In [ ]:

```
from __future__ import absolute_import, division, print_function, unicode_literals

try:
  %tensorflow_version 2.x
except Exception:
  pass
```

In [ ]:

```
import tensorflow as tf
import numpy as np

# random seed for always same result from TF2
tf.random.set_seed(1)
np.random.seed(1)
```

In [ ]:

```
class Node:
    def __init__(self):
        self.w = tf.Variable(tf.random.normal([2, 1]))
        self.b = tf.Variable(tf.random.normal([1, 1]))

    def __call__(self, x):
        return self.preds(x)

    def preds(self,x):
        # forward propagation
        out = tf.matmul(x,self.w)
        out = tf.add(out, self.b)
        out = tf.nn.sigmoid(out)
        return out

    def loss(self,y_pred, y):
        return tf.reduce_mean(tf.square(y_pred - y))

    def train(self, inputs, outputs, learning_rate):
        epochs = range(10000)
        for i, epoch in enumerate(epochs):
            with tf.GradientTape() as t:
                current_loss = self.loss(self.preds(inputs), outputs)
                if i % 1000 == 0:
                    print(str(i) + " epoch, loss: "+str(current_loss.numpy()))
                # back propagation
                dW, db = t.gradient(current_loss, [self.w, self.b])
                self.w.assign_sub(learning_rate * dW)
                self.b.assign_sub(learning_rate * db)
```

In [ ]:

```
# AND operation
inputs = tf.constant([[0.0,0.0], [0.0,1.0], [1.0,0.0], [1.0,1.0]])
outputs = tf.constant([[0.0], [0.0], [0.0], [1.0]])

node = Node()
# train
node.train(inputs, outputs, 0.01)
# test
assert node([[0.0,0.0]]).numpy()[0][0] < 0.5
assert node([[0.0,1.0]]).numpy()[0][0] < 0.5
assert node([[1.0,0.0]]).numpy()[0][0] < 0.5
assert node([[1.0,1.0]]).numpy()[0][0] >= 0.5
```

```
0 epoch, loss: 0.14193062
1000 epoch, loss: 0.12047812
2000 epoch, loss: 0.10478282
3000 epoch, loss: 0.09277938
```

```
4000 epoch, loss: 0.08325236
5000 epoch, loss: 0.075465575
6000 epoch, loss: 0.06895569
7000 epoch, loss: 0.06341742
8000 epoch, loss: 0.05864069
9000 epoch, loss: 0.054475207
```

In [ ]:

```
# OR operation
inputs = tf.constant([[0.0,0.0], [0.0,1.0], [1.0,0.0], [1.0,1.0]])
outputs = tf.constant([[0.0], [1.0], [1.0], [1.0]])

node = Node()
# train
node.train(inputs, outputs, 0.01)
# test
assert node([[0.0,0.0]]).numpy()[0][0] < 0.5
assert node([[0.0,1.0]]).numpy()[0][0] >= 0.5
assert node([[1.0,0.0]]).numpy()[0][0] >= 0.5
assert node([[1.0,1.0]]).numpy()[0][0] >= 0.5
```

```
0 epoch, loss: 0.2860349
1000 epoch, loss: 0.11865921
2000 epoch, loss: 0.09779619
3000 epoch, loss: 0.08329055
4000 epoch, loss: 0.07180754
5000 epoch, loss: 0.062597096
6000 epoch, loss: 0.055128783
7000 epoch, loss: 0.049000803
8000 epoch, loss: 0.04391424
9000 epoch, loss: 0.039646696
```

In [ ]:

```
print("Node Weights: ",node.w.numpy())
```

```
Node Weights:  [[2.461971 ]
 [2.4734807]]
```

In [ ]:

```
print("Node Bias: ",node.b.numpy())
```

```
Node Bias:  [[-0.8695577]]
```

In [ ]:

```

```