In [2]:
```python
import numpy as np
```

In [11]:
```python
class Perceptron:
    def __init__(self,input_size):
        self.weights=np.zeros(input_size)
        self.bias=0

    def predict(self,inputs):
        summation=np.dot(inputs,self.weights.T)+self.bias
        return 1 if summation>=0 else 0

    def train(self,inputs,labels,learning_rate,epochs):
        for epoch in range(epochs):
            for input,label in zip(inputs,labels):
                prediction=self.predict(input)
                self.weights+=learning_rate*(label-prediction)*input
                self.bias+=learning_rate*(label-prediction)
```

In [12]:
```python
inputs=np.array([[0,0],[0,1],[1,0],[1,1]])
labels=np.array([0,0,0,1])
```

In [13]:
```python
model=Perceptron(2)
```

In [14]:
```python
model.train(inputs,labels,0.01,10000)
```

In [15]:
```python
print("x1   x2   y")
print()
for i in range(len(inputs)):
    print(inputs[i][0]," ",inputs[i][1]," ",model.predict(inputs[i]))
```

```
x1   x2   y

0   0   0
0   1   0
1   0   0
1   1   1
```

In [16]:
```python
weights=model.weights
weights
```

Out[16]:
```
array([0.02, 0.01])
```

In [17]:
```python
w1=weights[0]
w2=weights[1]
```

In [18]:
```python
b=model.bias
b
```

Out[18]:
```
-0.03
```

In [19]:
```python
m=-(w1/w2)
c=-b/w2
```

In [20]:
```python
x_intercepts=np.linspace(-2,2,5)
y_intercepts=[]
for x in x_intercepts:
    y_intercepts.append((m*x)+c)
y_intercepts=np.array(y_intercepts)
```

In [21]:
```python
x_intercepts
```

Out[21]:
```
array([-2., -1.,  0.,  1.,  2.])
```

In [22]:
```python
y_intercepts/=1.5
```

In [23]:
```python
import matplotlib.pyplot as plt
```

In [24]:
```python
x_point=[x[0] for x in inputs]
y_point=[x[1] for x in inputs]

plt.plot(x_intercepts,y_intercepts)
plt.scatter(x_point,y_point)
```

Out[24]:
```
<matplotlib.collections.PathCollection at 0x1a042d3ff10>
```