VIT-Vellore, SCOPE

CSE6037 - Deep Learning and its Applications

SUTHAR MANAN BHARATKUMAR 20MAI0016

Assessment 2

Problem 1: Demonstrate Activation Functions used in Neural Networks

Activation Functions for Neural Networks

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated ("fired") or not, based on whether each neuron's input is relevant for the model's prediction.
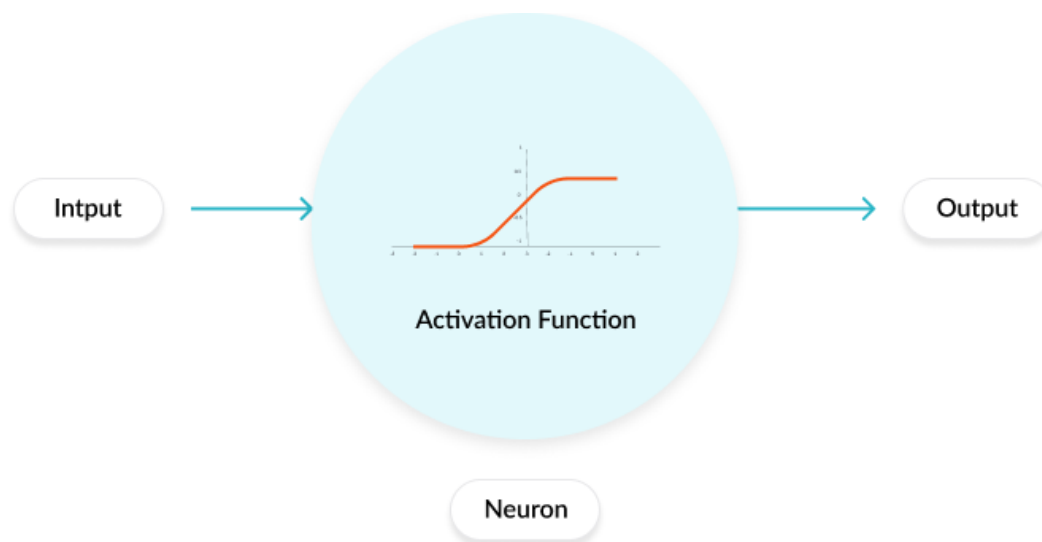
The Activation Functions can be basically divided into 2 types-

1. Linear Activation Function
2. Non-linear Activation Functions

There are several activations which are generally preffered, which we will see here...

1. Step function

The first thing that comes to our mind when we have an activation function would be a threshold based classifier i.e. whether or not the neuron should be activated based on the value from the linear transformation. In other words, if the input to the activation function is greater than a threshold, then the neuron is activated, else it is deactivated, i.e. its output is not considered for the next hidden layer.

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{For } x => 0 \end{cases}$$
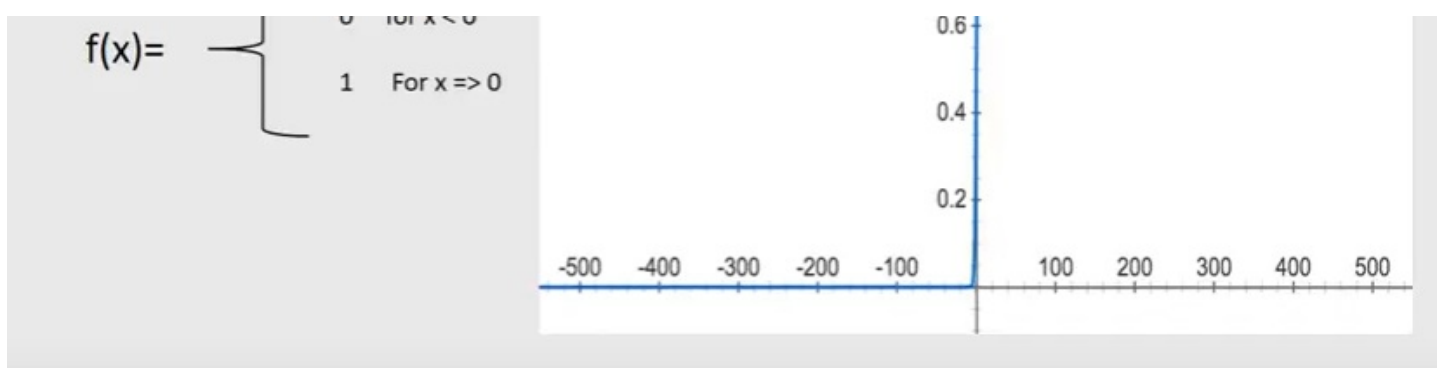
In [24]:

```python
import numpy as np
demonp = np.random.uniform(-5, 5, (5))
demonp
```

Out[24]:

```
array([ 3.85863755,  1.67161791, -0.97842446, -1.92102055, -4.53216778])
```

In [31]:

```python
def binary_step(x):
    if x<0:
        return 0
    else:
        return 1
step=[]
for i in demonp:
    step.append(binary_step(i))
step
```

Out[31]:

```
[1, 1, 0, 0, 0]
```

2. Sigmoid

The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output.Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice. The function is differentiable.That means, we can find the slope of the sigmoid curve at any two points. The function is monotonic but function's derivative is not. The logistic sigmoid function can cause a neural network to get stuck at the training time.

$$\phi(z) = \frac{1}{1 + e^{-z}}$$
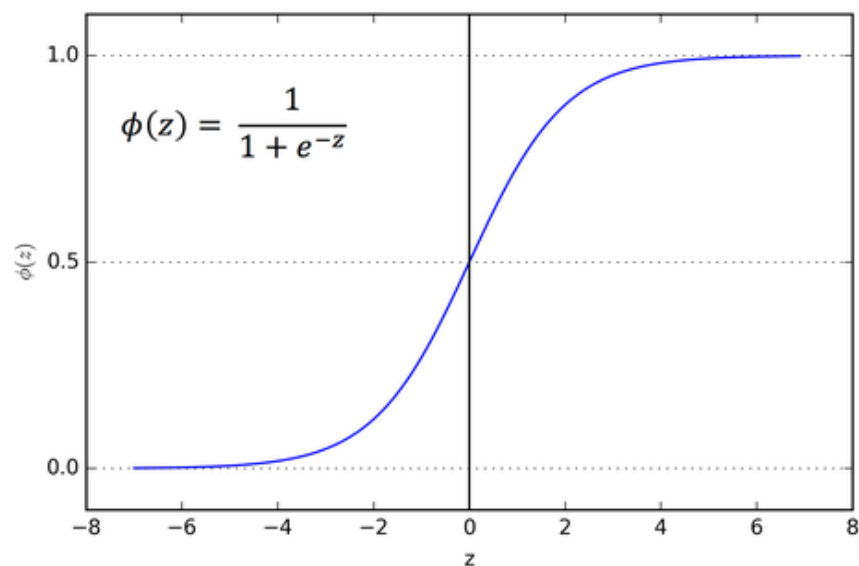
In [25]:

```python
def sigmoid(x):
```

```
    return 1/(1+np.exp(-x))
sgmd = sigmoid(demonp)
sgmd
```

Out[25]:

```
array([0.97933915, 0.84179141, 0.27320452, 0.1277478 , 0.01064284])
```

3. Tanh or hyperbolic tangent

tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped). The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph. The function is differentiable. The function is monotonic while its derivative is not monotonic. The tanh function is mainly used classification between two classes. Both tanh and logistic sigmoid activation functions are used in feed-forward nets.
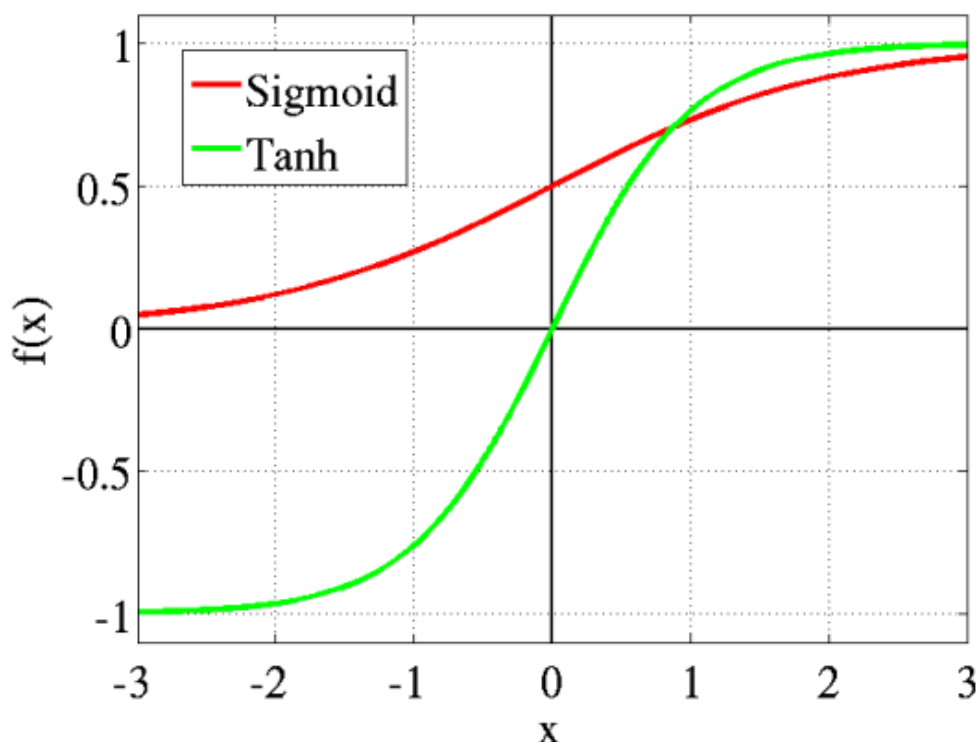
In [26]:

```
import math
def tanhactfun(x):
    return (np.exp(x) - np.exp(-x))/(np.exp(x) + np.exp(-x))
tan = tanhactfun(demonp)
tan
```

Out[26]:

```
array([ 0.99911025,  0.93176527, -0.75238306, -0.95800131, -0.99976859])
```

4. ReLU (Rectified Linear Unit)

The ReLU is the most used activation function in the world right now.Since, it is used in almost all the convolutional neural networks or deep learning. As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

Range: 0 to infinity

The function and its derivative both are monotonic. But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.
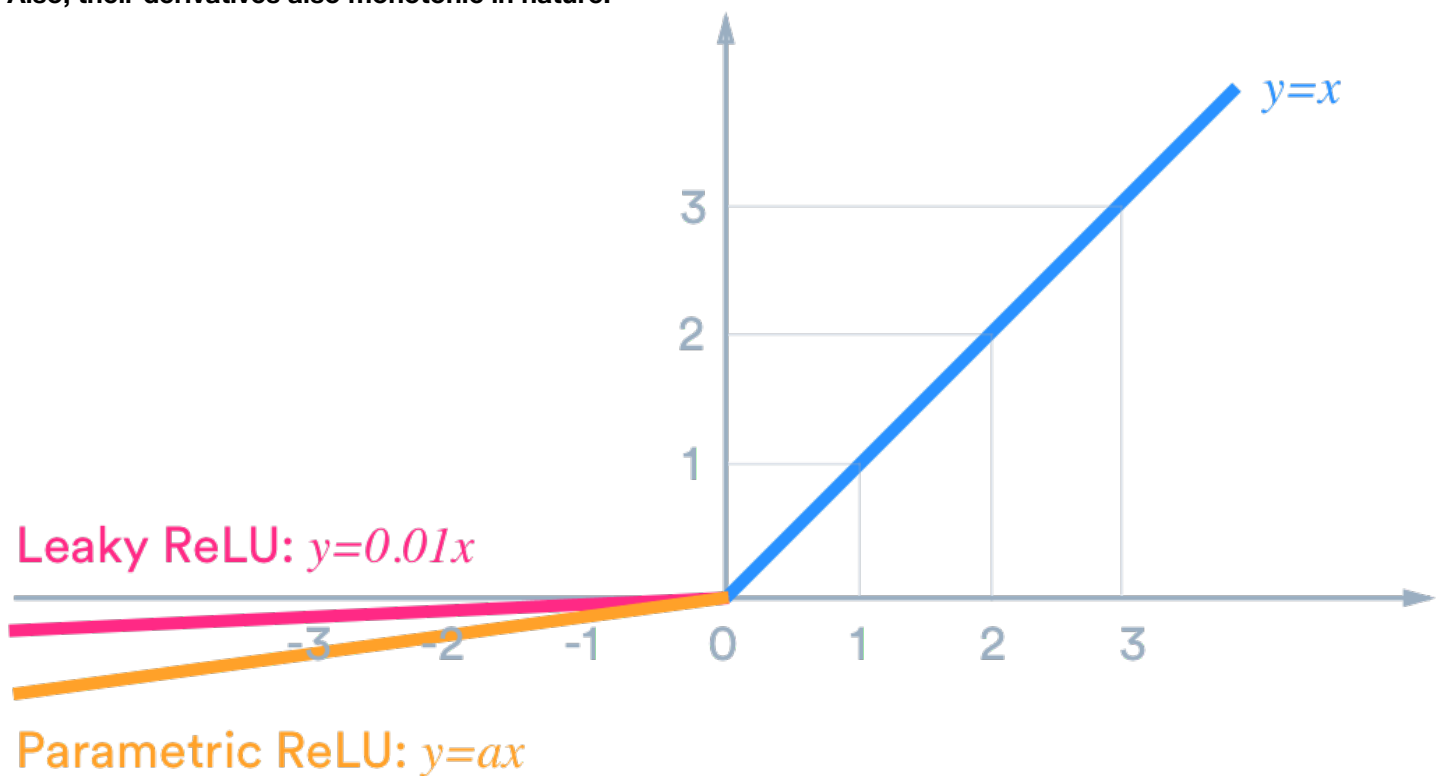
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$R(z) = max(0, \ z)$$
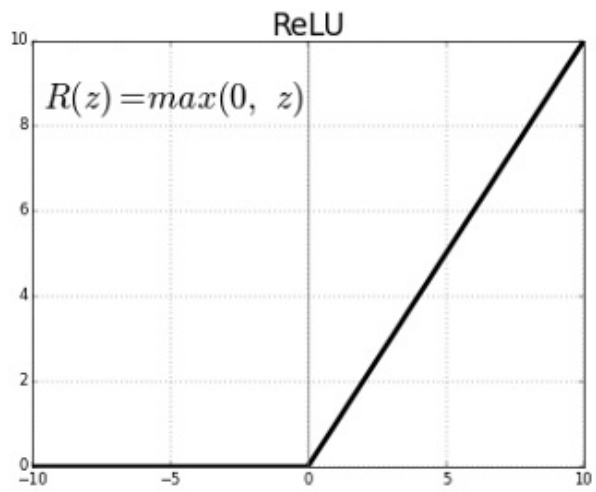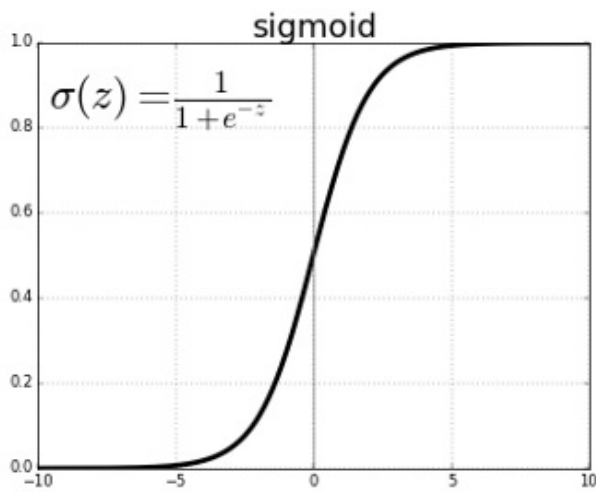
```
def reluactfun(x):
    return (max(0,x))
reluact=[]
for i in demonp:
    reluact.append(reluactfun(i))
reluact
```

```
[3.858637554028192, 1.6716179099478623, 0, 0, 0]
```

5. Leaky ReLU

It is an attempt to solve the dying ReLU problem. The leak helps to increase the range of the ReLU function. Usually, the value of a is 0.01 or so. When a is not 0.01 then it is called Randomized ReLU. Therefore the range of the Leaky ReLU is (-infinity to infinity). Both Leaky and Randomized ReLU functions are monotonic in nature. Also, their derivatives also monotonic in nature.

Leaky ReLU: $y=0.01x$

Parametric ReLU: $y=ax$

```
def reluactfun(x):
    return (max(0.1*x, x))
reluact=[]
for i in demonp:
    reluact.append(reluactfun(i))
```

```
reluact
```

```
[3.858637554028192,
 1.6716179099478623,
 -0.09784244587912419,
 -0.19210205498147817,
 -0.45321677767168994]
```

6. ELU (Exponential Linear Unit)

Exponential Linear Unit or ELU for short is also a variant of Rectiufied Linear Unit (ReLU) that modifies the slope of the negative part of the function. Unlike the leaky relu and parametric ReLU functions, instead of a straight line, ELU uses a log curve for defning the negatice values.
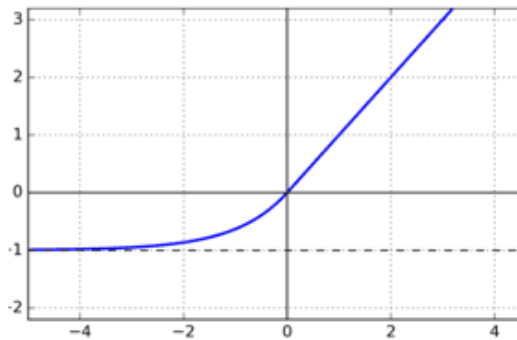
```
def elu_function(x, a):
    return max(a*(np.exp(x)-1),x)
elu=[]
a = 0.55
for i in demonp:
    elu.append(elu_function(i,a))
elu
```

```
[25.520399534340587,
 2.376423149489315,
 -0.34325341433817474,
 -0.46944841969106427,
 -0.5440834677924558]
```

7. Swish

Swish is a lesser known activation function which was discovered by researchers at Google. Swish is as computationally efficient as ReLU and shows better performance than ReLU on deeper models. The values for swish ranges from negative infinity to infinity.
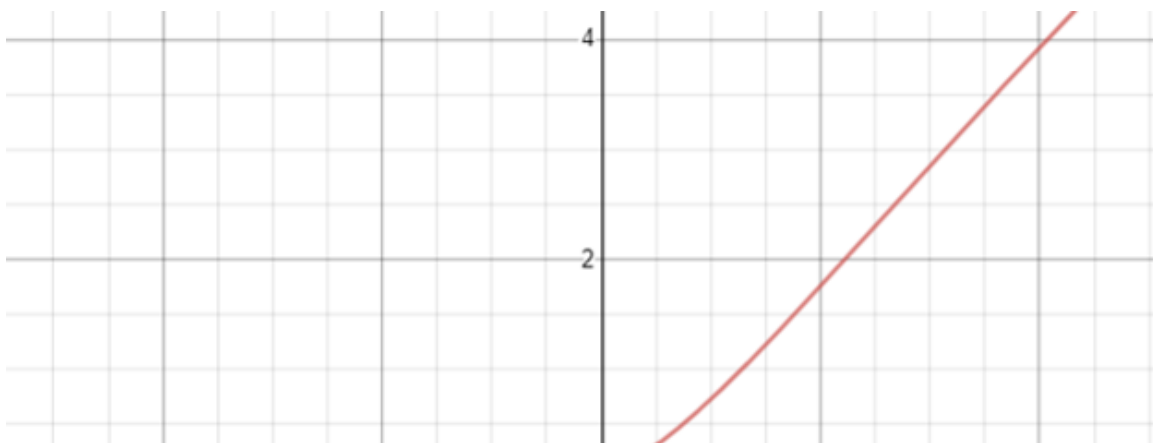
```
def swish_function(x):
    return x/(1-np.exp(-x))
swish=[]
for i in demonp:
    swish.append(swish_function(i))
swish
```

Out[41]:

```
[3.9417965442258094,
 2.0584975995641237,
 0.589319458258369,
 0.32962352114286336,
 0.049284196650446005]
```

8. Softmax

Softmax function is often described as a combination of multiple sigmoids. We know that sigmoid returns values between 0 and 1, which can be treated as probabilities of a data point belonging to a particular class. Thus sigmoid is widely used for binary classification problems. The softmax function can be used for multiclass classification problems.
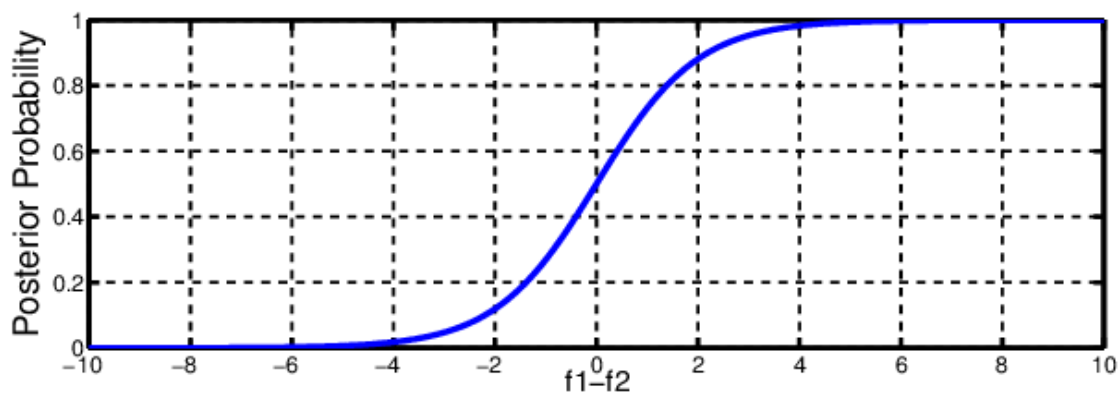
In [42]:

```
def softmax_function(x):
    z = np.exp(x)
    z_ = z/z.sum()
    return z_
softm = softmax_function(demonp)
softm
```

Out[42]:

```
array([8.90077373e-01, 9.99118953e-02, 7.05859753e-03, 2.75013580e-03,
       2.01998110e-04])
```