In [141]:

```
import numpy as np
```

In [142]:

```
def step_function(x): #ステップ関数
    y = x > 0
    return y.astype(np.int)
```

ANDゲート

In [143]:

```
def AND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    theta = -0.7
    y = np.sum(w*x) + theta
    if y <= 0:
        return 0
    else:
        return 1
```

In [144]:

```
for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
        out = AND(xs[0], xs[1])
        print(str(xs) + " -> " + str(out))
```

```
(0, 0) -> 0
(1, 0) -> 0
(0, 1) -> 0
(1, 1) -> 1
```

NANDゲート

In [145]:

```
def NAND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([-0.5, -0.5])
    theta = 0.7
    y = np.sum(w*x) + theta
    if y <= 0:
        return 0
    else:
        return 1
```

In [146]:

```
for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
        out = NAND(xs[0], xs[1])
        print(str(xs) +  " -> " + str(out))
```

```
(0, 0) -> 1
(1, 0) -> 1
(0, 1) -> 1
(1, 1) -> 0
```

ORゲート

In [147]:

```
def OR(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    theta = -0.2
    y = np.sum(w*x) + theta
    if y <= 0:
        return 0
    else:
        return 1
```

In [148]:

```
for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
        out = OR(xs[0], xs[1])
        print(str(xs) +  " -> " + str(out))
```

```
(0, 0) -> 0
(1, 0) -> 1
(0, 1) -> 1
(1, 1) -> 1
```

XORゲート

In [149]:

```
def XOR(x1, x2):
    return AND(NAND(x1, x2), OR(x1, x2))
```

In [150]:

```
for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
        out = XOR(xs[0], xs[1])
        print(str(xs) +  " -> " + str(out))
```

```
(0, 0) -> 0
(1, 0) -> 1
(0, 1) -> 1
(1, 1) -> 0
```

感想

AND,NAND,ORゲートはバイアスと重みだけが違い、あとは同じとは思っていなかった。 XORゲートはAND,NAND,ORゲートの組み合わせで作れると知れてよかった。

参考文献

ゼロから作るDeep Leaning Pythonで学ぶディープラーニングの基礎