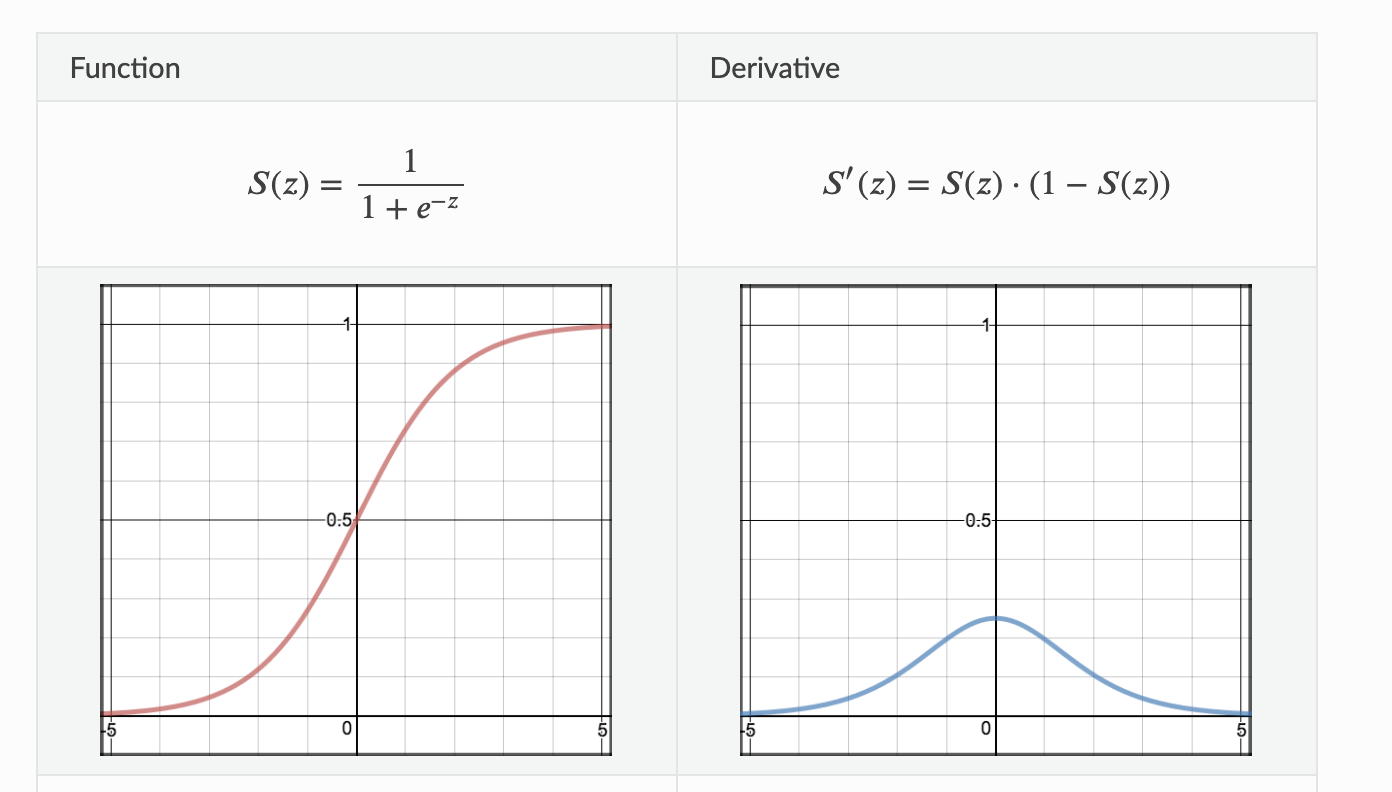
**Activation Functions: Sigmoid, Tanh, ReLU, Leaky ReLU, Softmax**

<https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5>

Generally, neural networks use **non-linear activation functions**, which can help the network learn complex data, compute and learn almost any function representing a question, and provide accurate predictions.They **allow back-propagation** because they have a derivative function which is related to the inputs.

1. **Sigmoid Activation Function:**

Sigmoid Activation function is very simple which takes a real value as input and gives probability that ‘s always between 0 or 1. It looks like ‘S’ shape.

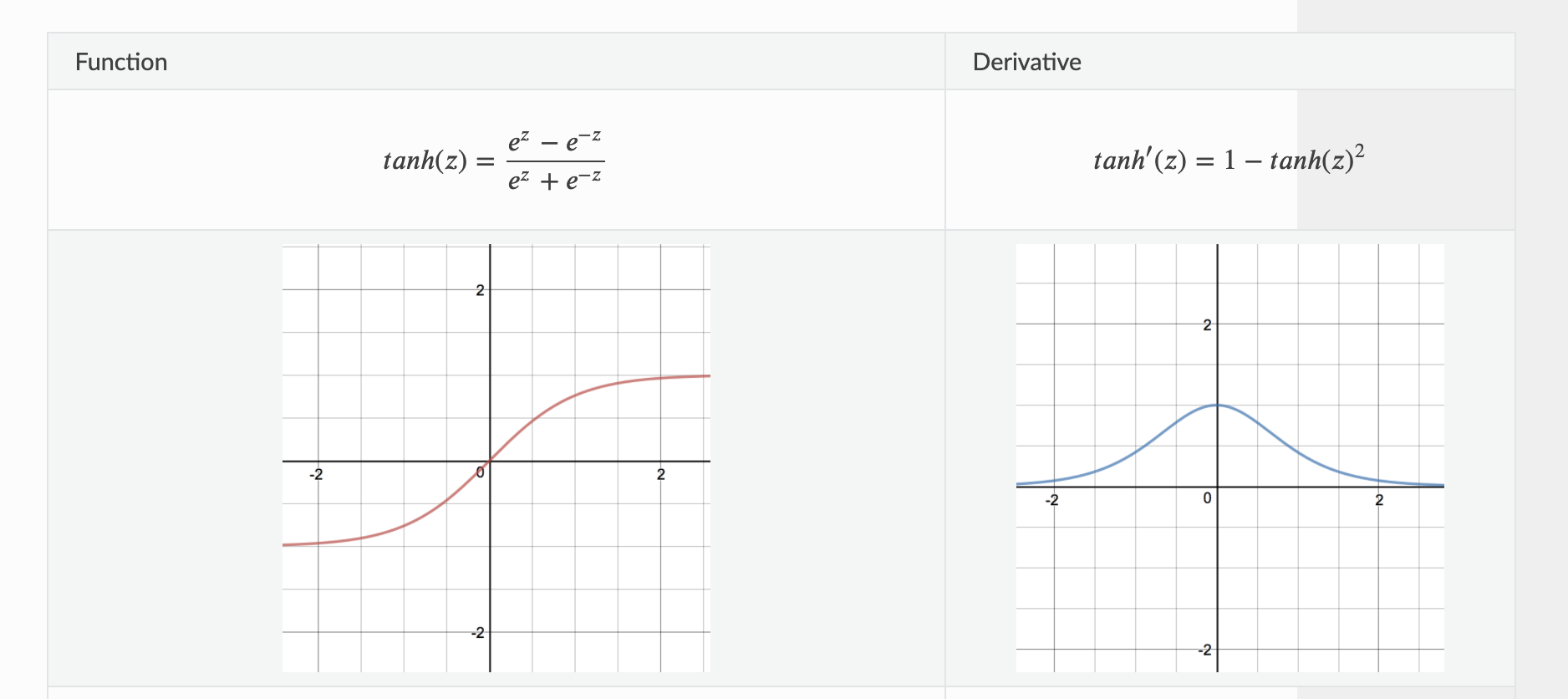


Slika S: sigmoid function and it's derivative

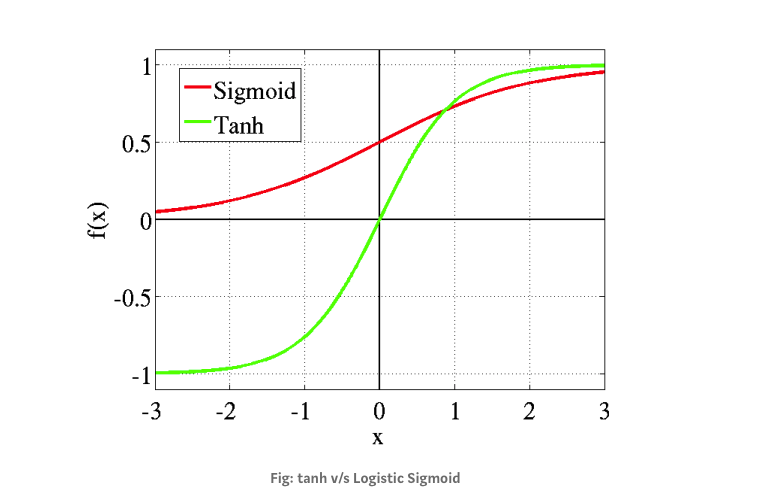
It’s non-linear, continuously differentiable, monotonic, and has a fixed output range. Main advantage is simple and good for classifier. But Big disadvantage of the function is that it It gives rise to a problem of “vanishing gradients” because Its output isn’t zero centered. It makes the gradient updates go too far in different directions. 0 < output < 1, and it makes optimization harder. That takes very high computational time in hidden layer of neural network

**2. Tanh or Hyperbolic tangent:**

Tanh help to solve non zero centered problem of sigmoid function. Tanh squashes a real-valued number to the range [-1, 1]. It’s non-linear too.

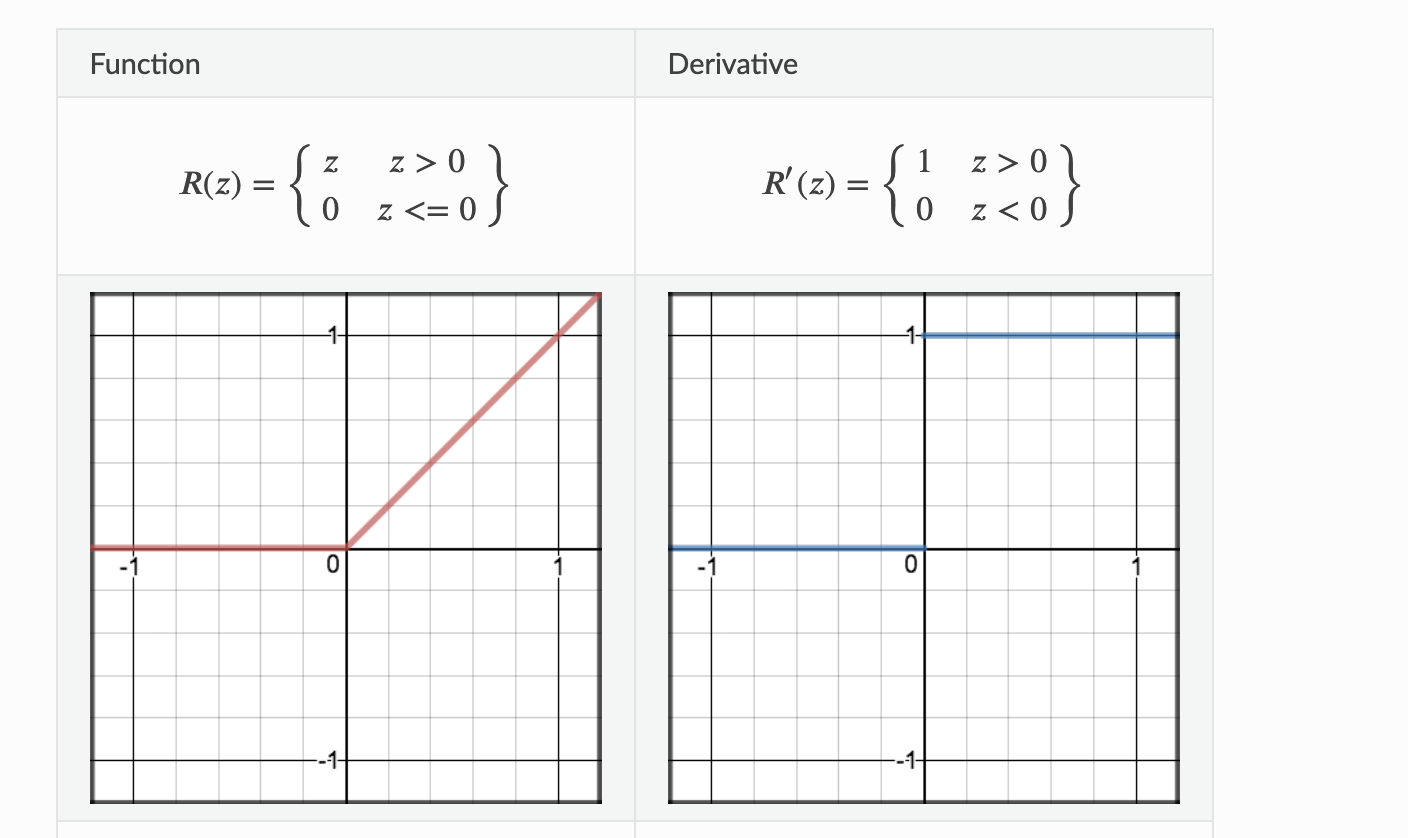


Derivative function give us almost same as sigmoid’s derivative function.It solve sigmoid’s drawback but it still can’t remove the vanishing gradient problem completely.When we compare tanh activation function with sighmoid , this picture give you clear idea.



**3. ReLU (Rectified Linear Unit):**

This is most popular activation function which is used in hidden layer of NN.The formula is deceptively simple: 𝑚𝑎𝑥(0,𝑧). Despite its name and appearance, it’s not linear and provides the same benefits as Sigmoid but with better performance.

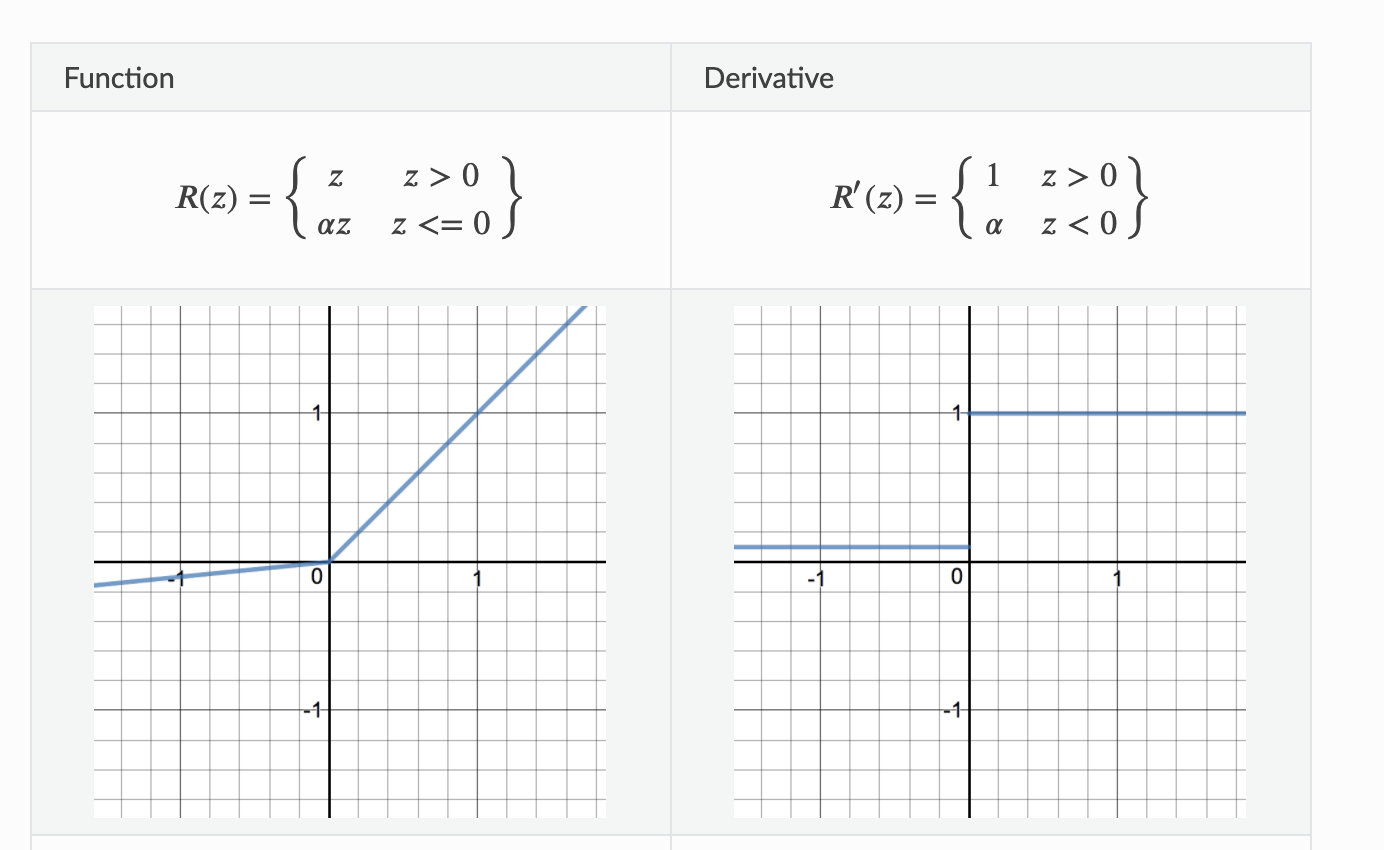


Slika : ReLU activation function and it's derivative

It’s main advantage is that it avoids and rectifies vanishing gradient problem and less computationally expensive than tanh and sigmoid. But it has also some draw back . Sometime some gradients can be fragile during training and can die. That leads to dead neurons.In another words, for activations in the region (x<0) of ReLu, gradient will be 0 because of which the weights will not get adjusted during descent. That means, those neurons which go into that state will stop responding to variations in error/ input ( simply because gradient is 0, nothing changes ). So We should be very carefully to choose activation function , and activation function should be as per business requirement.

**4. Leaky ReLU**

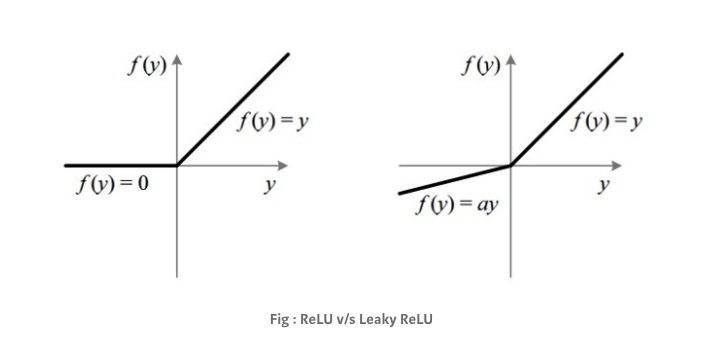
It prevents dying ReLU problem.T his variation of ReLU has a small positive slope in the negative area, so it does enable back-propagation, even for negative input values



Leaky ReLU does not provide consistent predictions for negative input values. During the front propagation if the learning rate is set very high it will overshoot killing the neuron.

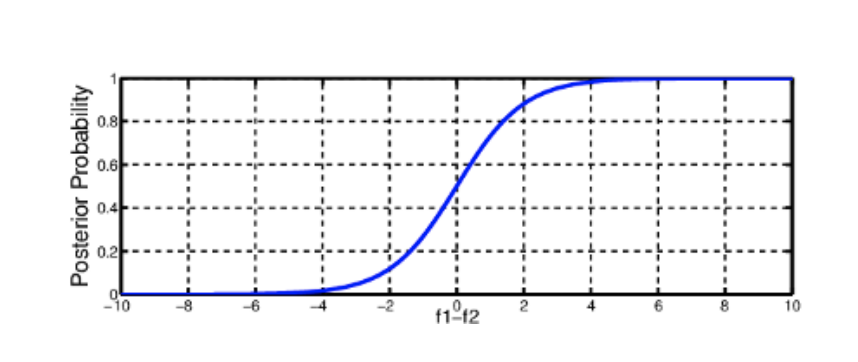
The idea of leaky ReLU can be extended even further. Instead of multiplying x with a constant term we can multiply it with a hyper-parameter which seems to work better the leaky ReLU. This extension to leaky ReLU is known as **Parametric ReLU**.

While we compare Leaky-ReLU with ReLU, then It shows clear concept of difference between them.

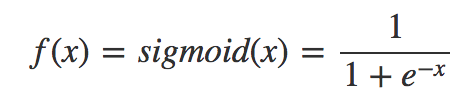


**5. Softmax**

Generally, we use the function at last layer of neural network which calculates the probabilities distribution of the event over ’n’ different events. The main advantage of the function is able to handle multiple classes

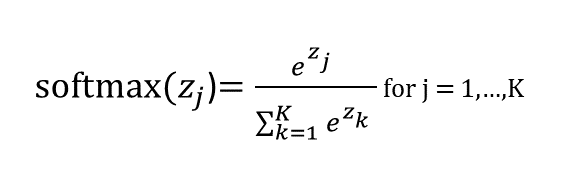


when we compare the sigmoid and softmax activation functions , they produce different results.



**Sigmoid input values**: -0.5, 1.2, -0.1, 2.4

**Sigmoid output values**: 0.37, 0.77, 0.48, 0.91



**SoftMax input values**: -0.5, 1.2, -0.1, 2.4

**SoftMaxo utput values**: 0.04, 0.21, 0.05, 0.70