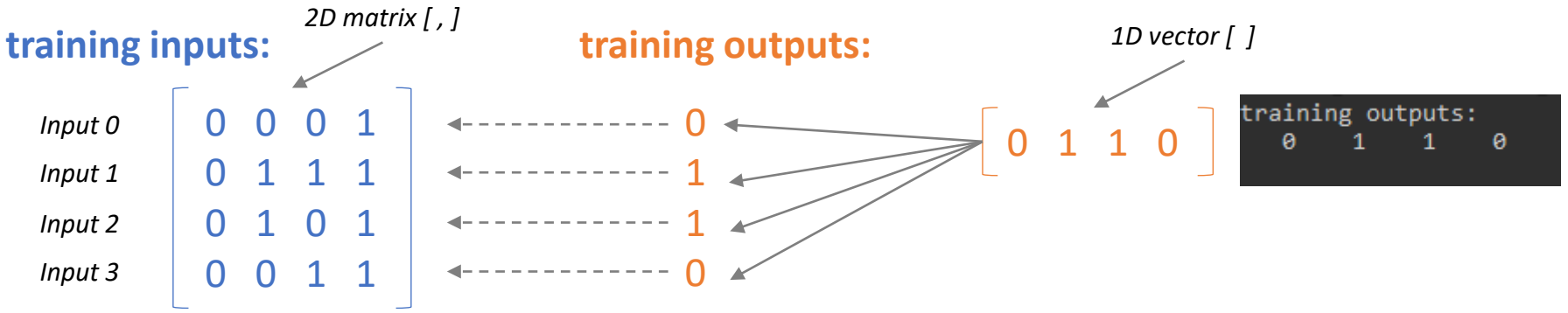
Perceptron simple




```
training inputs:
   0    0    0    1
   0    1    1    1
   0    1    0    1
   0    0    1    1
```

Program:

```
training inputs:
   0    0    0    1
   0    1    1    1
   0    1    0    1
   0    0    1    1

training outputs:
   0    1    1    0

Random weights:
0,236413 -0,597119 -0,201812 -0,335292

Weights after learning:
0,236413 10,3802378952057 -0,206701302097514 -4,98426316406065

Outputs after learning:
0,00679845890907241 0,994454722877481 0,995485559585221 0,00553595940335179

Test input: {   0    1    0    0 }
Result output: 0,999968961088614
```

Calculation

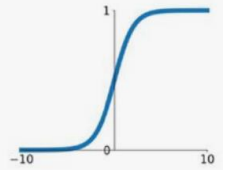
training input:

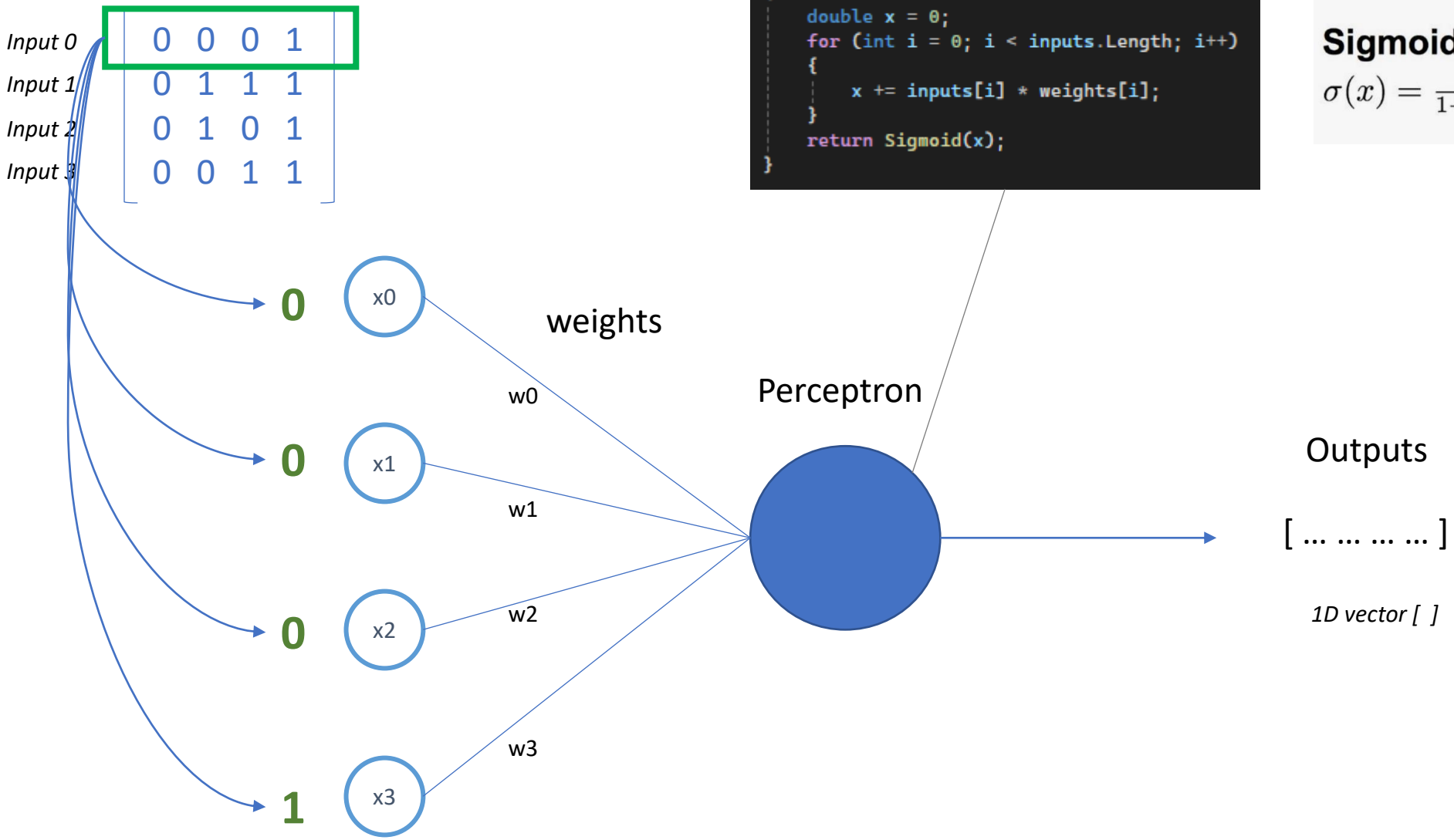
Input 0	0	0	0	1
Input 1	0	1	1	1
Input 2	0	1	0	1
Input 3	0	0	1	1

```
1 reference
public double Calculate(int[] inputs)
{
    double x = 0;
    for (int i = 0; i < inputs.Length; i++)
    {
        x += inputs[i] * weights[i];
    }
    return Sigmoid(x);
}
```

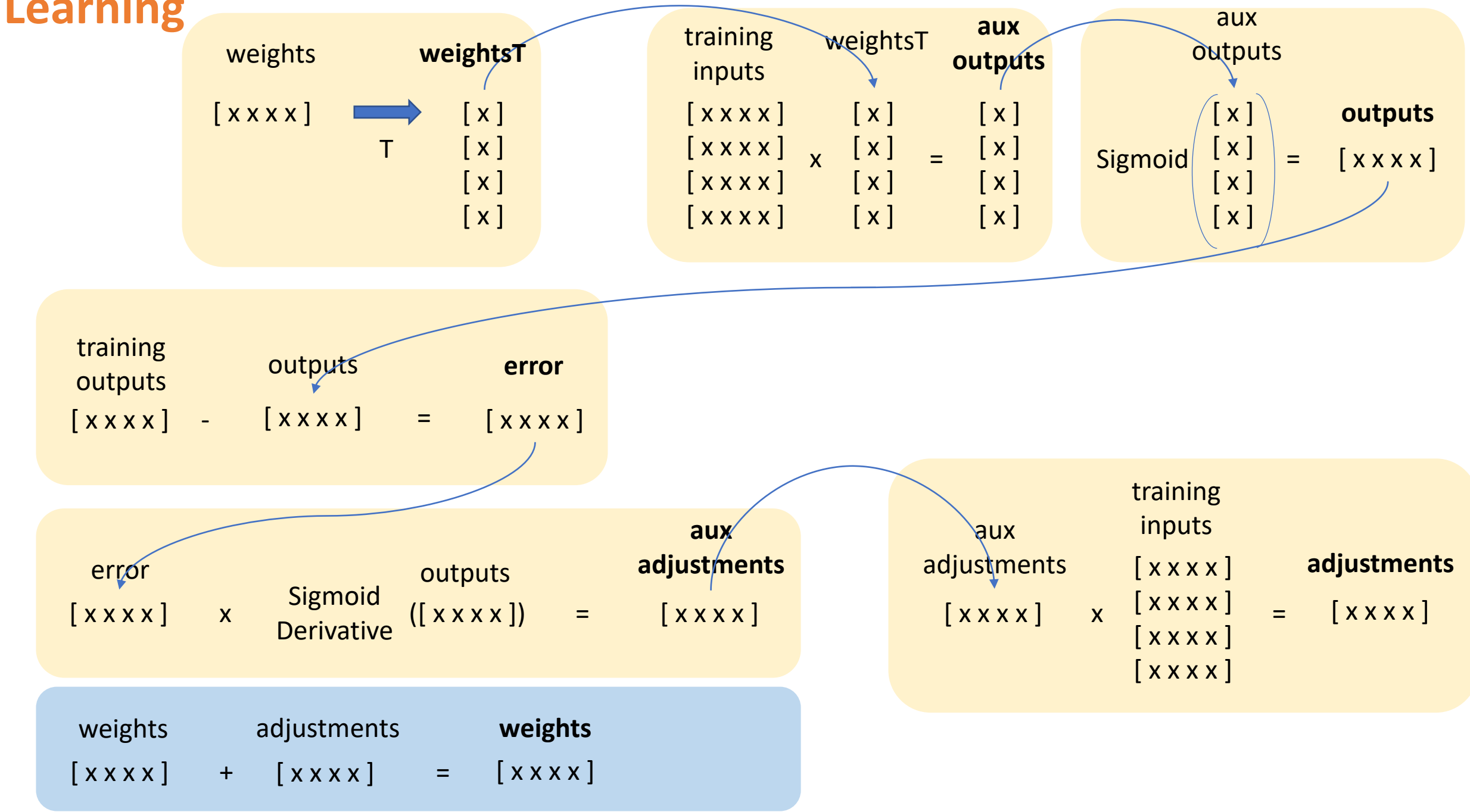
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$





Learning


weights
[x x x x]
T
weightsT
[x]
[x]
[x]
[x]
training inputs
[x x x x]
[x x x x]
[x x x x]
[x x x x]
x
weightsT
[x]
[x]
[x]
[x]
=
aux outputs
[x]
[x]
[x]
[x]
aux outputs
Sigmoid
[x]
[x]
[x]
[x]
=
outputs
[x x x x]
training outputs
[x x x x]
-
outputs
[x x x x]
=
error
[x x x x]
error
[x x x x]
x
Sigmoid Derivative
outputs
([x x x x])
=
aux adjustments
[x x x x]
aux adjustments
[x x x x]
x
training inputs
[x x x x]
[x x x x]
[x x x x]
[x x x x]
=
adjustments
[x x x x]
weights
[x x x x]
+
adjustments
[x x x x]
=
weights
[x x x x]