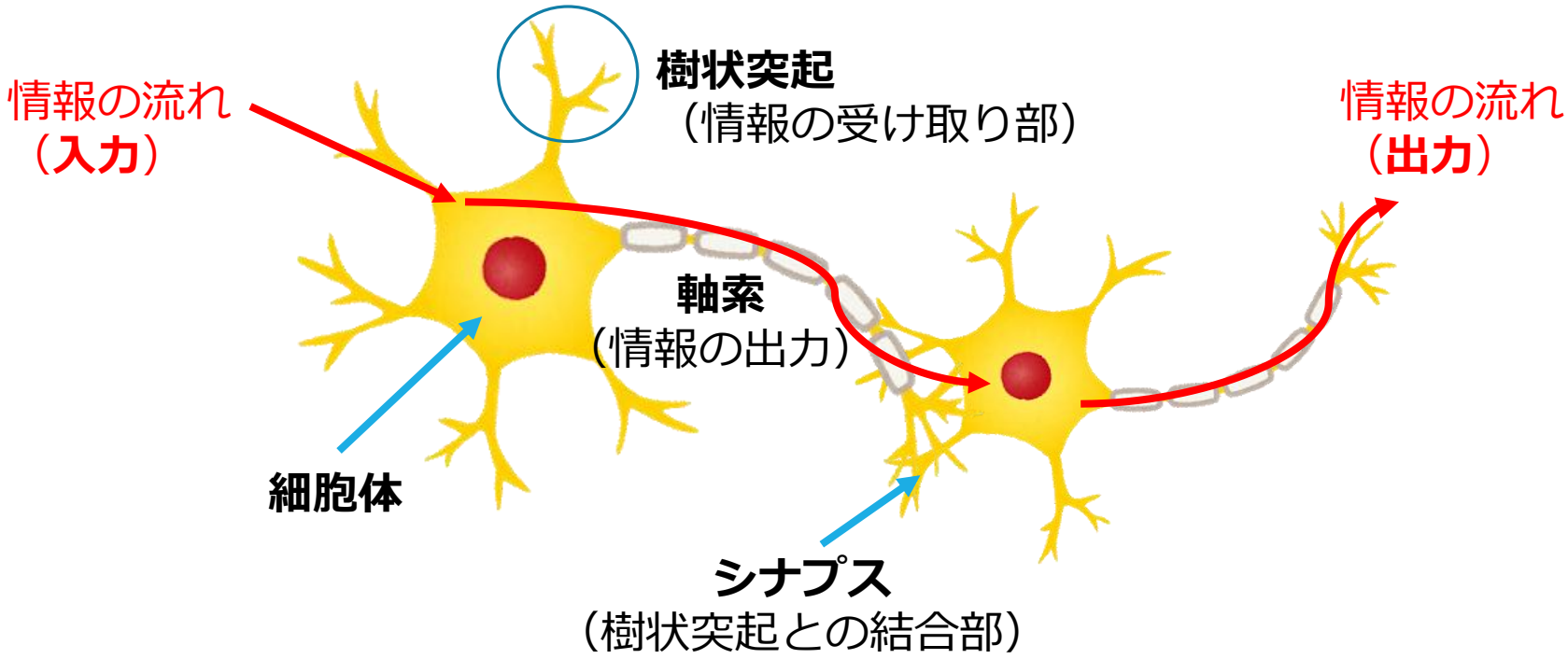
深層学習入門
＃1 単層パーセプトロンと論理回路

たくのろじぃ / takunology

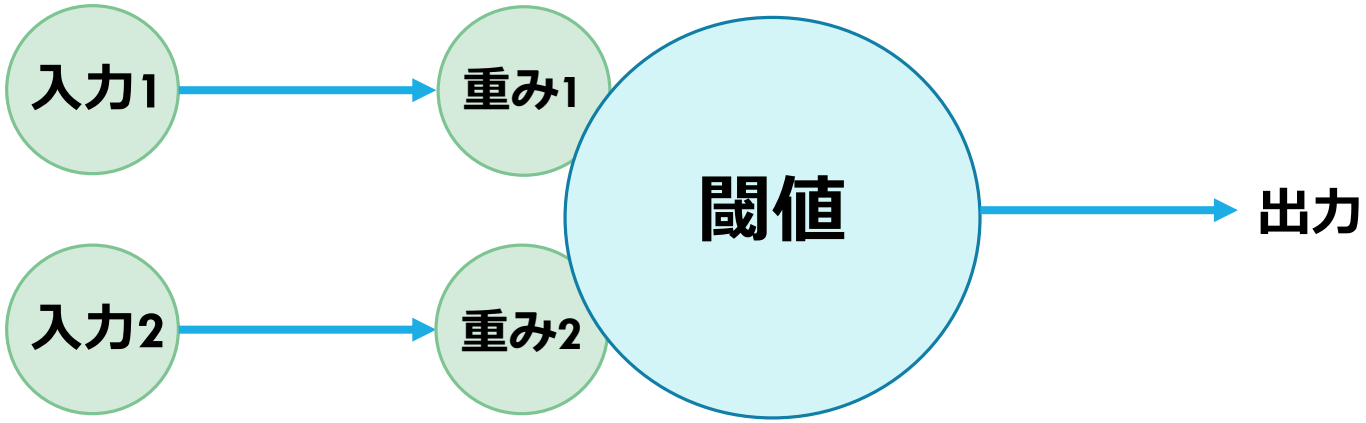
ニューロン

- 情報や刺激などを伝達する神経細胞
- 人間の脳に100億～1000億ほどある（成人）
- 情報は樹状突起で受け取り、細胞体で処理されて軸索を通って出力される
- シナプスから情報を受け取るのに 0.1～0.2 [ms] 程度



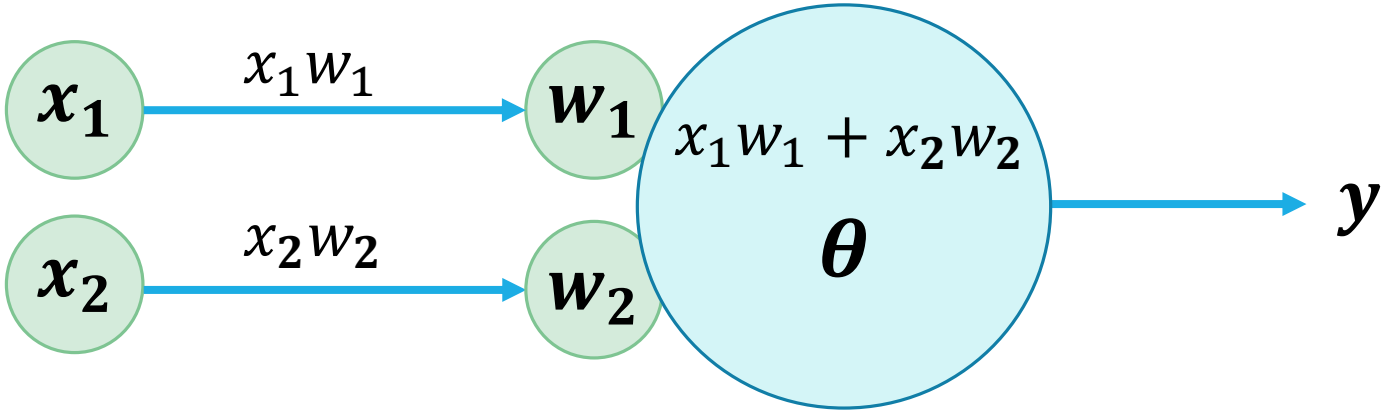
ニューロンモデル

- ニューロンを模した型
- 2入力に対して重みをつけ、閾値をもとに出力を定める
- 閾値を超えた状態を「ニューロンが発火した」という



単層パーセプトロン

- ニューロンモデルをもとにしたアルゴリズム
- パーセプトロンは1つで構成されているので単層
- パラメータ：入力 (x_1, x_2) 出力 (y) 重み (w_1, w_2) 閾値 (θ)
- 入力と重みの積を加算した値が閾値で判定される



$$y = \begin{cases} 0 \ (x_1 w_1 + x_2 w_2 \leq \theta) & \text{入力値が閾値以下の場合、出力は0} \\ 1 \ (x_1 w_1 + x_2 w_2 > \theta) & \text{入力値が閾値を超えた場合、出力は1} \end{cases}$$

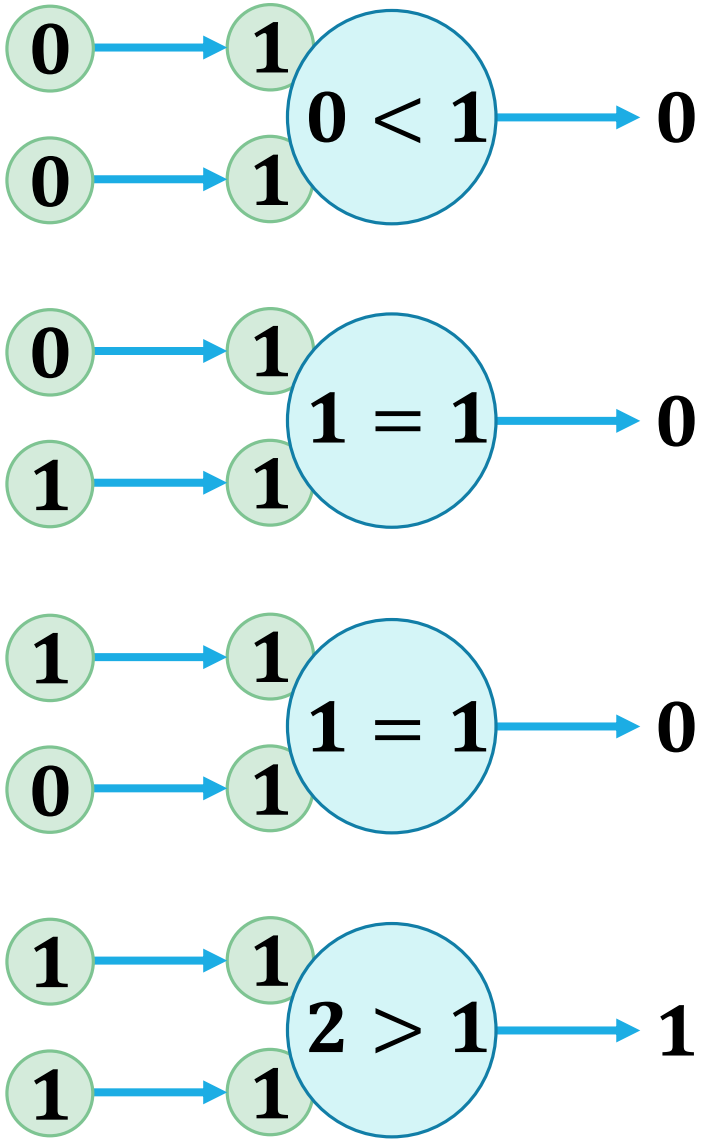
簡単な例1

- 最も単純な2値の論理演算
- 重みのパラメータは1 閾値も1
- 閾値を超えた(1を超えた)とき出力は1

例）AND 論理回路

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

閾値と重みは異なる値の組み合わせも可能

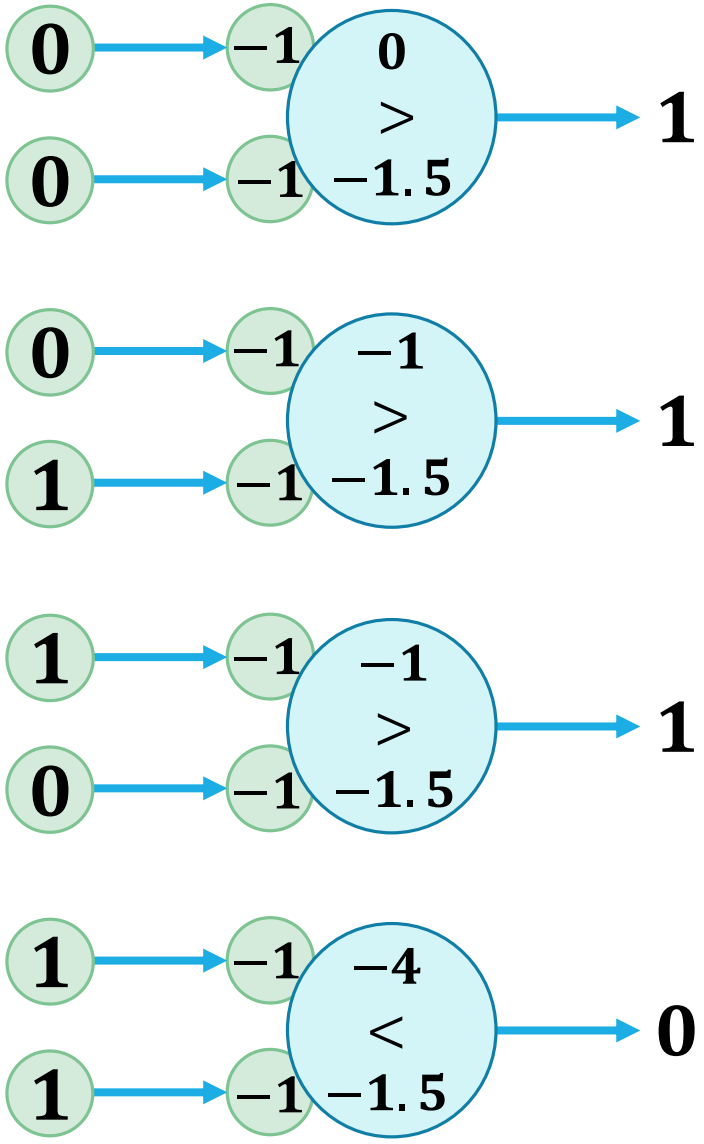


簡単な例2

- ANDを反転したNAND論理演算
- 重みのパラメータは -1 閾値は -1.5
- 閾値を超えた(-1.5を超えた)とき出力は1

例）NAND 論理回路

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

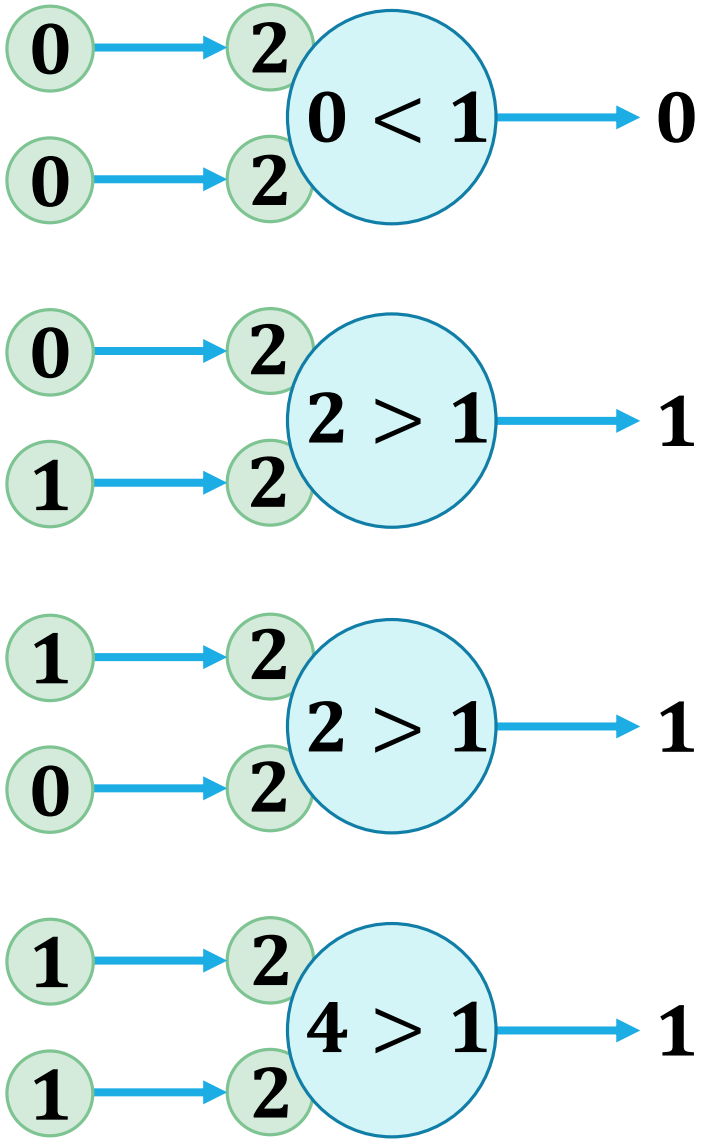


簡単な例3

- 最も単純な2値の論理演算
- 重みのパラメータは2 閾値は1
- 閾値を超えた(1を超えた)とき出力は1

例）OR 論理回路

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



ハンズオン ＃1

- C# (.NET Core 3.1) でAND論理回路の実装
- パラメータは $(w_1, w_2, \theta) = (1, 1, 1)$
- 閾値を超えた(1を超えた)とき出力は1
- パラメータをいじって色々試す

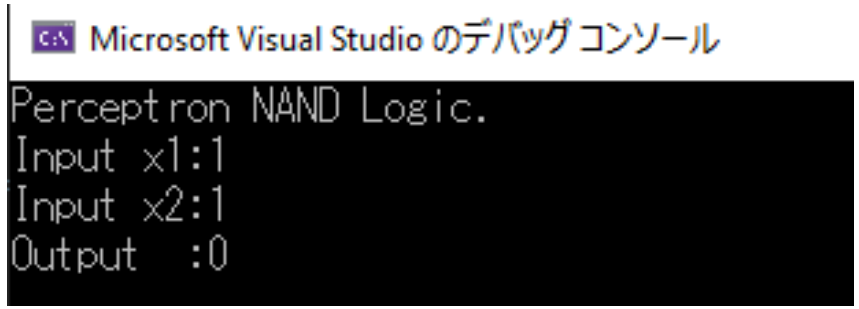
```
Microsoft Visual Studio のデバッグ コンソール
Perceptron AND Logic.
Input x1:0
Input x2:1
Output  :0
```

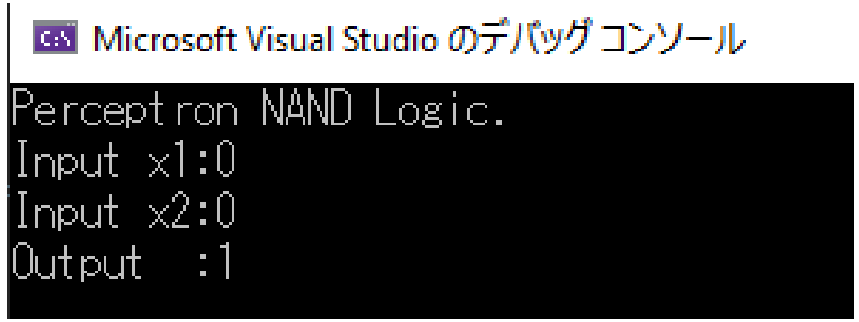
```
Microsoft Visual Studio のデバッグ コンソール
Perceptron AND Logic.
Input x1:1
Input x2:1
Output  :1
```

```
namespace Perceptron_AND
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Console.WriteLine("Perceptron AND Logic.");

                Console.Write("Input x1:");
                var x1 = double.Parse(Console.ReadLine());
                Console.Write("Input x2:");
                var x2 = double.Parse(Console.ReadLine());

                Console.WriteLine($"Output  :{AND(x1, x2)}");
            }
            catch(Exception e)
            {
                Console.WriteLine(e);
            }
        }

        public static int AND(double x1, double x2)
        {
            //重みのパラメータ
            var w1 = 1;
            var w2 = 1;
            var threshold = 1; // 閾値
            var tmp = (x1 * w1) + (x2 * w2); //積の和

            if(threshold < tmp){ return 1;} // 閾値を超えた場合
            else { return 0; } // 閾値以下の場合
        }
    }
}
```

ハンズオン ＃2

- C# (.NET Core 3.1) でNAND論理回路の実装
- パラメータは $(w_1, w_2, \theta) = (-1, -1, -1.5)$
- 閾値を超えた(-1.5を超えた)とき出力は1
- パラメータをいじって色々試す

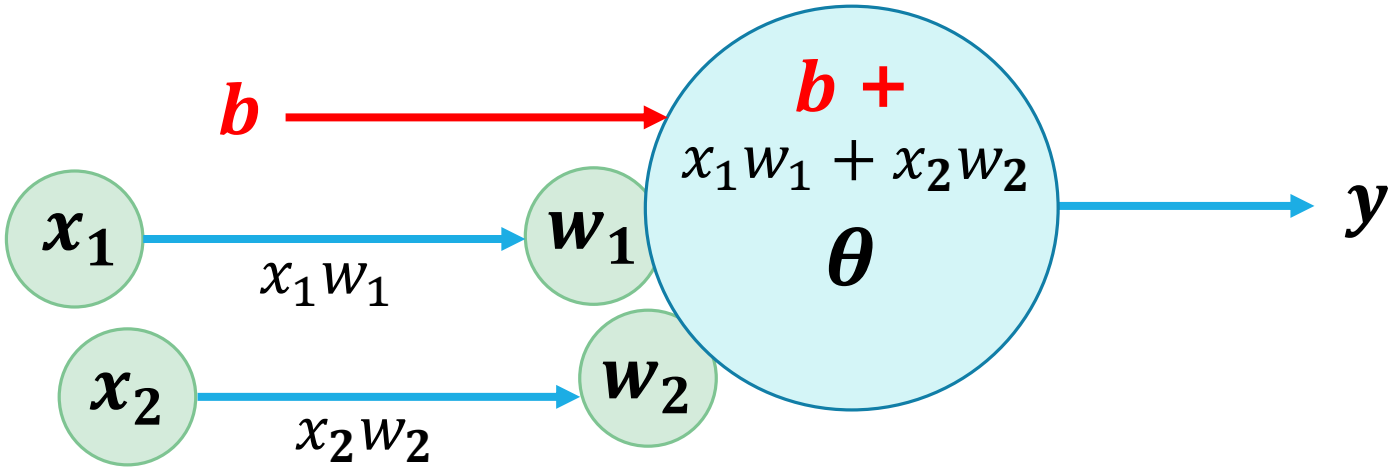





```
static void Main(string[] args)
{
    try
    {
        Console.WriteLine("Perceptron NAND Logic.");

        Console.Write("Input x1:");
        var x1 = double.Parse(Console.ReadLine());
        Console.Write("Input x2:");
        var x2 = double.Parse(Console.ReadLine());

        Console.WriteLine($"Output  :{NAND(x1, x2)}");
    }
    catch (Exception e)
    {
        Console.WriteLine(e);
    }
}

public static int NAND(double x1, double x2)
{
    //重みのパラメータ
    var w1 = -1;
    var w2 = -1;
    var threshold = -1.5; // 閾値
    var tmp = (x1 * w1) + (x2 * w2); //積の和

    if (threshold < tmp) { return 1; } // 閾値を超えた場合
    else { return 0; } // 閾値以下の場合
}
```

単層パーセプトロン(続き)

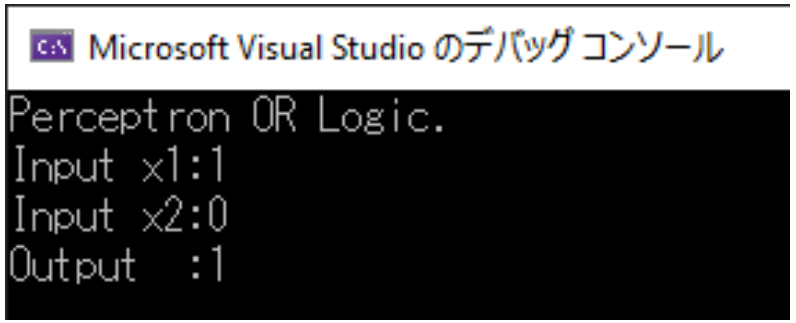
- 単層パーセプトロンにバイアスを付与する
- ハイアスはニューロンの発火のしやすさを調整する
- パラメータ：入力 (x_1, x_2) 出力 (y) 重み (w_1, w_2) 閾値 (θ) + バイアス(b)
- バイアス→出力調整，重み→入力調整

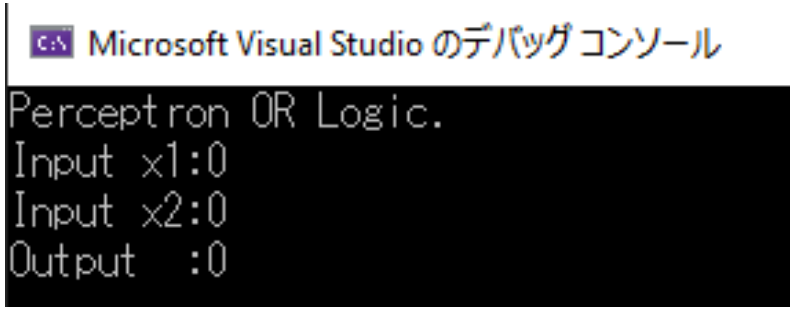


$$y = \begin{cases} 0 \ (b + x_1 w_1 + x_2 w_2 \leq \theta) & \text{入力値が閾値以下の場合、出力は0} \\ 1 \ (b + x_1 w_1 + x_2 w_2 > \theta) & \text{入力値が閾値を超えた場合、出力は1} \end{cases}$$

ハンズオン ＃3

- C# (.NET Core 3.1) でOR論理回路の実装
- パラメータは $(b, w_1, w_2, \theta) = (1, 2, 2, 1)$
- 閾値を超えた(1を超えた)とき出力は1
- パラメータをいじって色々試す

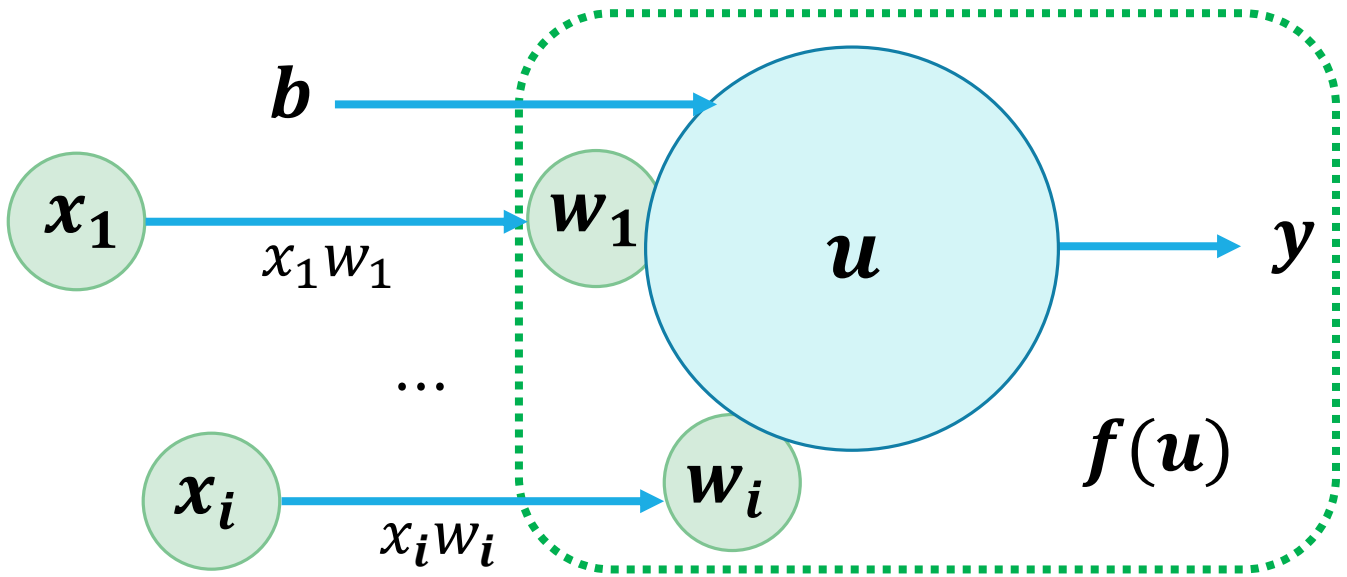





```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            Console.WriteLine("Perceptron OR Logic.");

            Console.Write("Input x1:");
            var x1 = double.Parse(Console.ReadLine());
            Console.Write("Input x2:");
            var x2 = double.Parse(Console.ReadLine());

            Console.WriteLine($"Output  :{OR(x1, x2)}");
        }
        catch (Exception e)
        {
            Console.WriteLine(e);
        }
    }

    public static int OR(double x1, double x2)
    {
        //重みのパラメータ
        var w1 = 2;
        var w2 = 2;
        var threshold = 1; // 閾値
        var b = 1; //バイアス
        var tmp = b + (x1 * w1) + (x2 * w2); //積の和

        if (threshold < tmp) { return 1; } // 閾値を超えた場合
        else { return 0; } // 閾値以下の場合
    }
}
```

補足

- 入力値とバイアスは総和で表せる
- 得られた値 $\boldsymbol{u}$ を用いて出力を決める部分を活性化関数という
- 入力は多数あってもよいが、出力は1つに絞られる



$$b + x_1 w_1 + \cdots + x_i w_i$$

$$\boldsymbol{u} = b + \sum_i x_i w_i$$