VANISHING AND EXPLODING GRADIENT PROBLEM

Dr. Umarani Jayaraman

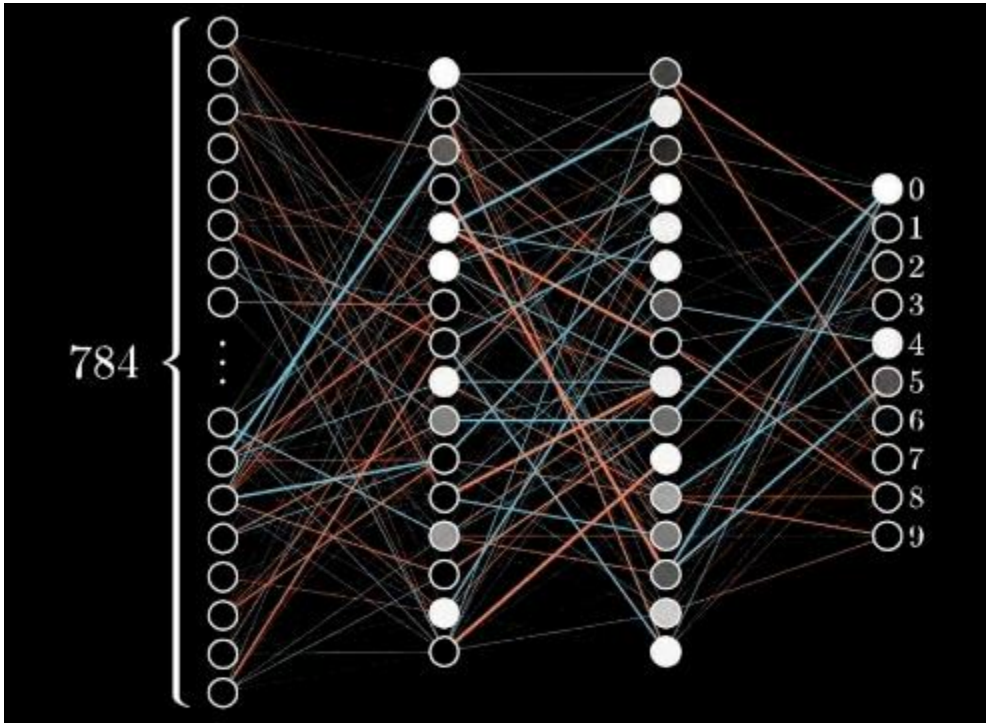
Vanishing and Exploding Gradient Problem

Vanishing and Exploding Gradient Problem

- The problems with training very deep neural network are vanishing and exploding gradients.
- When training a very deep neural network, sometimes derivatives becomes very small (vanishing gradient) or very big (exploding gradient) and this makes training difficult.

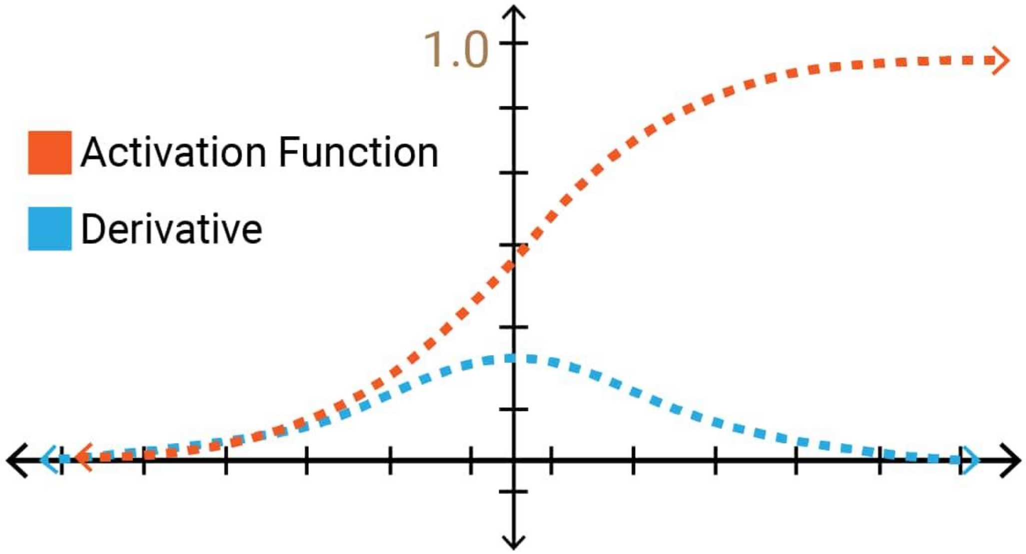
Vanishing and Exploding Gradient Problem

- This problem occurs during training on NN- Back Propagation Learning



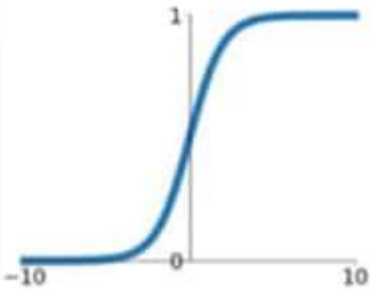
Vanishing Gradient Problem

- In earlier days, most commonly used activation function is sigmoid activation function.



Vanishing Gradient Problem

Function	Equation	Range	Derivative
Sigmoid (Logistic)	$f(x) = \dfrac{1}{1+e^{-x}}$	0,1	$f'(x) = f(x)(1 - f(x))$



Vanishing Gradient Problem

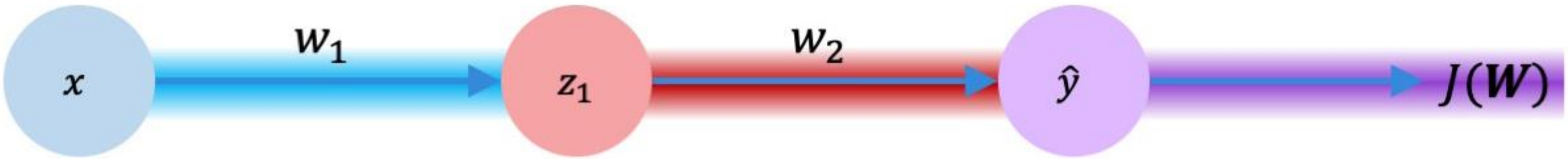
- In sigmoid the values are converted between 0 and 1 and their derivatives lies between 0 and 0.25
- The formula for weight updation is,

$$W_{new} = W_{old} - \eta \, (\partial L / \partial W_{old})$$

$\eta \rightarrow$ Learning rate

$(\partial L/\partial W_{old}) \rightarrow$ Derivation of loss function with respect to the old weight

Vanishing Gradient Problem



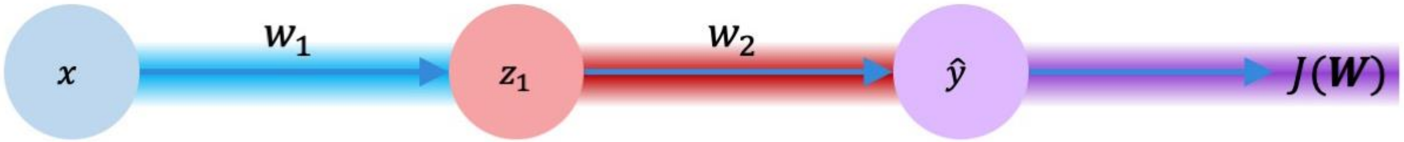
$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

$$W_{new} = W_{old} - \eta\ (\partial L/\partial W_{old})$$

$\eta \rightarrow$ Learning rate

$(\partial L/\partial W_{old}) \rightarrow$ Derivation of loss function with respect to the old weight

Vanishing Gradient Problem



$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

$$= 0.2 * 0.15 * 0.05$$

$$= 0.0015$$

$$W_{new} = W_{old} - \eta\ (\partial L / \partial W_{old})$$

$$= 2.5 - (1)\ 0.0015$$

$$= 2.4985$$

Vanishing Gradient Problem

- As the number of layers in a neural network increases, **the derivative keeps on decreasing.**
- Hence, addition of more layers would lead to almost 0 derivative
- at that time we could see

$$W_{new} \cong W_{old}$$

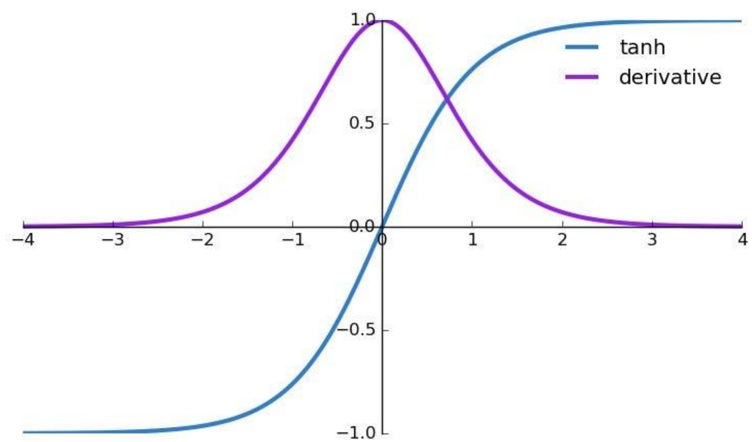
- If new weight is approximately equal to old weight, it actually stops learning. This is called vanishing gradient problem

Vanishing Gradient Problem

- This is the reason why sigmoid is no longer used as activation function in hidden layers.
- The activation function tanh is also not used because it's derivatives lies between 0 and 1. Therefore, it also leads to vanishing gradient problem.

$$a = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\frac{da}{dz} = 1 - a^2$$

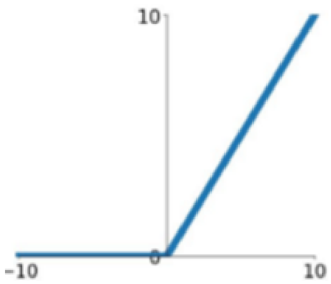


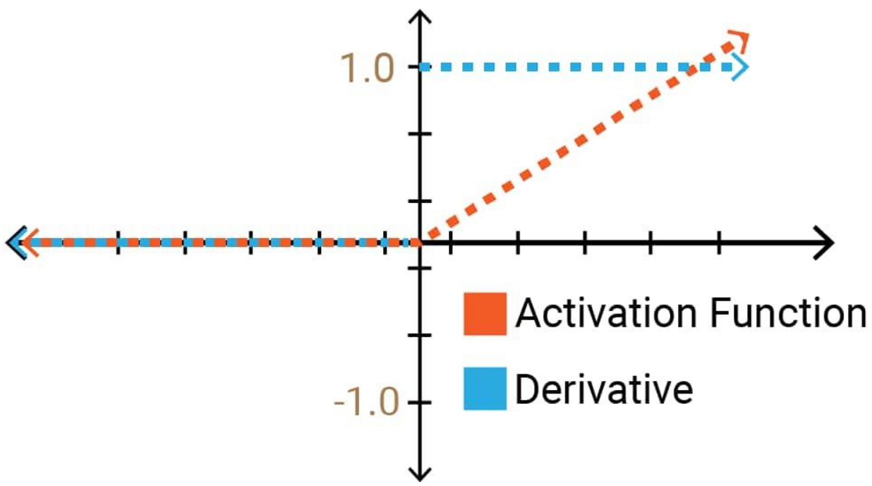
Vanishing Gradient Problem

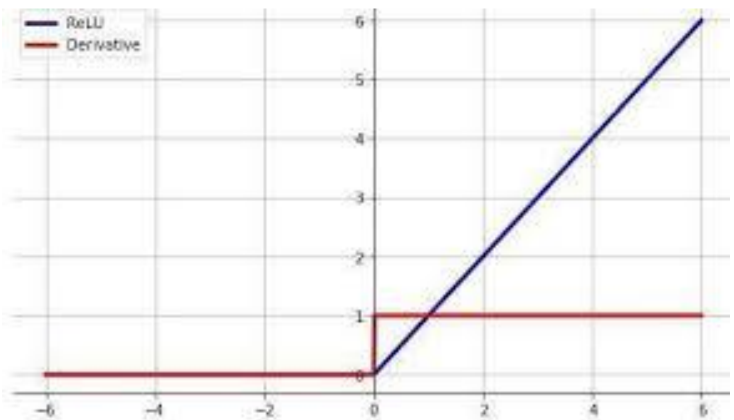
- **Solutions for Vanishing gradient problem:**
- The simplest solution is to use other activation functions, such as ReLU, Leaky ReLU, Pameteric ReLU
- This activation only saturates on one direction and thus are more resilient to the vanishing of gradients.

ReLU Activation function

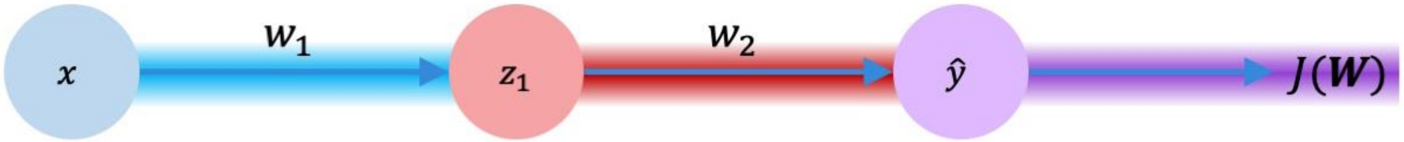
Function	Equation	Range	Derivative
ReLu (Rectified Linear Unit)	$f(x) = \begin{cases} 0\ for\ x < 0 \\ x\ for\ x \geq 0 \end{cases}$	0, +∞	$f'(x) = \begin{cases} 0\ for\ x < 0 \\ 1\ for\ x \geq 0 \end{cases}$







ReLU Activation function



$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

$= 1 * 1 * 1$

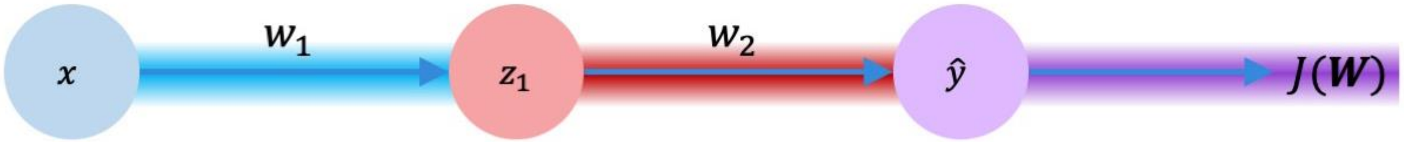
$= 1$

$$W_{new} = W_{old} - \eta \, (\partial L / \partial W_{old})$$

$= 2.5 – (1)\ 1$

$= 1.5$

ReLU Activation function



$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

$= 1 * 1 * 0$

$= 0$ (**dead neuron**)

$$W_{new} = W_{old} - \eta \, (\partial L / \partial W_{old})$$

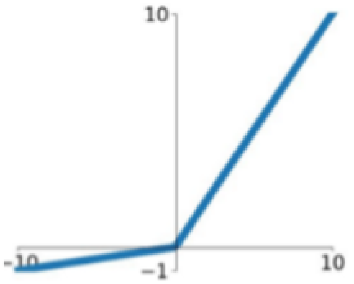
$= 2.5 - (1)\ 0$

$= 2.5$ → $W_{new} \cong W_{old}$

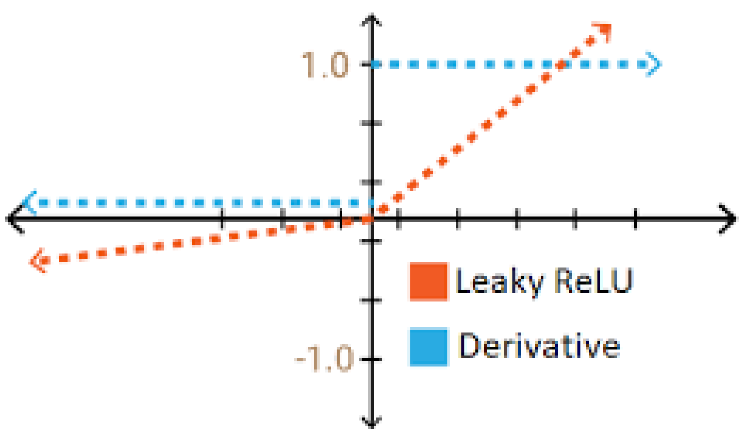
leaky ReLU

- To fix the problem of **dead neuron** leaky ReLU is introduced.

Function	Equation	Range	Derivative
Leaky ReLu	$f(x) = \begin{cases} \alpha x \ for \ x < 0 \\ x \quad for \ x \geq 0 \end{cases}$	-∞, +∞	$f'(x) = \begin{cases} \alpha \ for \ x < 0 \\ 1 \ for \ x \geq 0 \end{cases}$

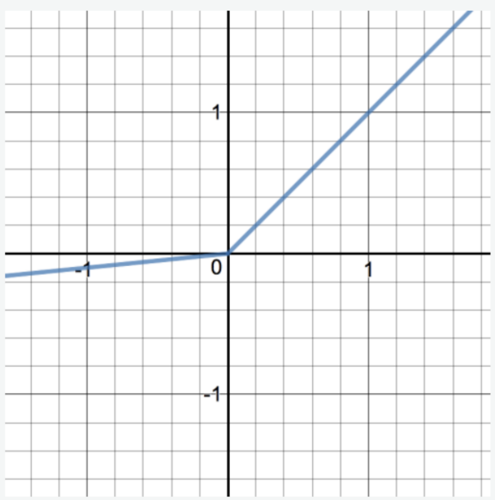
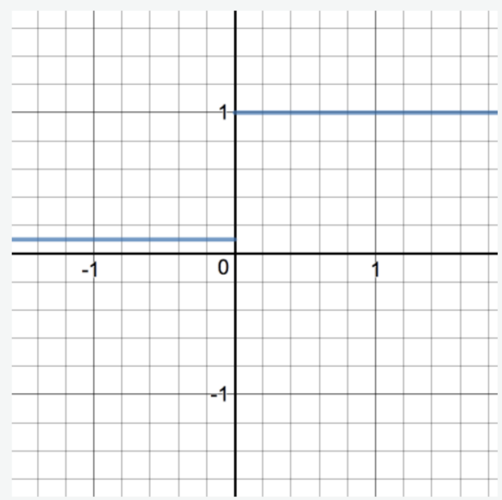


α =0.01

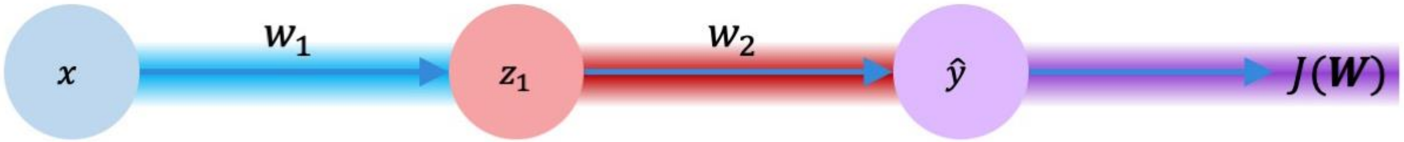


leaky ReLU- derivatives

- The derivatives is no more zero and hence **no dead neurons**

Function	Derivative
$R(z) = \begin{Bmatrix} z & z > 0 \\ \alpha z & z <= 0 \end{Bmatrix}$	$R'(z) = \begin{Bmatrix} 1 & z > 0 \\ \alpha & z < 0 \end{Bmatrix}$
	

Leaky ReLU Activation function



$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

= 0.01 * 1 * 1

= 0.01 (**No dead neuron**)

$$W_{new} = W_{old} - \eta \, (\partial L/\partial W_{old})$$

= 2.5 – (1) 0.01

= 2.49

Vanishing Gradient Problem

- The others solutions are,
  - Use Residual networks (ResNets)
  - Use Batch Normalization
  - Use Multi-level hierarchy
  - Use Long short term memory(LSTM) network
  - Use Faster hardware
  - Genetic algorithms for weight search

Residual neural networks (ResNets)

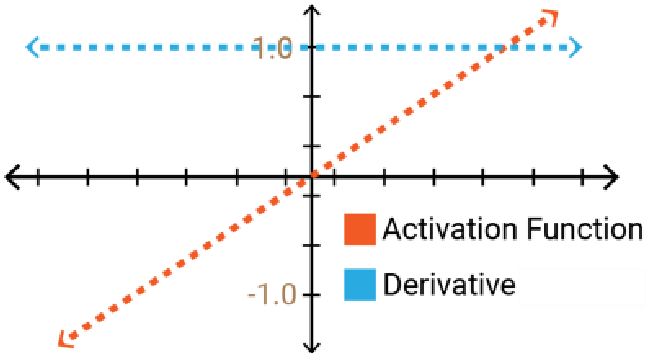
- One of the newest and most effective **ways** to resolve the **vanishing gradient** problem is with residual neural networks, or ResNets (not to be confused with recurrent neural networks).
- It is noted before ResNets that a deeper network would have higher training error than the shallow network.

Residual neural networks (ResNets)

- As this gradient keeps flowing backwards to the initial layers, this value keeps getting multiplied by each local gradient.
- Hence, the gradient becomes smaller and smaller, making the updates to the initial layers very small, or no updation in weights (it stops learning)
- We can solve this problem if the local gradient somehow become 1.

Linear or Identity Activation Function

- **Equation:** f(x) = x
- **Derivative:** f'(x) = 1



Residual neural networks (ResNets)

- How can the local gradient be 1, i.e, the derivative of which function would always be 1?
- The Identity function!

Identity

Some function of y

$$z = f(x,\ y) = x + g(y)$$

$$\boxed{\frac{\delta z}{\delta x} = 1}$$

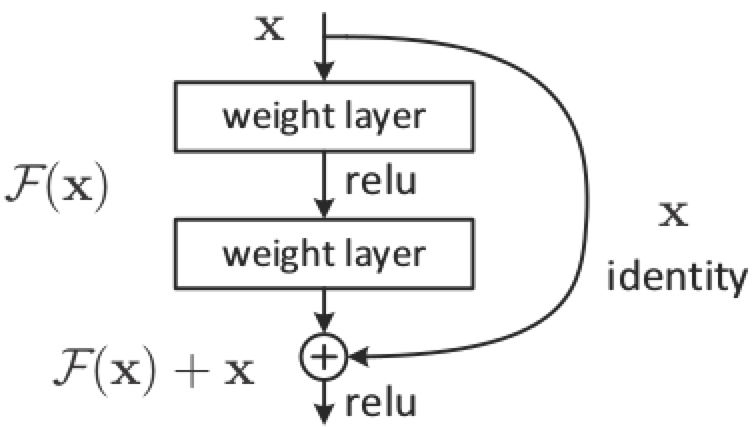
Local Gradent is 1

$$\frac{\delta z}{\delta y} = \frac{\delta g(y)}{\delta y}$$

- As this gradient is back propagated, it does not decrease in value because the local gradient is 1.

Residual neural networks (ResNets)

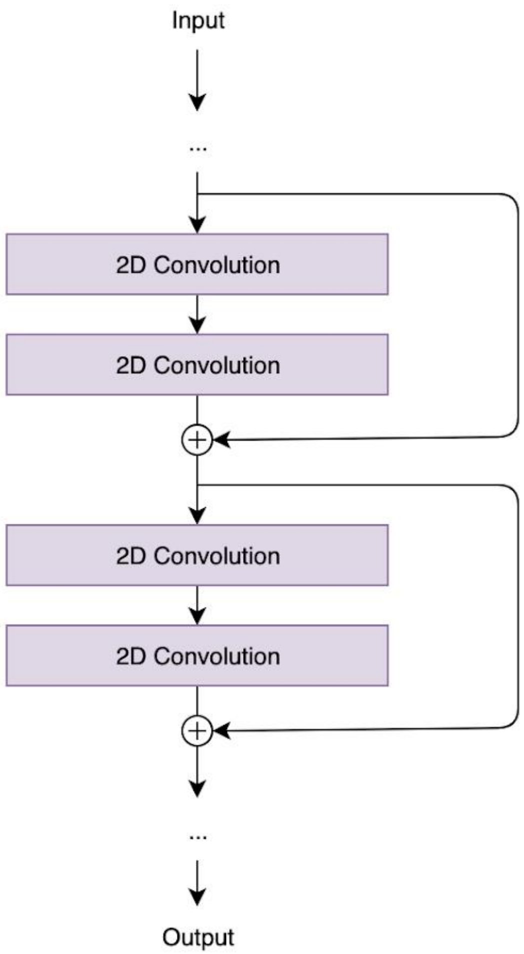
- The residual connection directly adds the value at the beginning of the block, **x**, to the end of the block (F(x)+x).



- This residual connection doesn't go through activation functions that "squashes" the derivatives, resulting in a higher overall derivative of the block.

Residual neural networks (ResNets)

- These *skip connections* act as gradient *superhighways*, allowing the gradient to flow unhindered (without restriction).

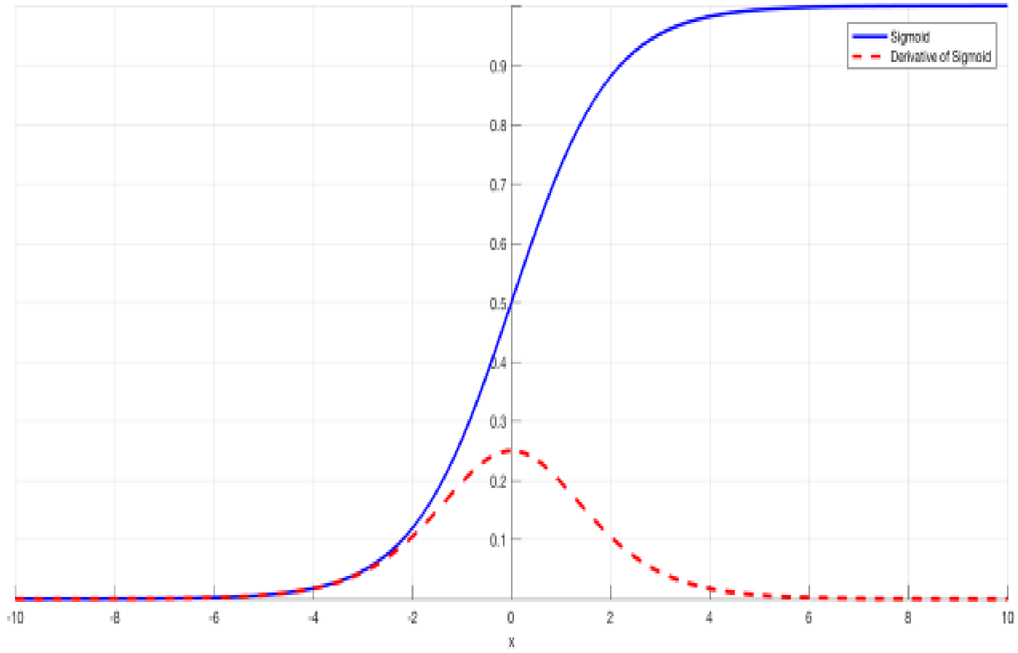


Batch Normalization

- batch normalization layers can also resolve the vanishing gradient problem
- As stated before, the problem arises when a large input space is mapped to a small one, causing the derivatives to disappear.

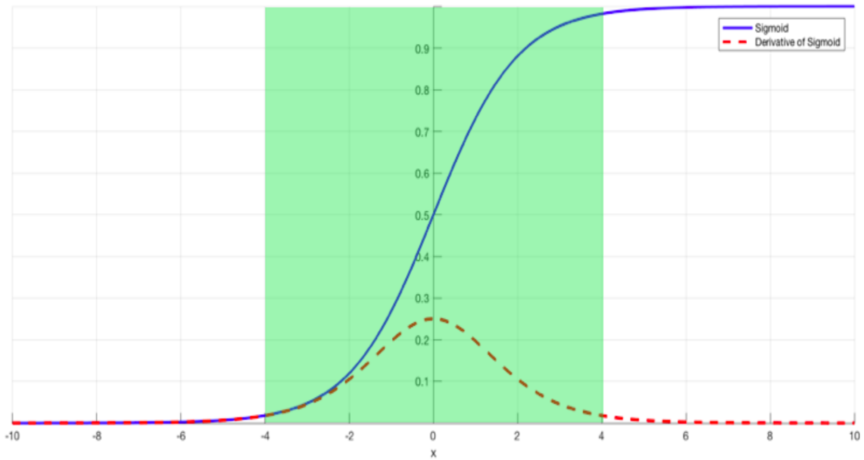
Batch Normalization

- In Image below, this is most clearly seen at when |x| is big.

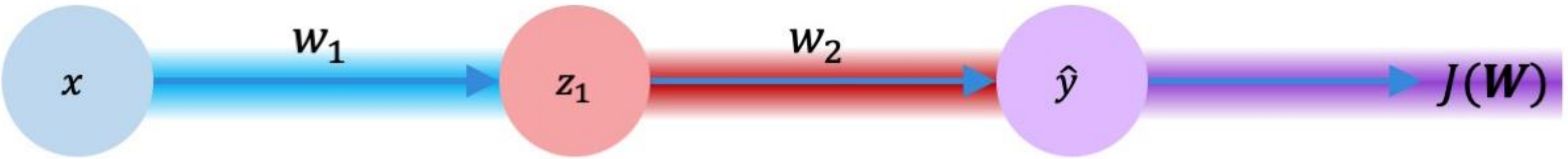


Batch Normalization

- Batch normalization reduces this problem by simply normalizing the input so |x| doesn't reach the outer edges of the sigmoid function.
- It normalizes the input so that most of it falls in the the green region, where the derivative isn't too small.



Exploding Gradient Problem



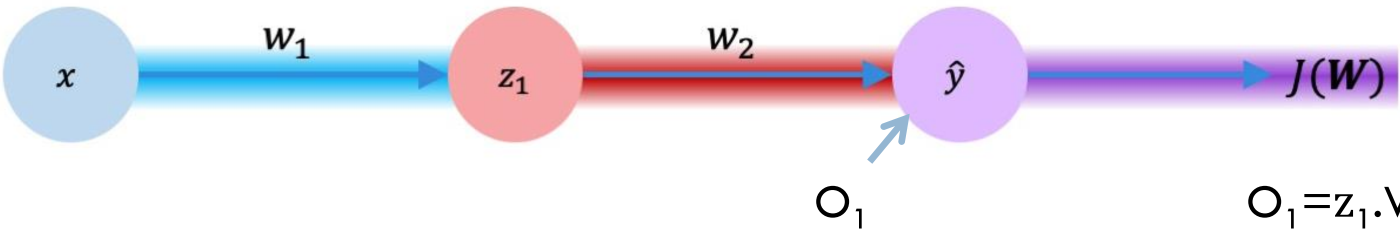
$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

$$W_{new} = W_{old} - \eta \, (\partial L/\partial W_{old})$$

$\eta \rightarrow$ Learning rate

$(\partial L/\partial W_{old}) \rightarrow$ Derivation of loss function with respect to the old weight

Exploding Gradient Problem



$O_1 = z_1.W_2 + b$

$\hat{y} = f(O_1)$

$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

$$\frac{\partial \hat{y}}{\partial z_1} = \frac{\partial f(O_1)}{\partial O_1} * \frac{\partial O_1}{\partial z_1}$$

$$\frac{\partial \hat{y}}{\partial z_1} = 0.25 * \frac{\partial (z_1.W_2 + b)}{\partial z_1}$$

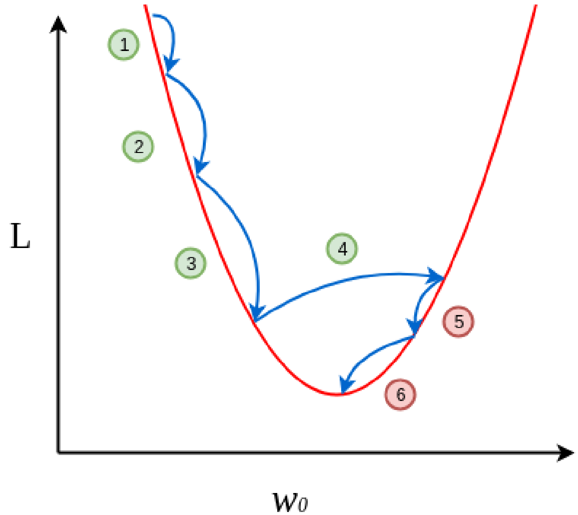
$$\frac{\partial \hat{y}}{\partial z_1} = 0.25 * W_2 = 0.25 * 500 = 125 \quad \text{(when weights are higher)}$$

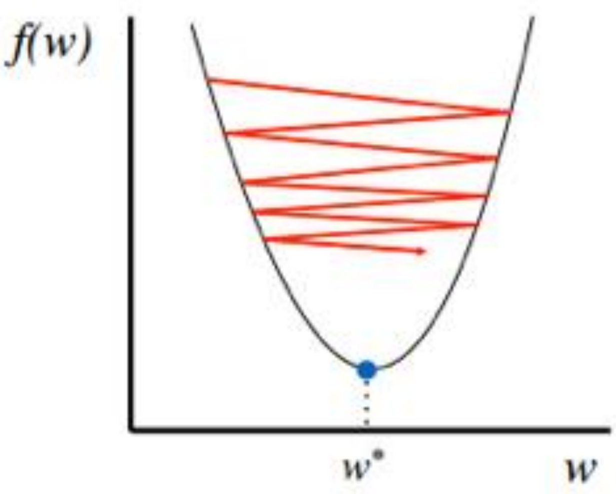
Exploding Gradient Problem

- Exploding problem is not because of sigmoid function
- This problem occurs due to larger weight value
- During initialization of weights if the weights are initialized with larger value, instead of converging, it keep oscillating.
- Hence, we should properly select initial weight vectors

Exploding Gradient Problem

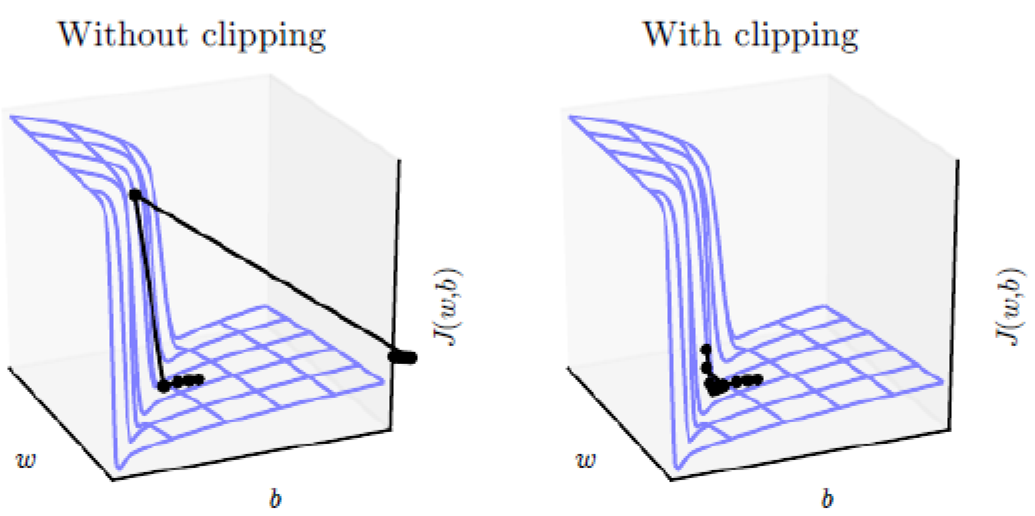
- When gradients explode, the gradients could become NaN (Not a Number) because of the numerical overflow
- We might see irregular oscillations in training cost when we plot the learning curve.





Dealing with Exploding Gradients

- A solution to fix this is to apply **gradient clipping**; which places a predefined threshold on the gradients to prevent it from getting too large, and by doing this it doesn't change the direction of the gradients it only change its length.



$$\text{if } ||g|| > \quad threshold$$

$$g \leftarrow \frac{threshold \times g}{||g||}$$

where: g *is the gradient and*

$||g||$ *is the norm of the gradient*

Note: Step function

The derivative of a step function, also known as the Heaviside step function, is a generalized function called the Dirac delta function.

The Heaviside step function is defined as:

$$H(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases}$$

Its derivative, which is not a conventional function but a distribution, is represented by the Dirac delta function:

$$H'(x) = \delta(x)$$

The Dirac delta function is defined such that it is zero everywhere except at $x = 0$, where it is infinitely large in such a way that its integral over any finite interval containing $x = 0$ is equal to 1. Mathematically, the Dirac delta function is often represented as:

$$\int_{-\infty}^{\infty} \delta(x)\, dx = 1$$

Note that dealing with derivatives of generalized functions like the Dirac delta function requires the framework of distribution theory.

Note: Step function

The derivative of a threshold function, sometimes called a hard threshold function, depends on the specific definition of the function.

If we define the threshold function $f(x)$ as:

$$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases}$$

then its derivative $f'(x)$ would be a Dirac delta function:

$$f'(x) = \delta(x)$$

This is because the threshold function has a jump discontinuity at $x = 0$, and its derivative is non-zero only at $x = 0$, which is the defining property of the Dirac delta function.

Note: RelU function

The ReLU (Rectified Linear Unit) function is defined as:

$$f(x) = \max(0, x)$$

Its derivative $f'(x)$ is defined piecewise:

$$f'(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x > 0 \\ \text{undefined}, & \text{if } x = 0 \end{cases}$$

At points where $x < 0$, the derivative is 0, as the function is flat and not changing. At points where $x > 0$, the derivative is 1, as the function is a straight line with a slope of 1. At $x = 0$, the function is not differentiable, so the derivative is undefined from the left and right.

However, in practice, for computational purposes, the derivative at $x = 0$ is often taken as 0 or 1 depending on the implementation. Some implementations use subgradients and define the derivative at $x = 0$ as the set of values between 0 and 1.

Linear function definition

Alternatively, a linear function can also be defined as a function that satisfies two properties:

1. **Additivity:** $f(x+y)=f(x)+f(y)$ for all x,y in the function's domain.
2. **Homogeneity:** $f(ax)=af(x)$ for all x in the function's domain and any scalar a.

These properties ensure that the function's behavior is consistent and predictable across its domain. Graphically, linear functions produce straight lines when plotted on a Cartesian plane.

Linear functions have several properties that distinguish them from other types of functions. Here are some key properties of linear functions:

1. **Linearity:** The defining property of a linear function is that it is linear, meaning it satisfies the linearity condition: $f(ax+by)=af(x)+bf(y)$, where a and b are constants, and x and y are variables. This property essentially says that the function's output is directly proportional to its inputs.

Non-linear function definition

The mathematical definition of a nonlinear function is a function that does not satisfy the properties of linearity. Specifically, a function $f(x)$ is considered nonlinear if it cannot be expressed in the form of a linear function, i.e., if it cannot be written as:

$$f(x) = mx + b$$

for some constants m and b, where x is the variable.

Alternatively, a nonlinear function can be defined as a function that does not satisfy the following properties of linearity:

1. **Additivity:** $f(x+y) = f(x) + f(y)$ for all x, y in the function's domain.
2. **Homogeneity:** $f(ax) = af(x)$ for all x in the function's domain and any scalar a.

Source: Activation function

- https://inblog.in/ACTIVATION-FUNCTION-BREAKTHROUGH-VOyvxhTELU

Source: Vanishing gradient problem

- https://www.mygreatlearning.com/blog/the-vanishing-gradient-problem/
- https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484
- https://medium.com/analytics-vidhya/vanishing-and-exploding-gradient-problems-c94087c2e911

THANK YOU