Tenserflow Basics

TensorFlow provides a variety of math functions including: Basic arithmetic operators and trigonometric functions. Special math functions (like: tf.math.igamma and tf.math.zeta) Complex number functions (like: tf.math.imag and tf.math.angle) Reductions and scans (like: tf.math.reduce_mean and tf.math.cumsum)

import tensorflow as tf import numpy as np

In [5]:
```
x=tf.constant([0.0,2.0,50.0,100.0])
y=tf.math.sigmoid(x)
```

In [8]:
```
y.numpy()
```

Out[8]:
```
array([0.5      , 0.8807971, 1.       , 1.       ], dtype=float32)
```

In [13]:
```
x=tf.constant([[3,5],[5,6]])
y=tf.constant([[7,1],[2,9]])
z=tf.math.accumulate_n([x,y,y])
```

In [14]:
```
z.numpy()
```

Out[14]:
```
array([[17,  7],
       [ 9, 24]])
```

In [15]:
```
x=tf.constant([3,5,6])
y=tf.constant([7,2,9])
tf.divide(x,y).numpy()
```

Out[15]:
```
array([0.42857143, 2.5       , 0.66666667])
```

In [18]:
```
x=tf.constant(45.0)
tf.math.cos(x).numpy()
```

Out[18]:
```
0.52532196
```

In [19]:
```
x=tf.constant(45.0)
tf.math.sin(x).numpy()
```

Out[19]:
```
0.8509035
```

In [21]:
```
x=tf.constant(0.0)
tf.math.cos(x).numpy()
```

Out[21]:
```
1.0
```

In [22]:
```
x=tf.constant(0.0)
tf.math.sin(x).numpy()
```

Out[22]:
```
0.0
```

In [23]:
```
x=tf.constant([3.0,20.0,50.0,100.0])
tf.cumsum(x).numpy()
```

Out[23]:
```
array([  3.,  23.,  73., 173.], dtype=float32)
```

In [25]:
```
x=tf.constant([[3.0,20.0,50.0,100.0],[3.0,20.0,5.0,10.0]])
tf.cumsum(x).numpy()
```

Out[25]:
```
array([[  3.,  20.,  50., 100.],
       [  6.,  40.,  55., 110.]], dtype=float32)
```

In [31]:
```
x=tf.constant(["A","b"])
y=tf.constant("A")
tf.math.equal(y,x).numpy()
```

Out[31]:
```
array([ True, False])
```

RANDOM NUMBER GENERATOR

In [45]:
```
g=tf.random.Generator.from_seed(1234)
g.normal(shape=(2,3)).numpy()
```

Out[45]:
```
array([[ 0.9356609 ,  1.0854305 , -0.93788373],
       [-0.5061547 ,  1.3169702 ,  0.7137579 ]], dtype=float32)
```

In [46]:
```
g = tf.random.Generator.from_non_deterministic_state()
g.normal(shape=(2, 3))
```

Out[46]:
```
<tf.Tensor: shape=(2, 3), dtype=float32, numpy=
array([[ 0.38565588,  0.04238863,  0.43340236],
       [ 0.65298355, -0.29912132,  0.7654733 ]], dtype=float32)>
```

In [53]:
```
samples=tf.random.categorical(tf.math.log([[0.3,0.5,0.2]]),5)
samples.numpy()
```

Out[53]:
```
array([[0, 1, 2, 1, 1]], dtype=int64)
```

In [55]:
```
t1=([1,2,3],[4,5,6])
t2=([3,2,1],[0,7,8])
tf.concat([t1,t2],1).numpy()
```

Out[55]:
```
array([[1, 2, 3, 3, 2, 1],
       [4, 5, 6, 0, 7, 8]])
```

In [56]:
```
t1=([1,2,3],[4,5,6])
t2=([3,2,1],[0,7,8])
tf.concat([t1,t2],0).numpy()
```

Out[56]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [3, 2, 1],
       [0, 7, 8]])
```

In [58]:
```
tf.fill([2,3],1)
```

Out[58]:
```
<tf.Tensor: shape=(2, 3), dtype=int32, numpy=
array([[1, 1, 1],
       [1, 1, 1]])>
```

In [63]:
```
tf.eye(10).numpy()
```

Out[63]:
```
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]], dtype=float32)
```

In [65]:
```
tf.linspace(10.0,15.0,5, name="linespace").numpy()
```

Out[65]:
```
array([10.  , 11.25, 12.5 , 13.75, 15.  ], dtype=float32)
```

In [68]:
```
indices=[1,2,3,0]
depth=4
tf.one_hot(indices,depth).numpy()
```

Out[68]:
```
array([[0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [1., 0., 0., 0.]], dtype=float32)
```

In [69]:
```
start=5
limit=30
delta=3
tf.range(start,limit,delta).numpy()
```

Out[69]:
```
array([ 5,  8, 11, 14, 17, 20, 23, 26, 29])
```

In [71]:
```
a=tf.constant([5,1,6,4,3,8,7,2])
tf.sort(a,direction="ASCENDING").numpy()
```

Out[71]:
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

In [79]:
```
a=tf.constant([5,1,6,4,3,8,7,2])
tf.sort(a,direction="DESCENDING").numpy()
```

Out[79]:
```
array([8, 7, 6, 5, 4, 3, 2, 1])
```

In [84]:
```
a=tf.constant([[5,1,6],[3,3,3]],tf.int32)
b=tf.constant([1,3])
tf.tile(a,b)
```

Out[84]:
```
<tf.Tensor: shape=(2, 9), dtype=int32, numpy=
array([[5, 1, 6, 5, 1, 6, 5, 1, 6],
       [3, 3, 3, 3, 3, 3, 3, 3, 3]])>
```

Activation Function

Activation function scales output of a node given an input or set of input.

Four Activation functions are defined below.

In [12]:
```
x = np.linspace(-5,5,200)
y_relu = tf.nn.relu(x)
```

In [13]:
```
y_sigmoid = tf.nn.sigmoid(x)
y_sigmoid
```

Out[13]:
```
<tf.Tensor 'Sigmoid_1:0' shape=(200,) dtype=float64>
```

In [14]:
```
y_tanh = tf.nn.tanh(x)
y_tanh
```

Out[14]:
```
<tf.Tensor 'Tanh_1:0' shape=(200,) dtype=float64>
```

In [15]:
```
y_softplus = tf.nn.softplus(x)
y_softplus
```

Out[15]:
```
<tf.Tensor 'Softplus:0' shape=(200,) dtype=float64>
```

In [26]:
```
import tensorflow as tf
tf.compat.v1.disable_eager_execution()

with tf.compat.v1.Session() as val:
    y_relu, y_sigmoid, y_tanh, y_softplus = val.run([y_relu, y_sigmoid, y_tanh, y_softplus])
```

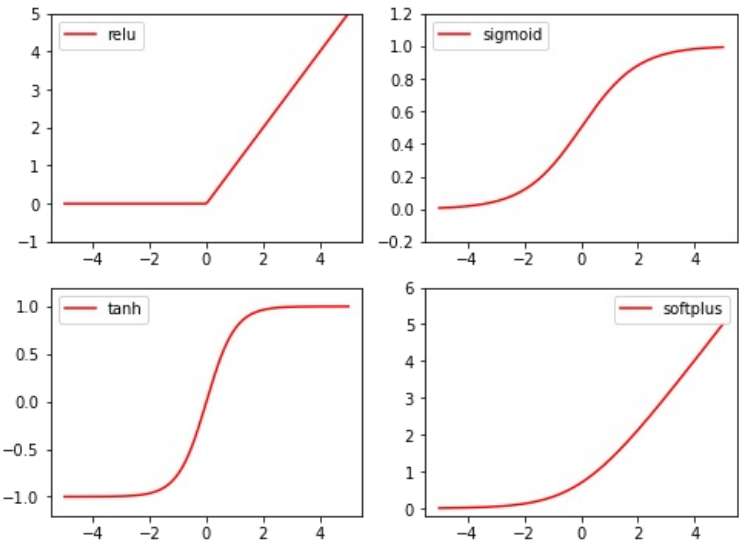
In [29]:
```
from matplotlib import pyplot as plt
plt.figure(1, figsize=(8, 6))
plt.subplot(221)
plt.plot(x, y_relu, c='red', label='relu')
plt.ylim((-1, 5))
plt.legend(loc='best')

plt.subplot(222)
plt.plot(x, y_sigmoid, c='red', label='sigmoid')
plt.ylim((-0.2, 1.2))
plt.legend(loc='best')

plt.subplot(223)
plt.plot(x, y_tanh, c='red', label='tanh')
plt.ylim((-1.2, 1.2))
plt.legend(loc='best')

plt.subplot(224)
plt.plot(x, y_softplus, c='red', label='softplus')
plt.ylim((-0.2, 6))
plt.legend(loc='best')

plt.show()
```



MUHAMMAD BILAL