Other types of activation functions and when to use them?

If we didn’t use the activation functions, the linearity will be shown so, the problem will be that linearity couldn’t handle the relations’ complexties.

Types:

Binary Step Function

It is a threshold based classifier, if the value Y is above a given threshold value then activate the neuron else leave it deactivated.

It can be used while creating a binary classifier. When we simply need to say yes or no for a single class, step function would be the best choice.

Linear Function

The input x, will be transformed following its a\*x. This can be applied to various neurons and multiple neurons can be activated at the same time.

Sigmoid

the output is non linear as well. The function ranges from 0–1 having an S shape. A small change in x would also bring about large changes in the value of Y. So the function essentially tries to push the Y values towards the extremes. This is a very desirable quality when we’re trying to classify the values to a particular class.

Tanh

All other properties are the same as that of the sigmoid function. It is continuous and differentiable at all points. The function is non linear so we can easily back-propagate the errors.

ReLU

 does not activate all the neurons at the same time. If you look at the ReLU function when the input is negative it will convert it to zero and the neuron does not get activated. This means that at a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

But, the magic is still not stop working. A smooth approximation to the rectifier is the analytic function

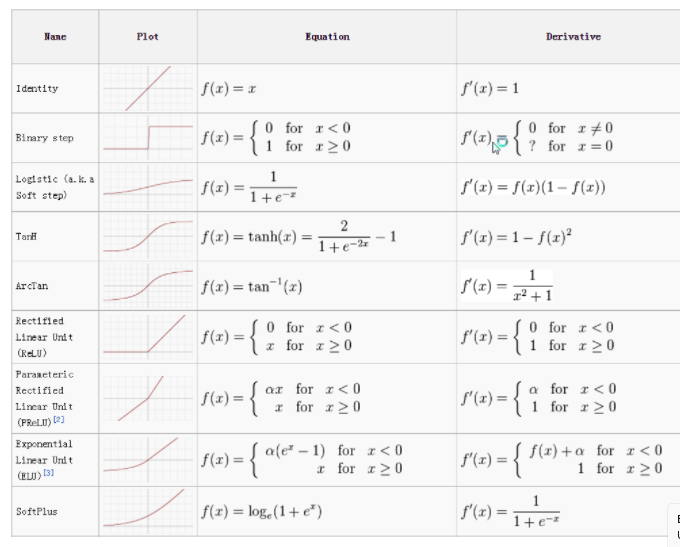
It is less computationally because it involves simpler mathematical operations. That is a good point to consider when we are designing deep neural nets.

ReLU function should only be used in the hidden layers. At current time, ReLu works most of the time as a general approximator

Leaky ReLU

Leaky ReLU is an improved version of the ReLU function. As we saw above the gradient of ReLU is 0 when x<0, which made the neurons die. Leaky ReLU is defined to address this problem. Instead of defining the ReLU function as 0 for x less than 0, we define it as a small linear component of x.

* If we encounter a case of dead neurons in our networks the leaky ReLU function is the best choice



You can create your own activation function

Ref: <https://phuctrt.medium.com>