(x_i)Var(w_{1i})$$

$$Var(s_{11}) = \sum_{i=1}^{n} Var(x_i)Var(w_{1i})$$

$$Var(s_{11}) = (nVar(w))(Var(x))$$

If $nVar(w) >> 1$ the variance of s_{1i} will be large and if $nVar(w)$ is small then variance is small. If we add one more hidden layer to our network



$Var(s_{i1} = nVar(w_1)Var(x)$
Using same process as above

$$Var(s_{21}) = \sum_{i=1}^{n} Var(s_{1i})Var(w_{2i})$$

$$Var(s_{21}) = nVar(s_{1i})Var(w_2)$$

$$Var(s_{21}) \propto [nVar(w_2)][nVar(w_1)]Var(x)$$

$$Var(s_{21}) \propto [nVar(w)]^2 Var(x)$$

So in general

$$Var(s_{ki}) = [nVar(w)]^k Var(x)$$

To b ensure that variance doesn't blowup or shrink we want $nVar(w) = 1$. So if we draw weights from unit Gaussian and scale then by $\frac{1}{\sqrt{n}}$ then we have:

$$nVar(w) = nVar(\frac{z}{\sqrt{n}}) = n\frac{1}{n}Var(z) = 1$$

So out of different initialisation strategies we should use unit Gaussian and scale the weights by $\frac{1}{\sqrt{n}}$ or $Var(w) = \frac{1}{n}$

End

References

[1] Lecture 8 of DeepLearning CS7015 by Prof. Mitesh M Khapra, NPTEL