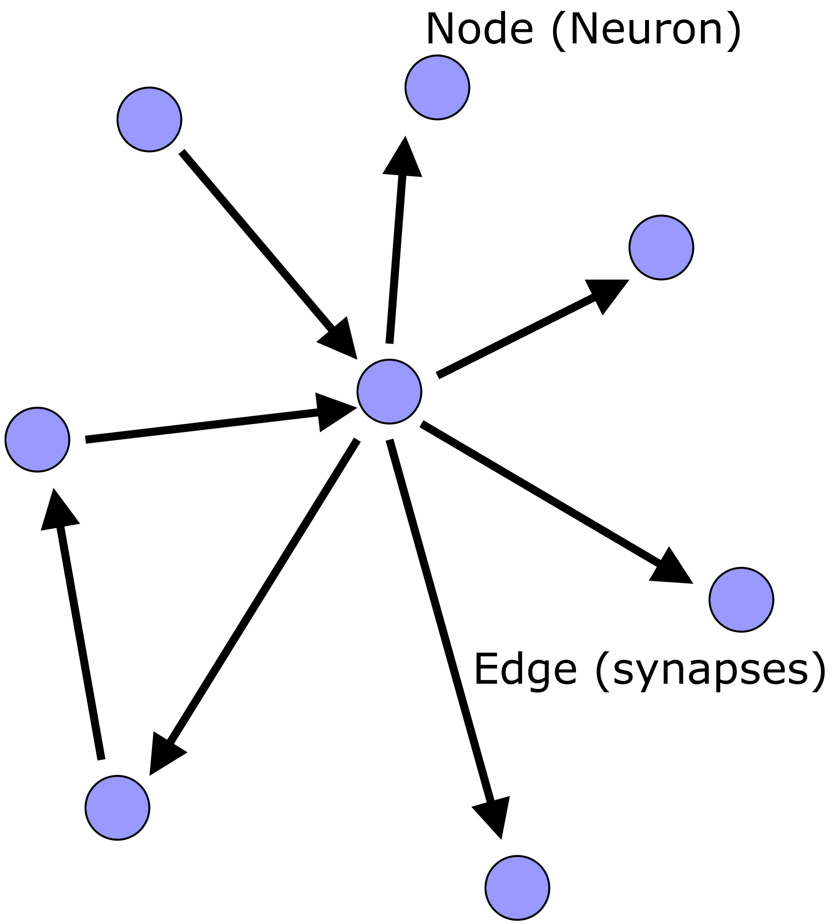
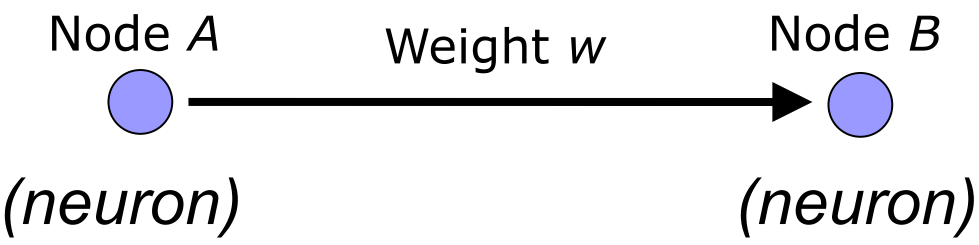
Perceptron Learning

Dave Kauchak
Alexandra Papoutsaki
Zilong Ye
CSCI 051A
Spring 2022

Artificial Neural Networks



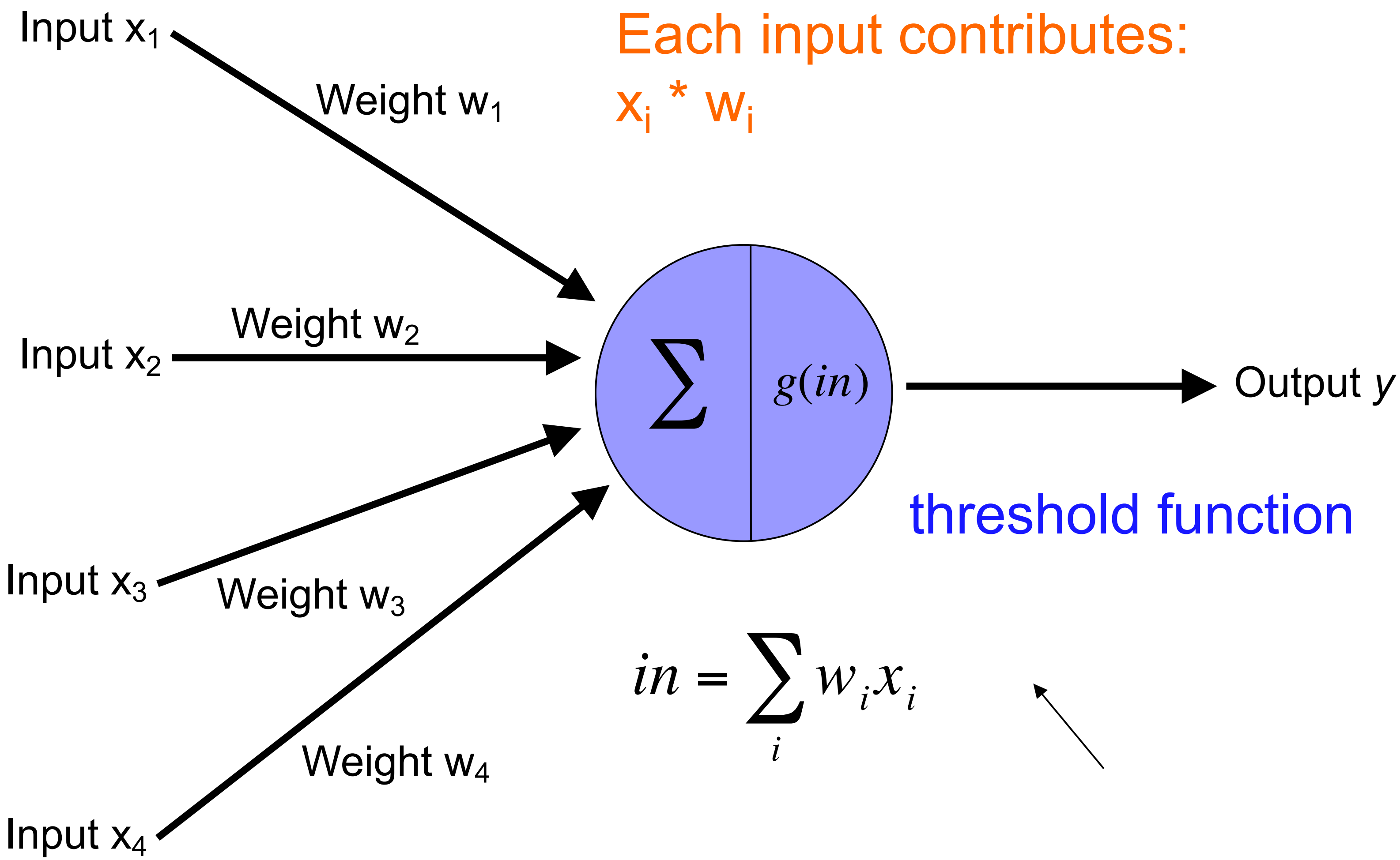


W is the strength of signal sent between A and B.

If *A* fires and *w* is **positive**, then *A* **stimulates** *B*.

If *A fires* and *w* is **negative**, then *A* **inhibits** *B*.

A Single Neuron/Perceptron



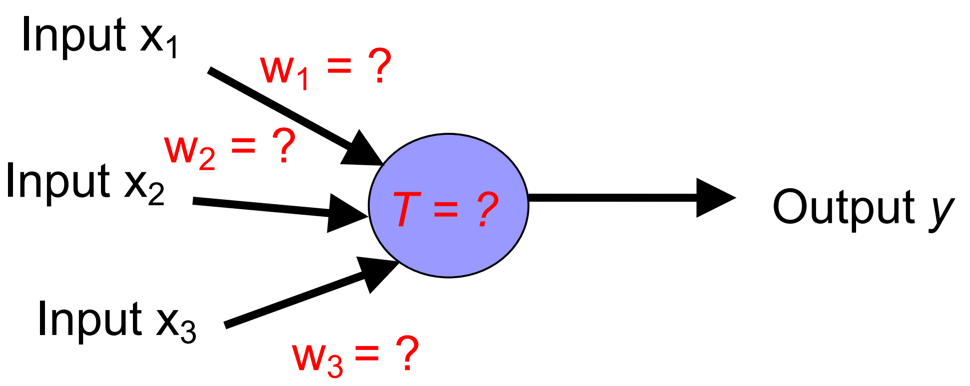
Each input contributes:
$x_i * w_i$

threshold function

$$in = \sum_i w_i x_i$$

Training neural networks

x_1	x_2	x_3	y
0	0	0	1
0	1	0	0
1	0	0	1
1	1	0	0
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	0



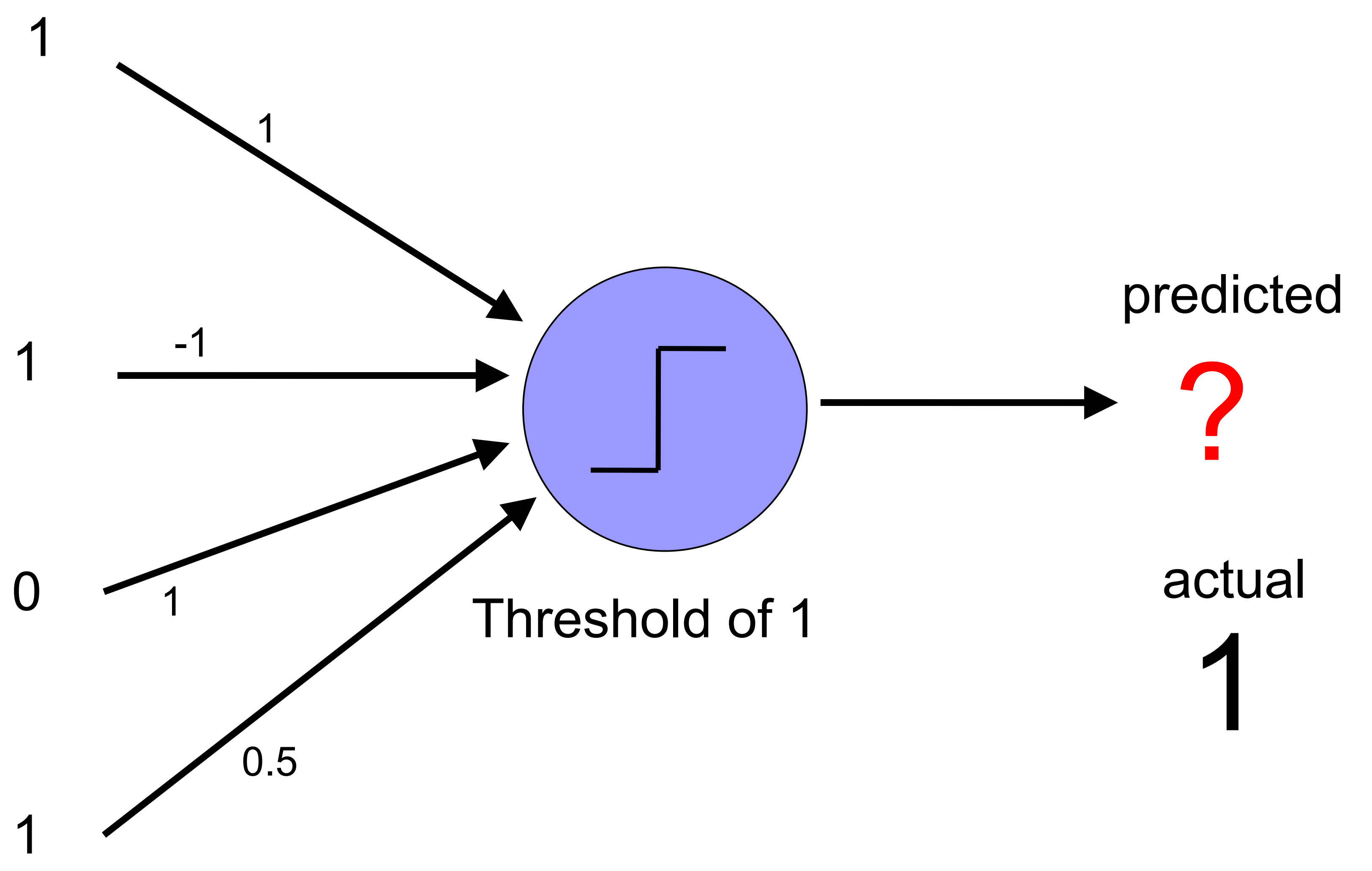
1. start with some initial weights and thresholds
2. show examples repeatedly to NN
3. update weights/thresholds by comparing NN output to actual output

Perceptron learning algorithm

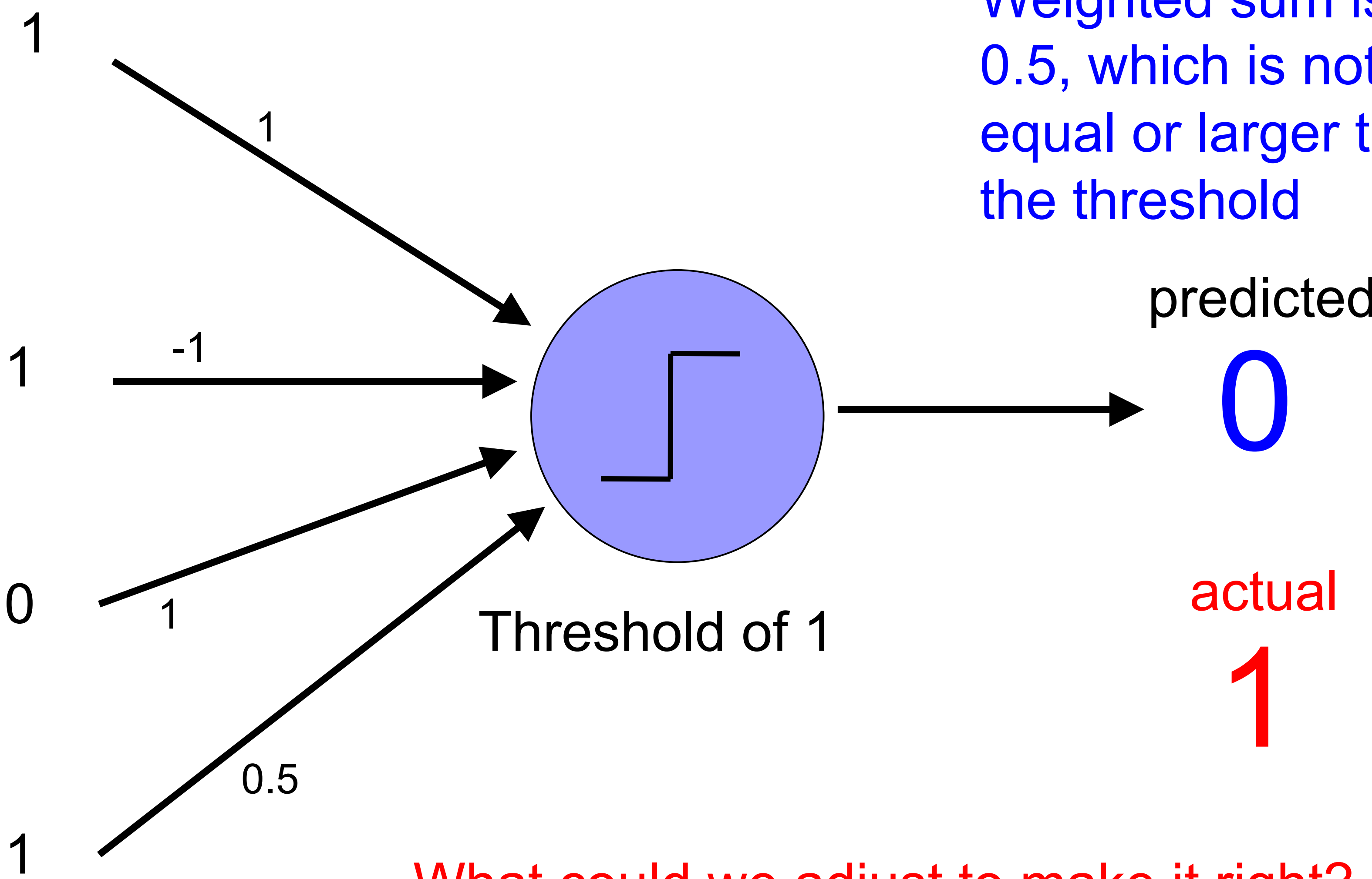
repeat until you get all examples right:

- for each “training” example:
  - calculate current prediction on example
  - if *wrong*:
    - update weights and threshold towards getting this example correct

Perceptron learning



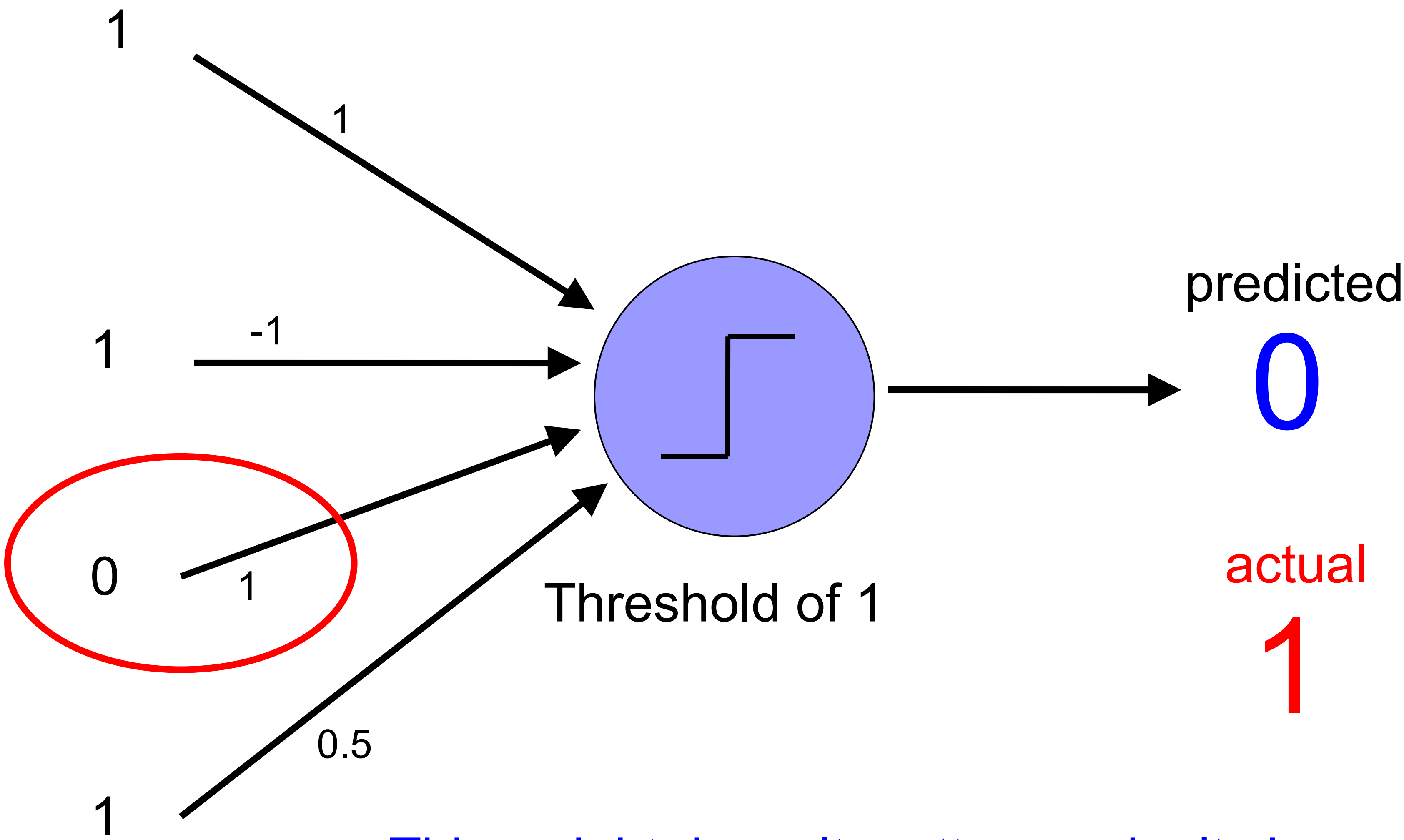
Perceptron learning



Weighted sum is 0.5, which is not equal or larger than the threshold

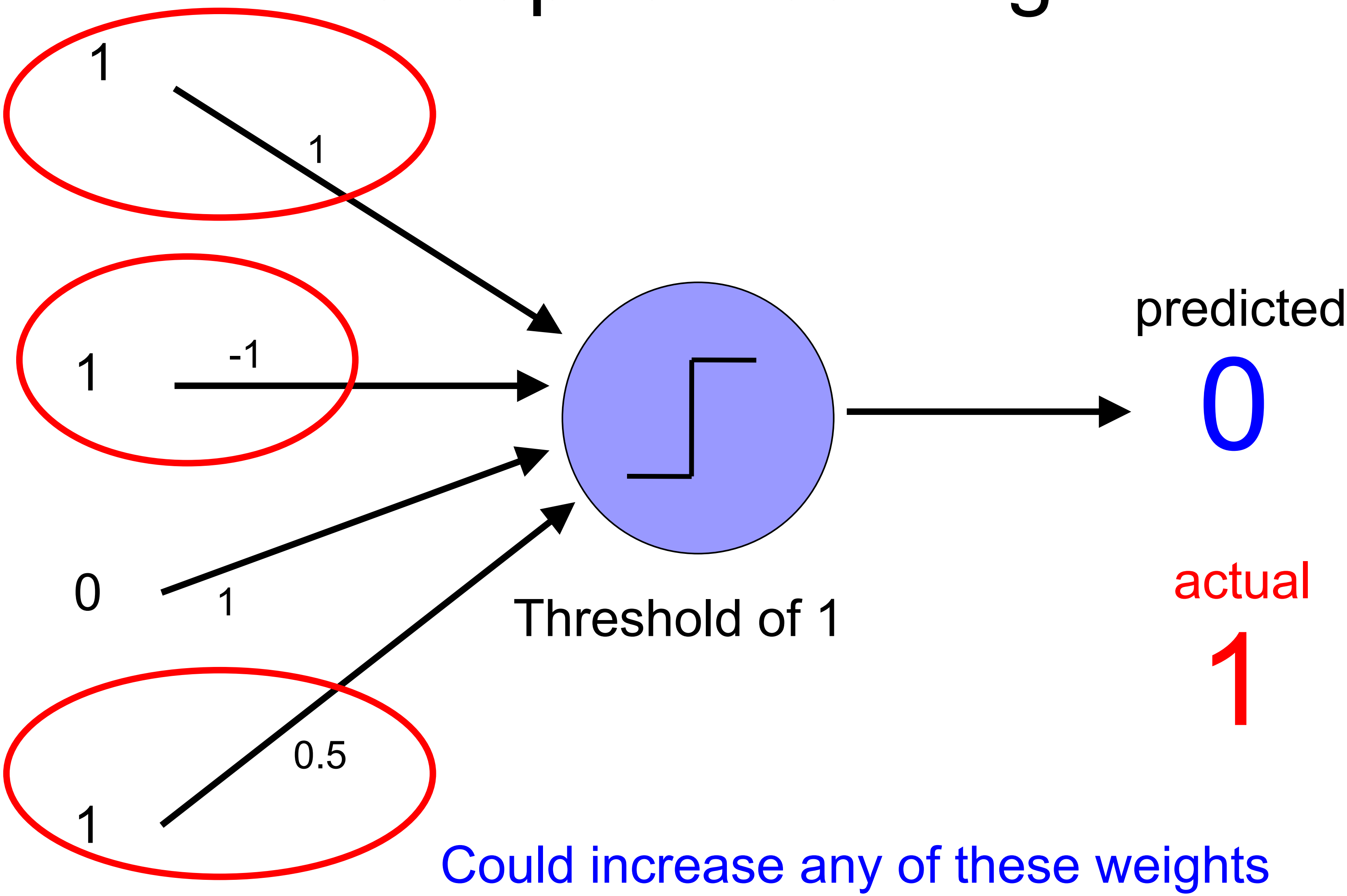
What could we adjust to make it right?

Perceptron learning

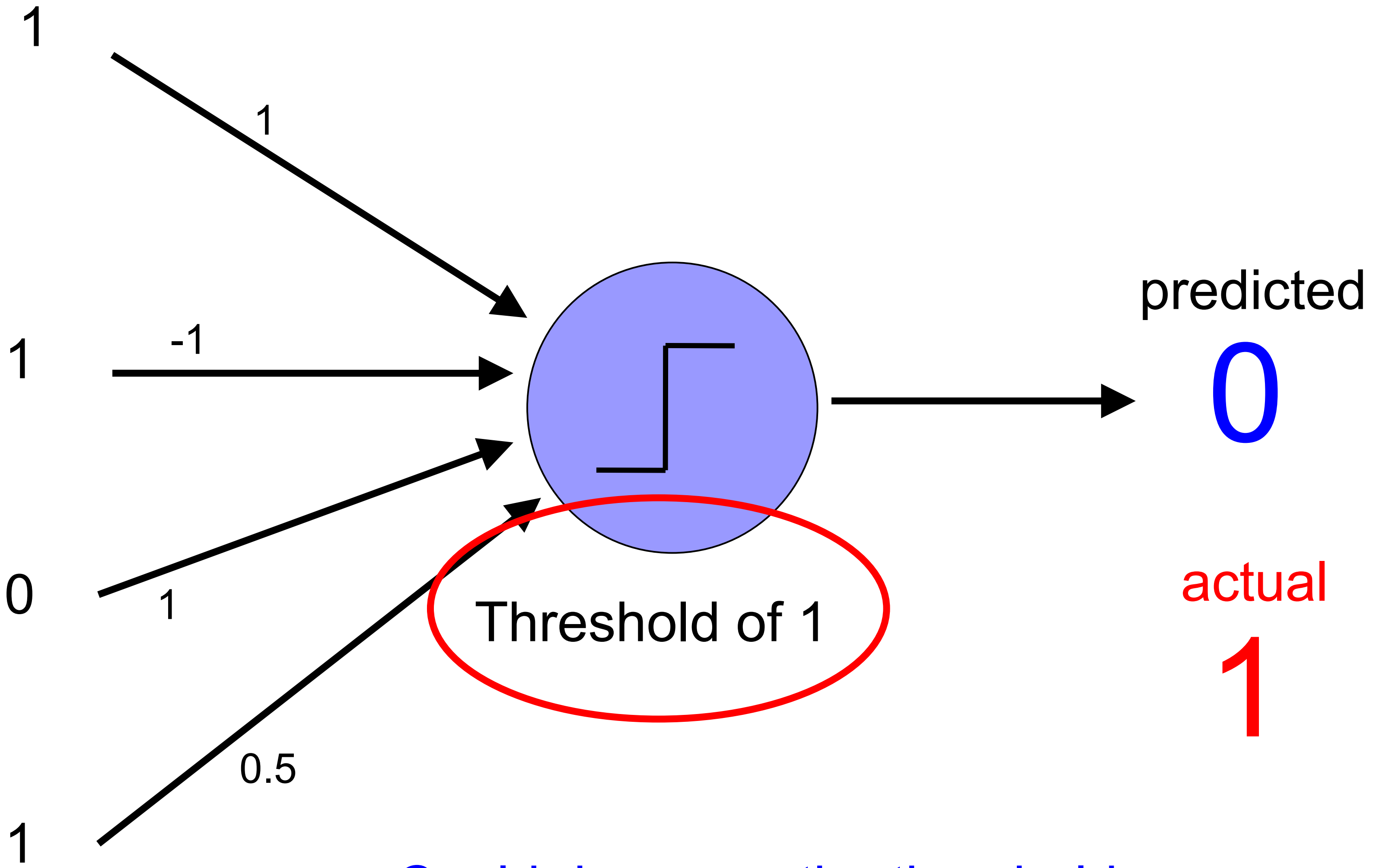


This weight doesn't matter, so don't change

Perceptron learning



Perceptron learning



Could decrease the threshold

Perceptron update rule

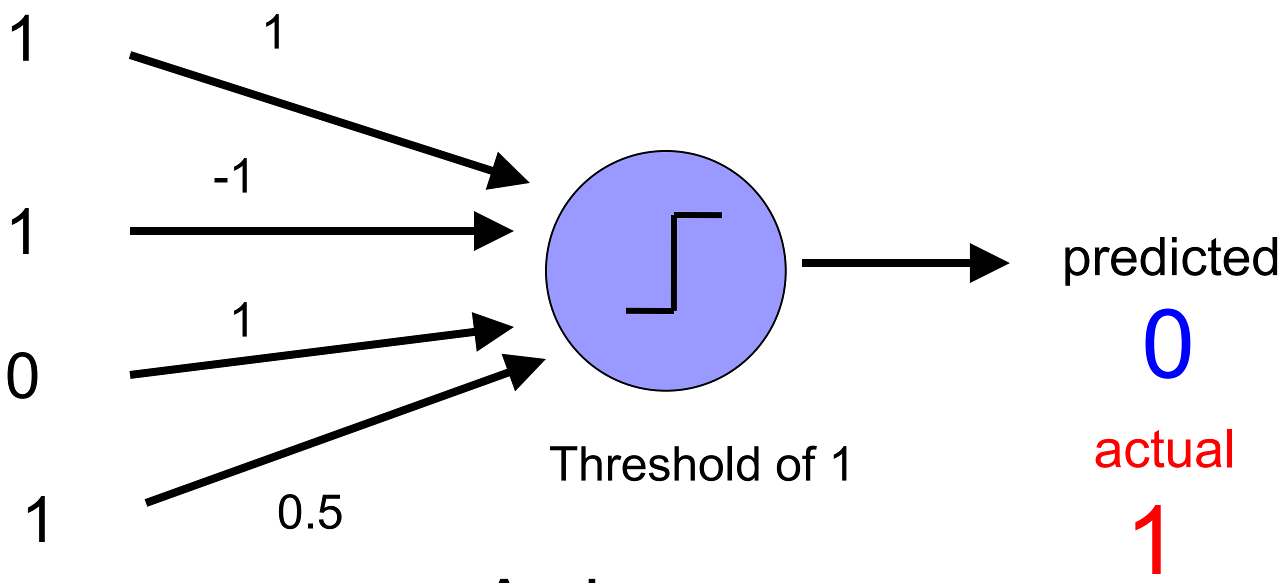
- if *wrong*:
  - update weights and threshold towards getting this example correct

- if *wrong*:

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = \lambda * (\text{actual} - \text{predicted}) * x_i$$

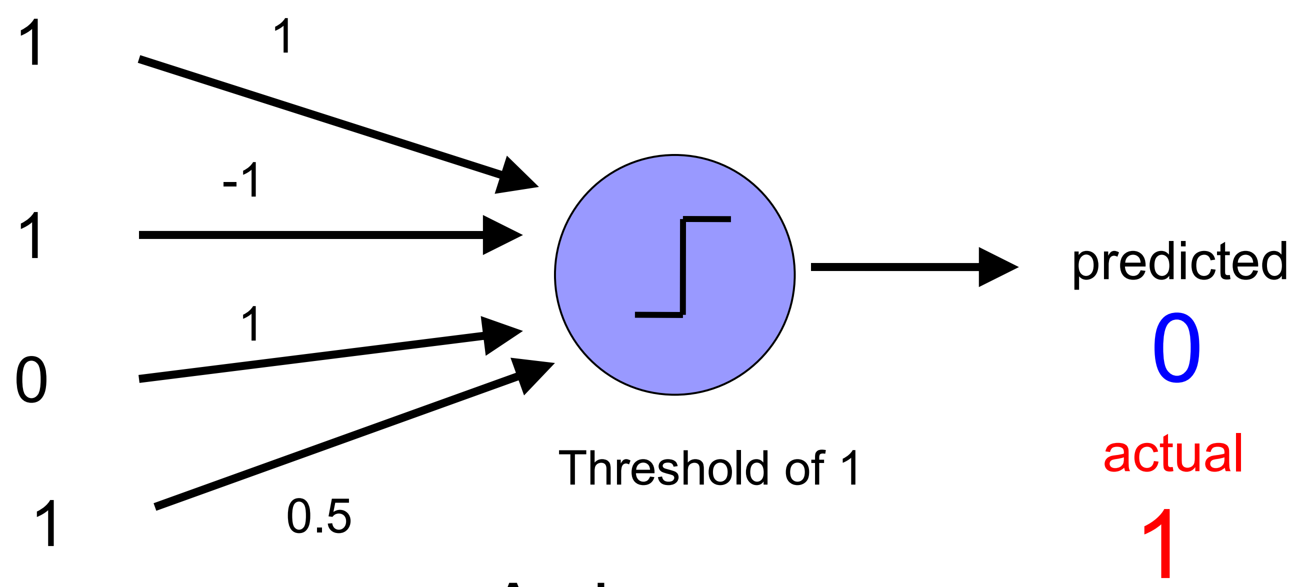
Perceptron learning



$$w_i = w_i + \Delta wi$$

$$\Delta w_i = \lambda * \underbrace{(\text{actual - predicted})}_{\text{What does this do in this case?}} * x_i$$

Perceptron learning

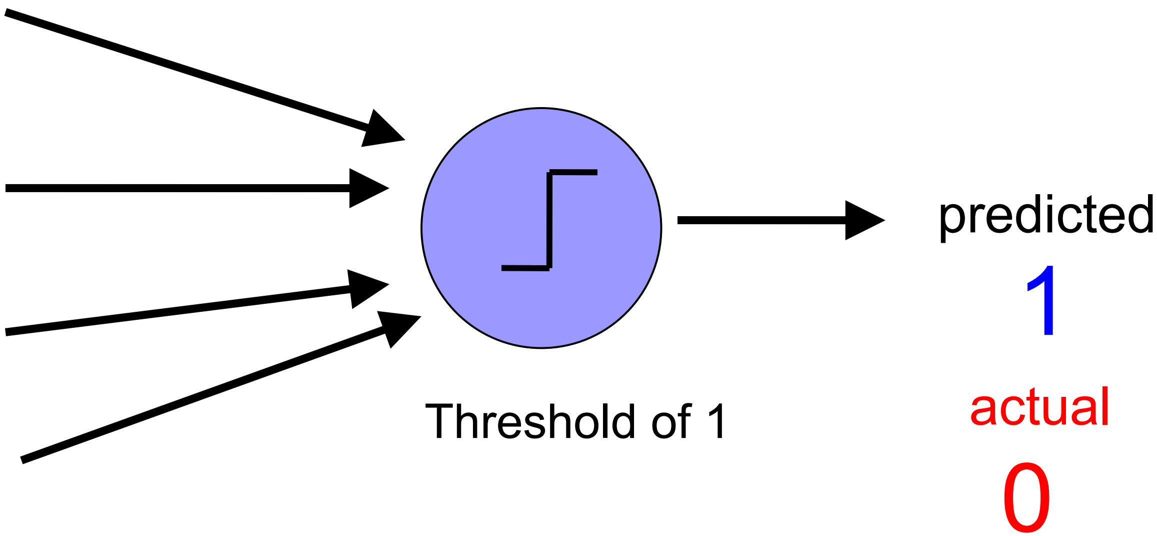


$$w_i = w_i + \Delta wi$$

$$\Delta w_i = \lambda * \underbrace{(\text{actual} - \text{predicted})} * x_i$$

causes us to increase the weights!

Perceptron learning

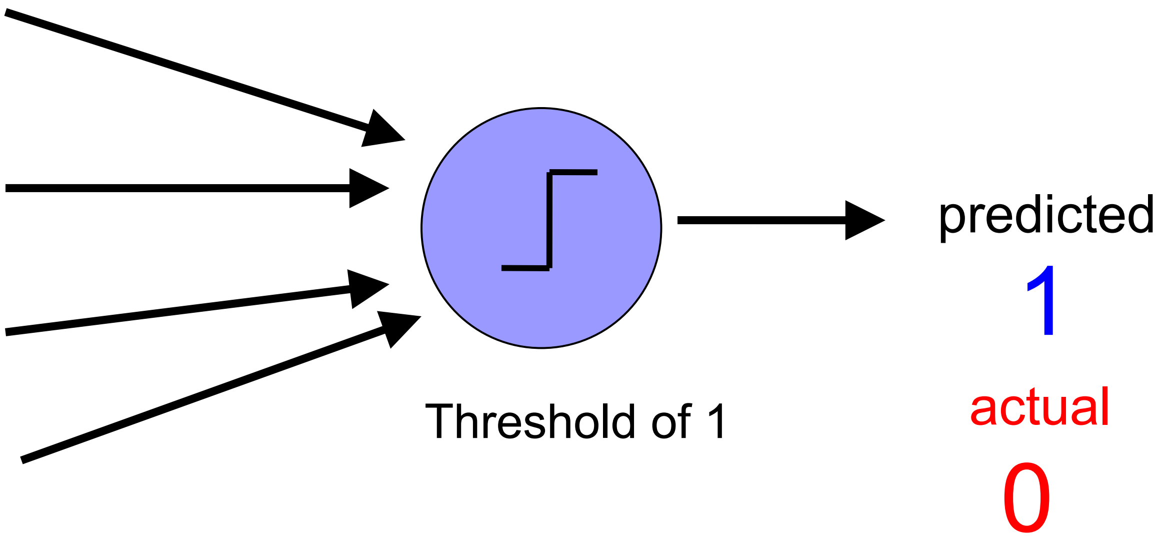


$$w_i = w_i + \Delta wi$$

$$\Delta w_i = \lambda * \underbrace{(\text{actual - predicted})} * x_i$$

What if predicted = 1 and actual = 0?

Perceptron learning

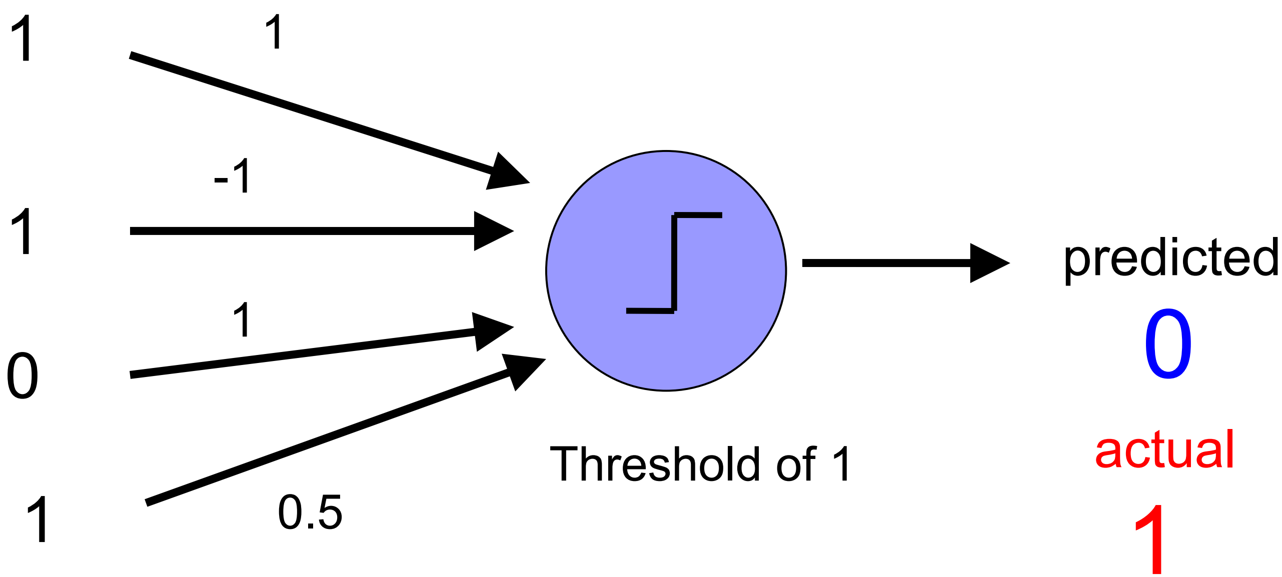


$$w_i = w_i + \Delta wi$$

$$\Delta w_i = \lambda * \underbrace{(\text{actual - predicted})} * x_i$$

We're over the threshold, so want to decrease weights:
actual - predicted = -1

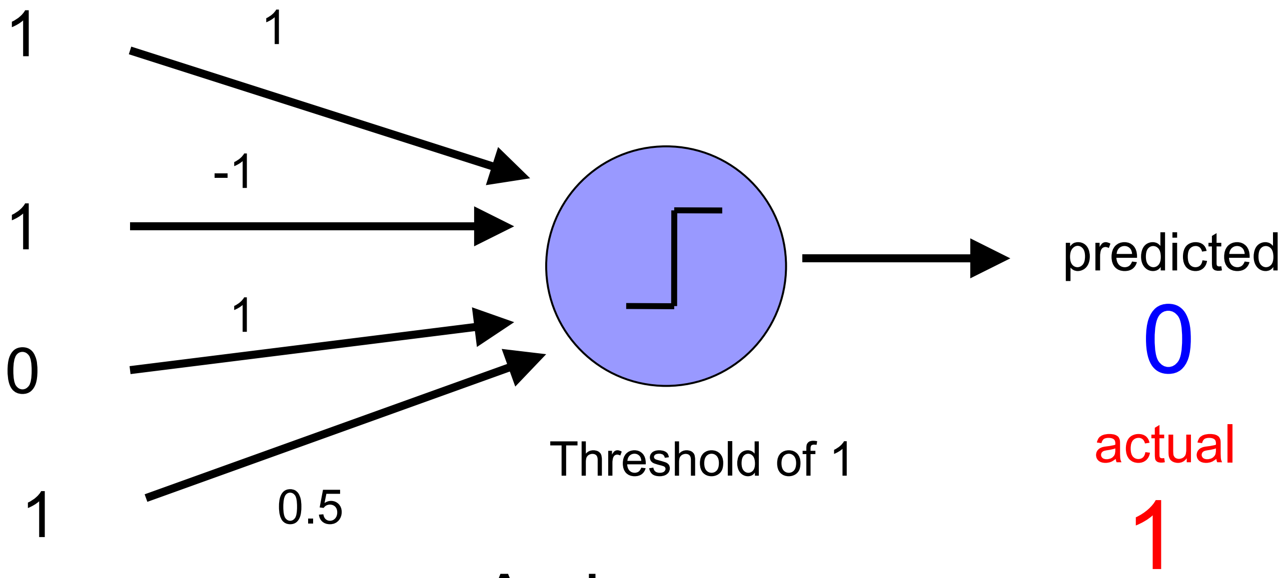
Perceptron learning



$$w_i = w_i + \Delta wi$$

$$\Delta w_i = \lambda * (\text{actual - predicted}) * \underbrace{x_i}_{\text{What does this do?}}$$

Perceptron learning

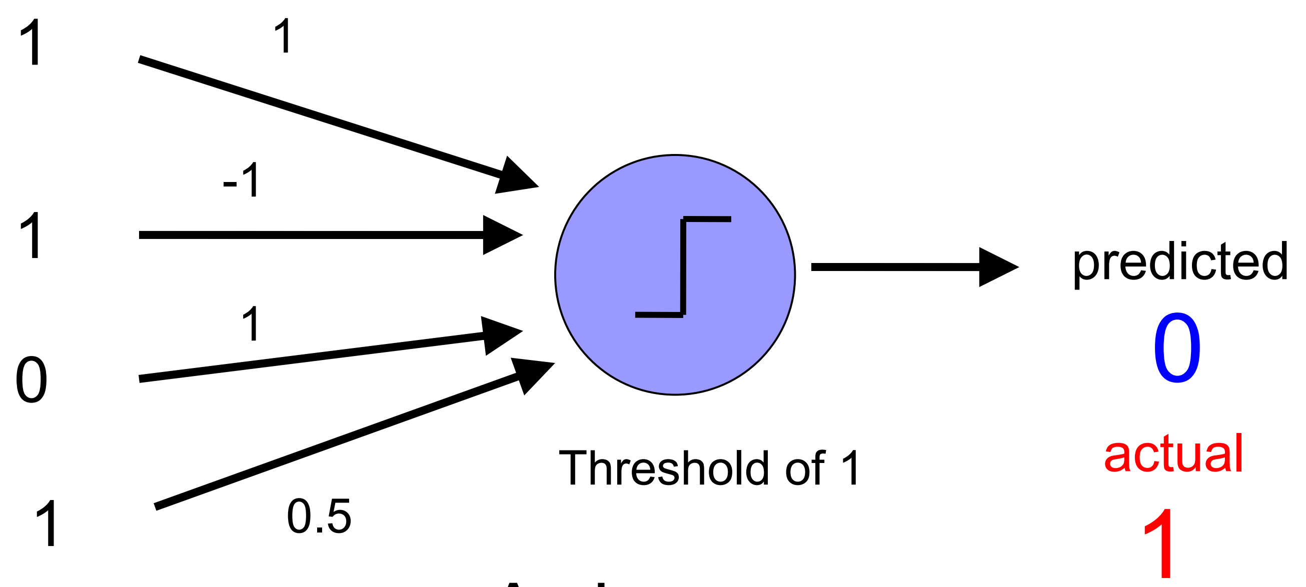


$$w_i = w_i + \Delta wi$$

$$\Delta w_i = \lambda * (\text{actual - predicted}) * x_i$$

Only adjust those weights that actually contributed!

Perceptron learning

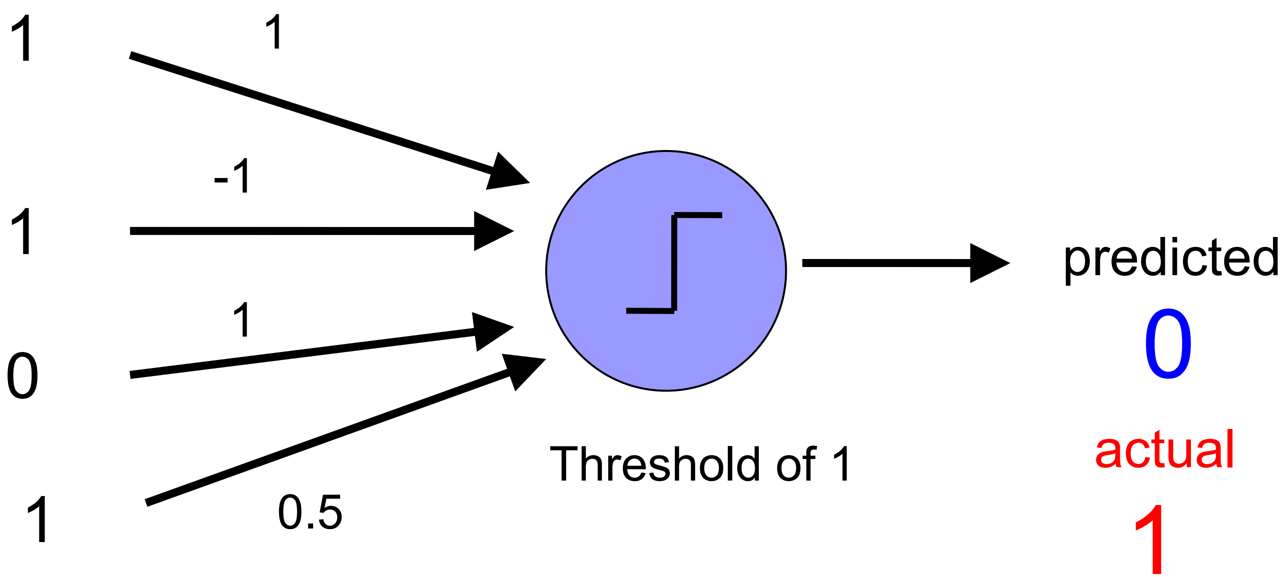


$$w_i = w_i + \Delta wi$$

$$\Delta w_i = \underbrace{\lambda}_{} * (\text{actual - predicted}) * x_i$$

What does this do?

Perceptron learning

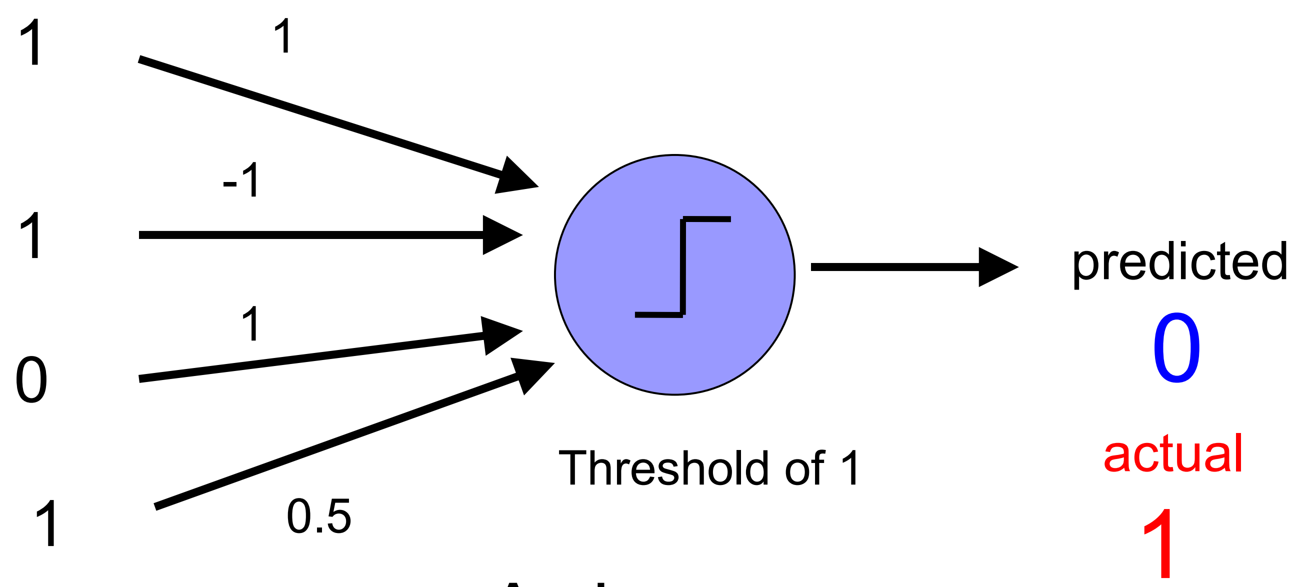


$$w_i = w_i + \Delta wi$$

$$\Delta w_i = \lambda * (\text{actual - predicted}) * x_i$$

"learning rate": value between 0 and 1 (e.g., 0.1)
adjusts how abrupt the changes are to the model

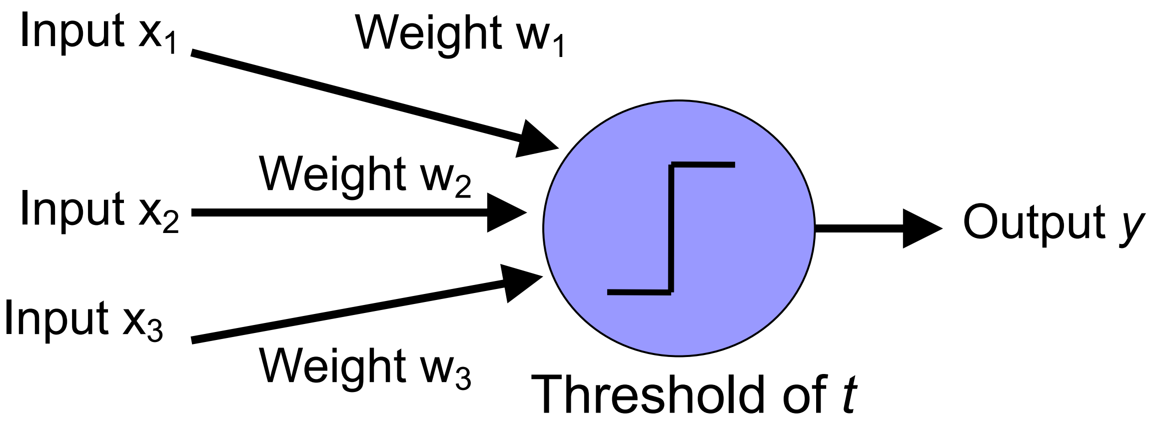
Perceptron learning



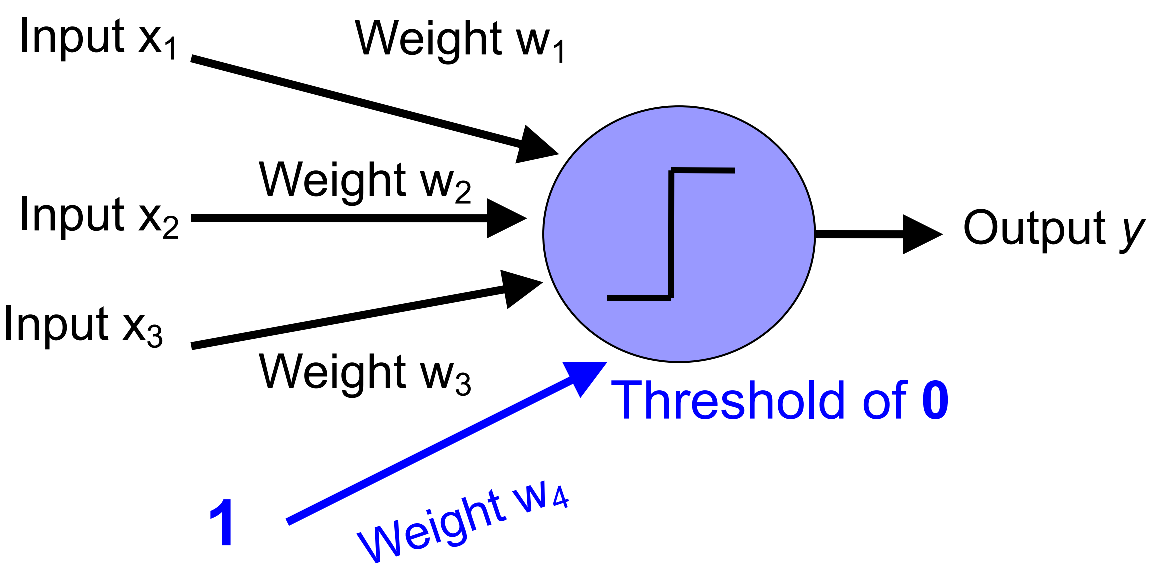
$w_i = w_i + \Delta wi$

$\Delta w_i = \lambda * (\text{actual - predicted}) * x_i$

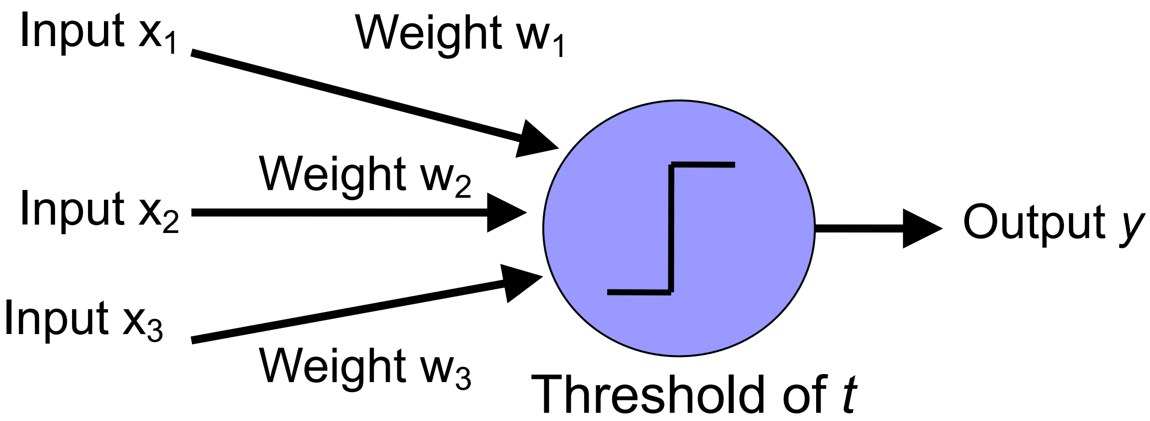
What about the threshold?



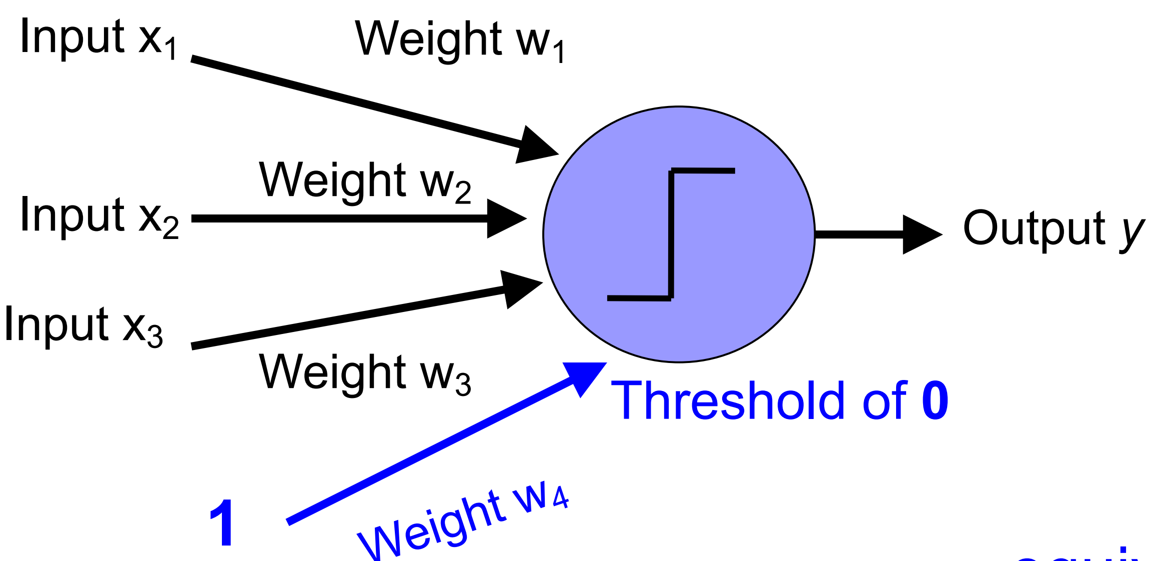
$$1 \text{ if } \sum_{i=1}^{3} w_i x_i \geq t$$



$$1 \text{ if } w_4 + \sum_{i=1}^{3} w_i x_i \geq 0$$



$$1 \text{ if } \sum_{i=1}^{3} w_i x_i \geq t$$



$$1 \text{ if } w_4 + \sum_{i=1}^{3} w_i x_i \geq 0$$

equivalent when $w_4 = -t$

Perceptron learning algorithm

initialize weights of the model randomly

repeat until you get all examples right:

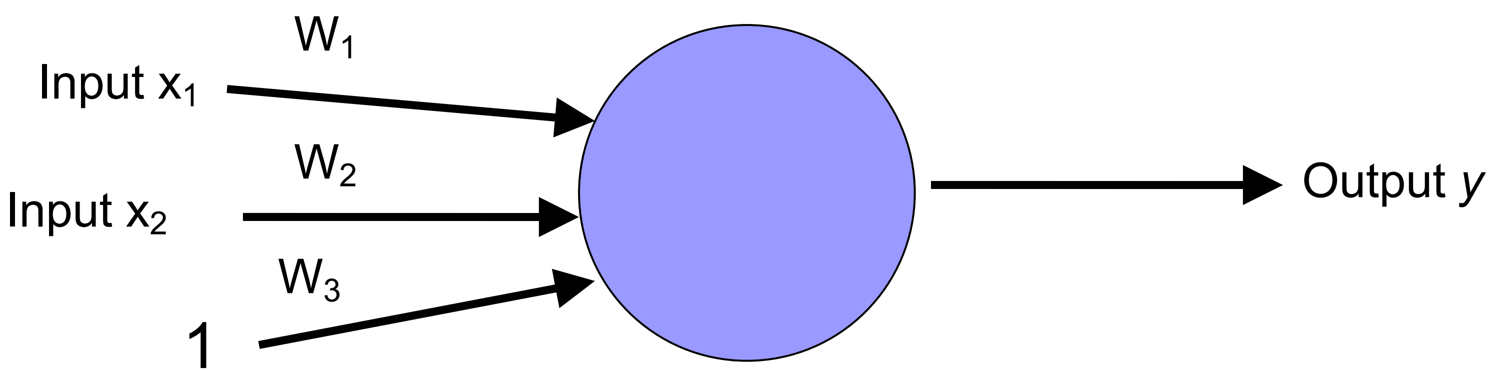
- for each "training" example (*in a random order*):
  - calculate current prediction on the example
  - if *wrong*:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

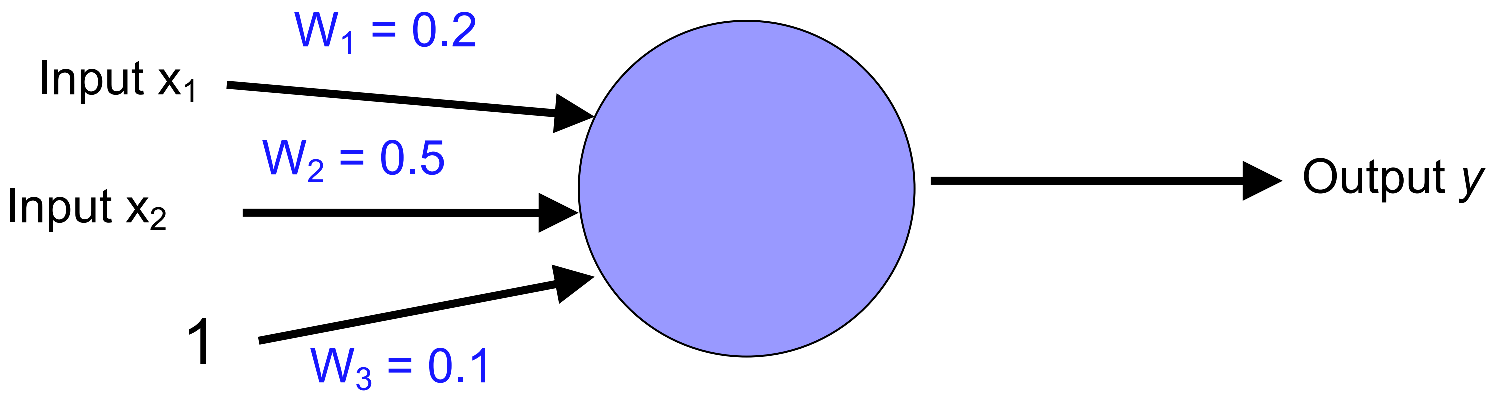
$\lambda = 0.1$

initialize with random weights



x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

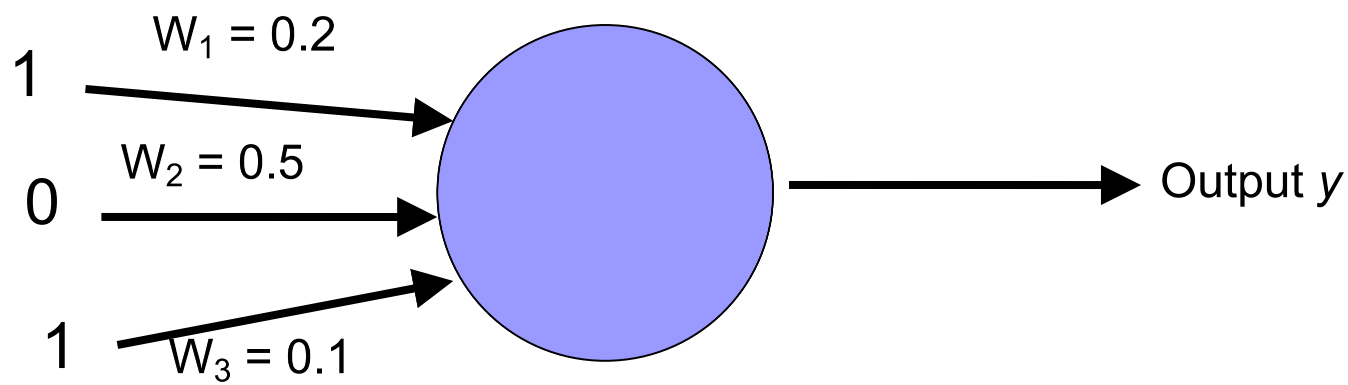


x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual - predicted}) * x_i$$



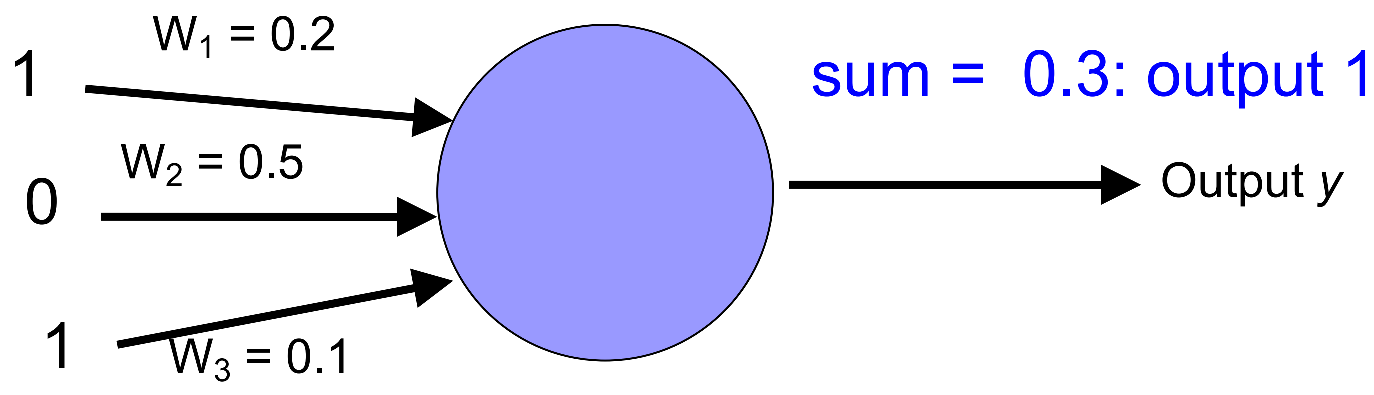
Right or wrong?

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



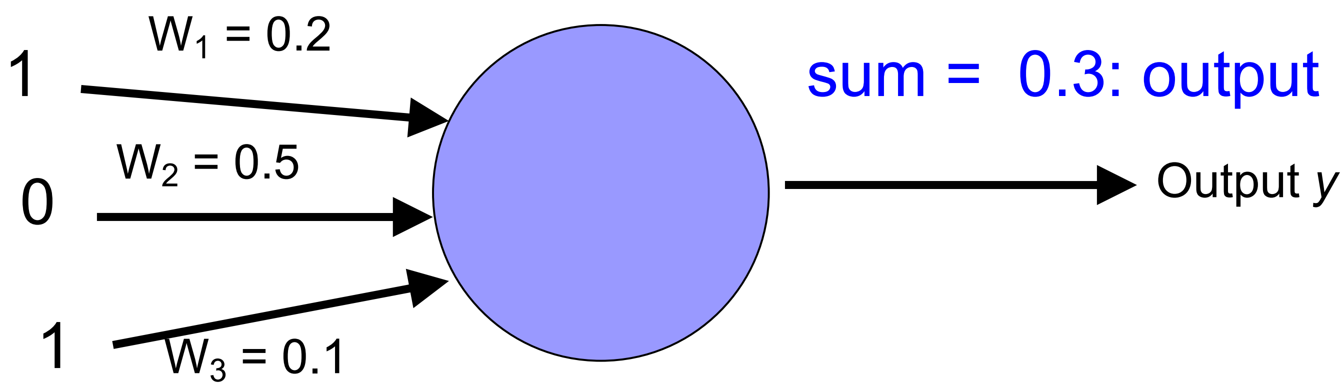
Wrong

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



sum = 0.3: output 1

new weights?

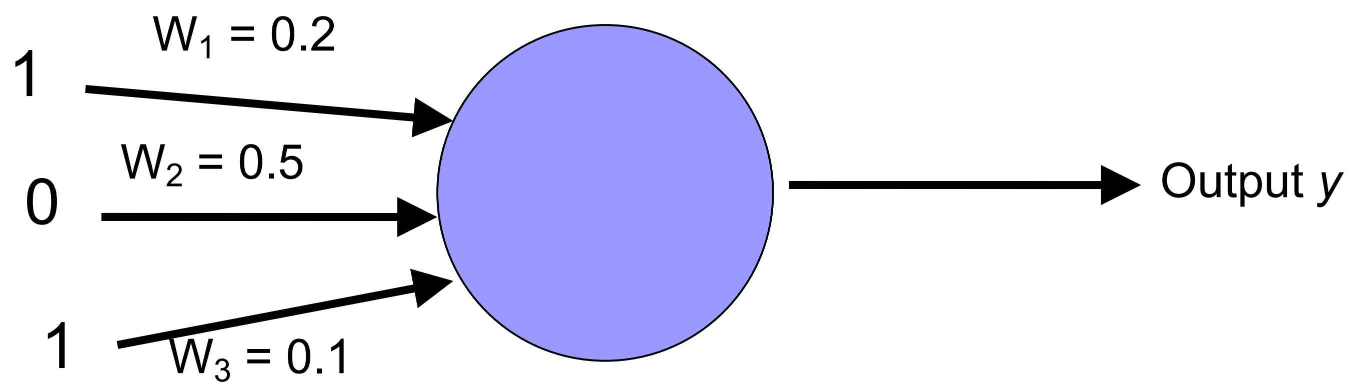
x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$w_i = w_i + \lambda * (\text{actual - predicted}) * x_i$

decrease (0-1=-1) all non-zero x_i by 0.1



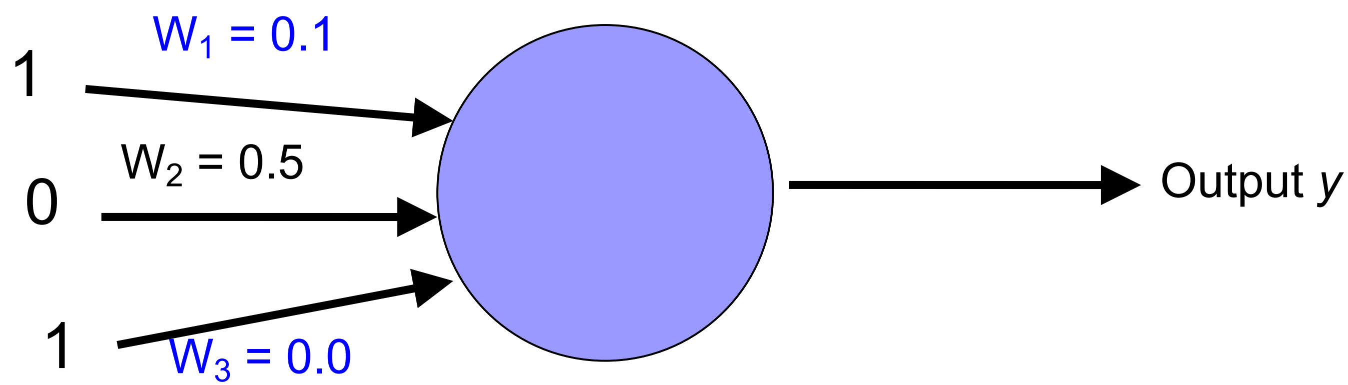
x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$w_i = w_i + \lambda * (\text{actual - predicted}) * x_i$

decrease (0-1=-1) all non-zero x_i by 0.1

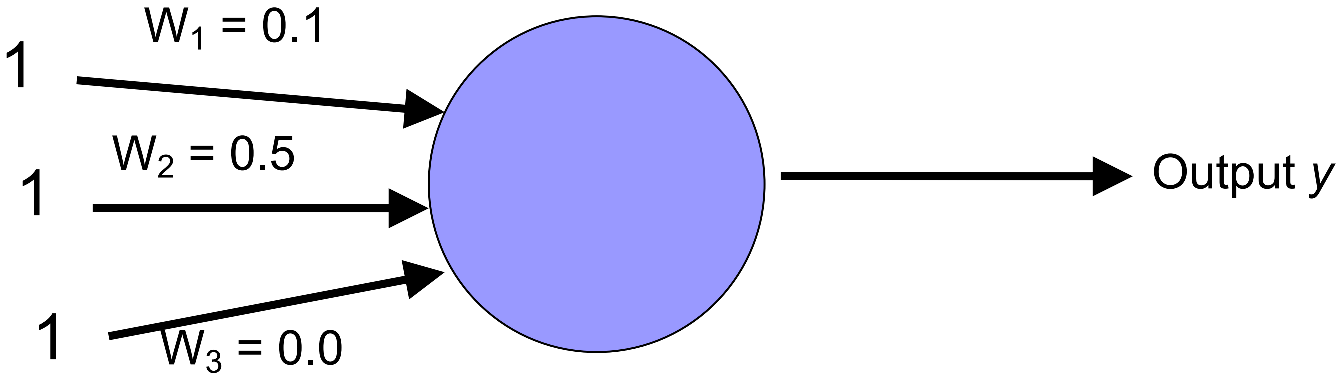


x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual - predicted}) * x_i$$



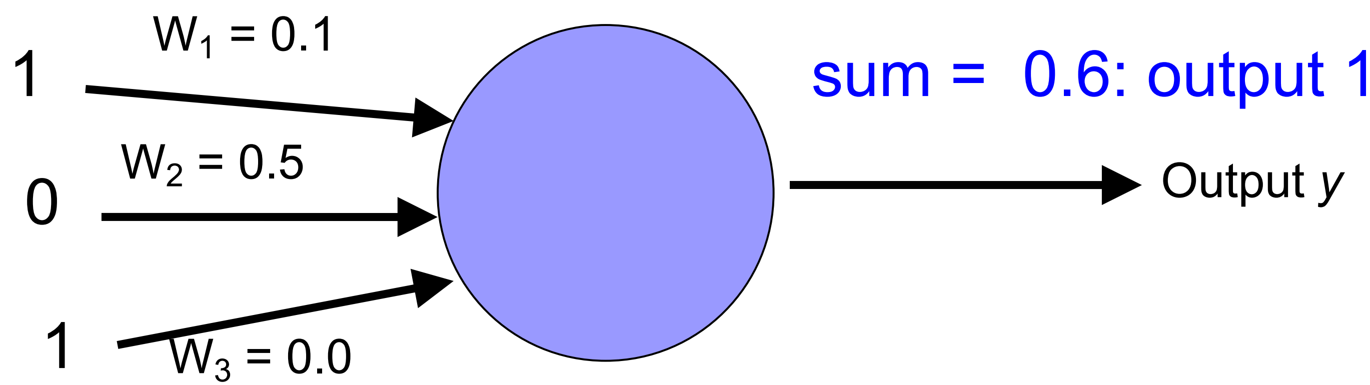
Right or wrong?

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



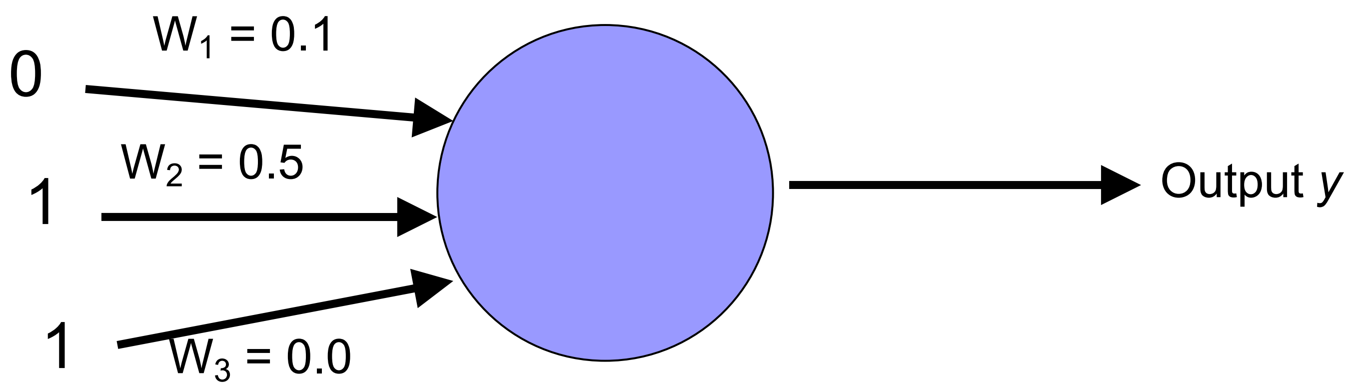
Right. No update!

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



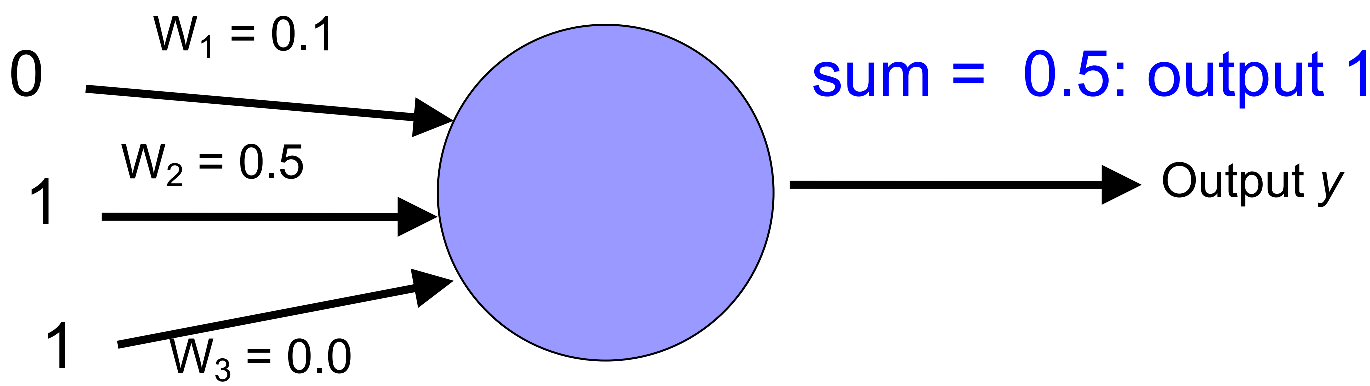
Right or wrong?

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



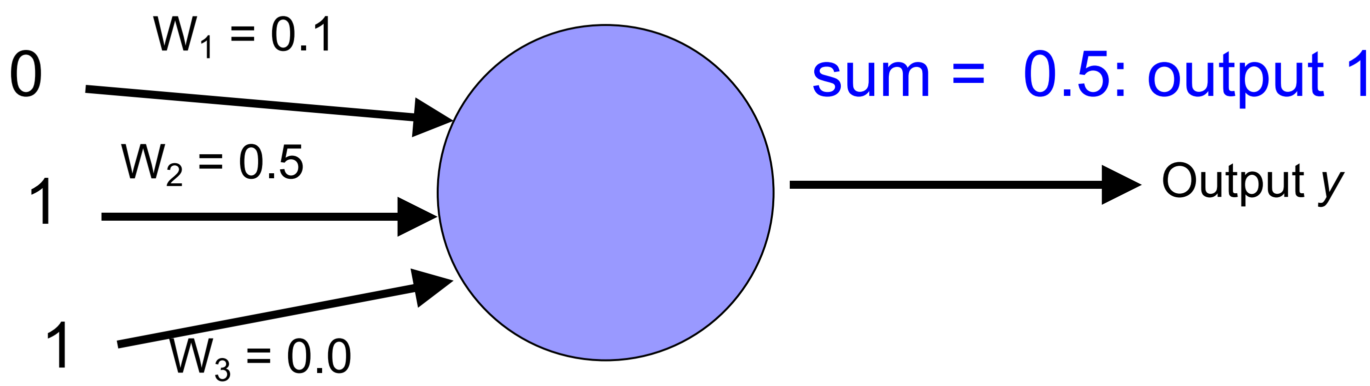
Wrong

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



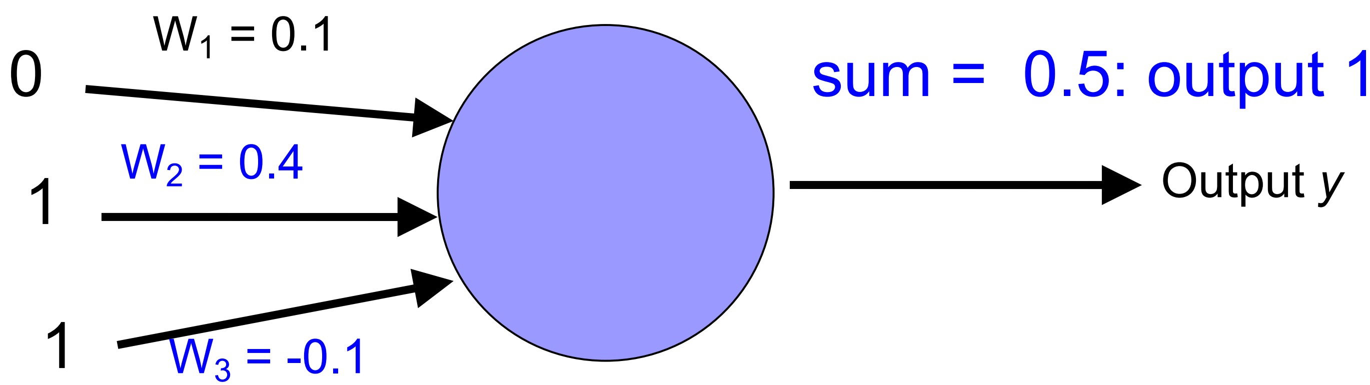
new weights?

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual - predicted}) * x_i$$



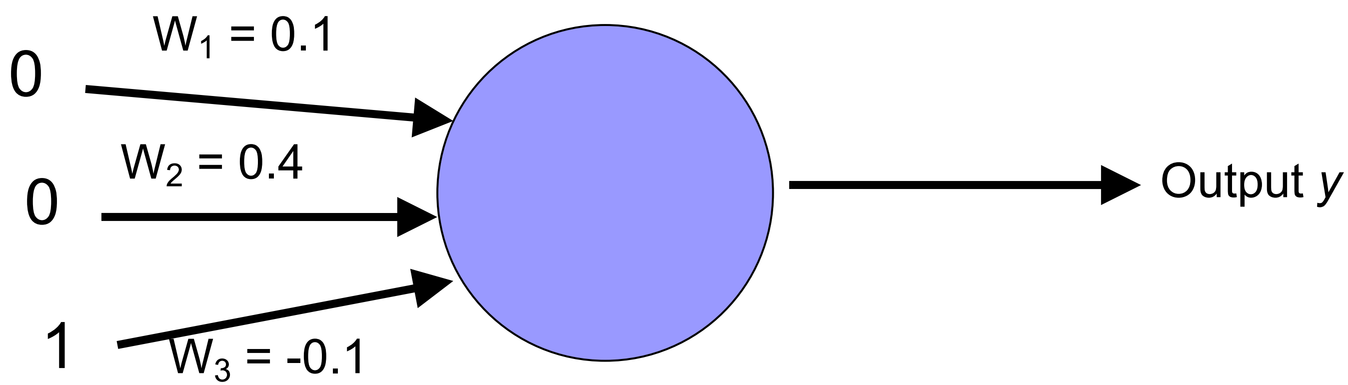
decrease (0-1=-1) all non-zero x_i by 0.1

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



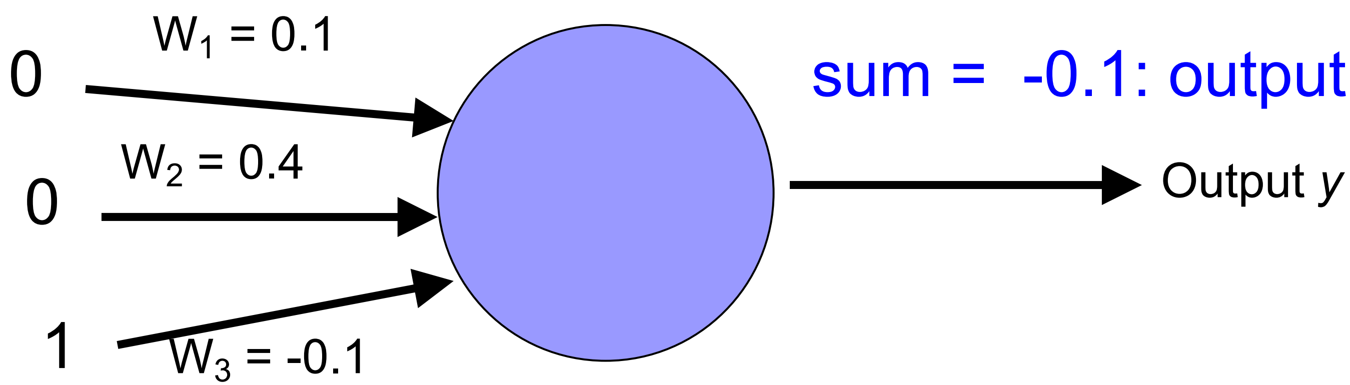
Right or wrong?

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual - predicted}) * x_i$$



sum = -0.1: output 0

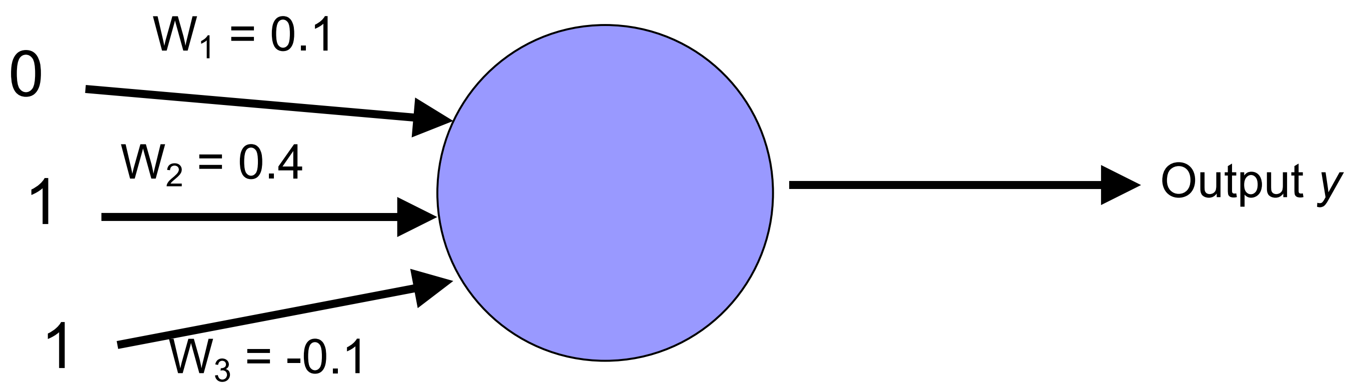
Right. No update!

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



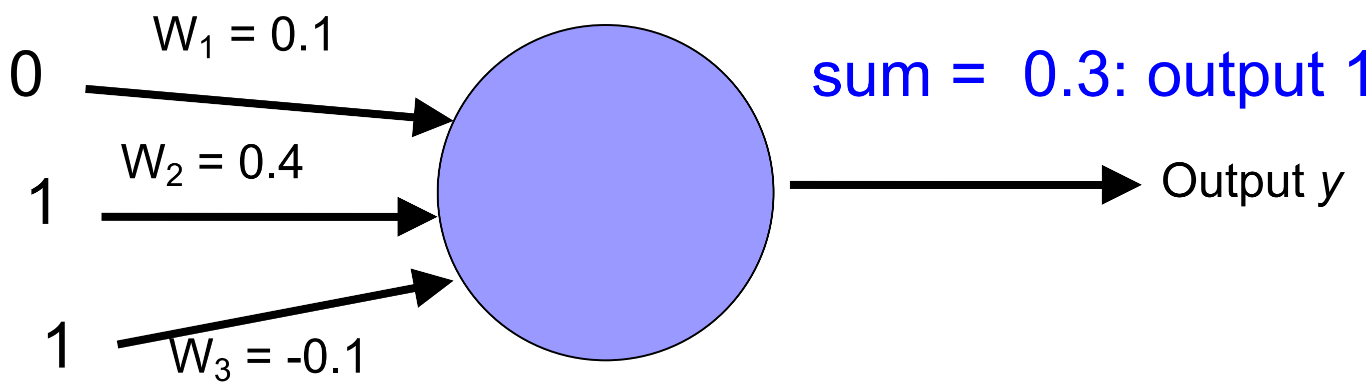
Right or wrong?

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



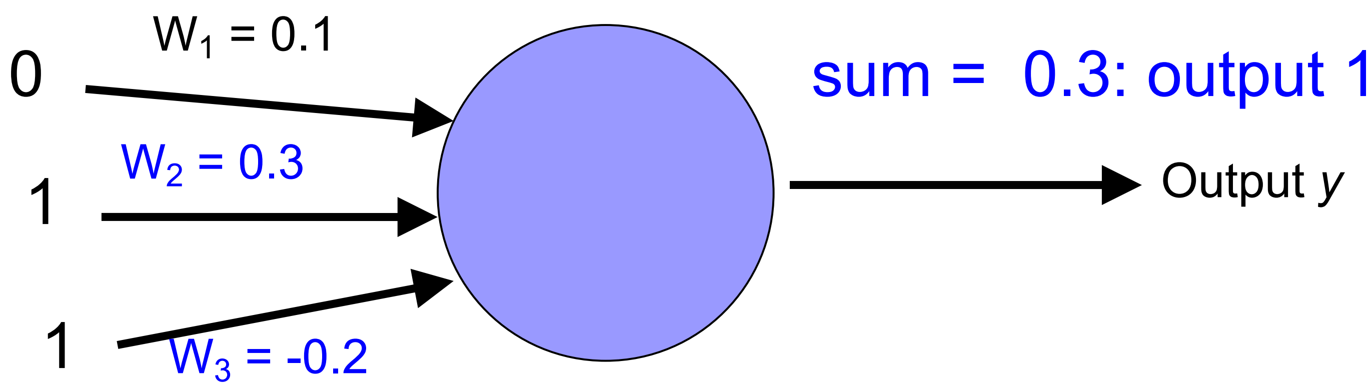
Wrong

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



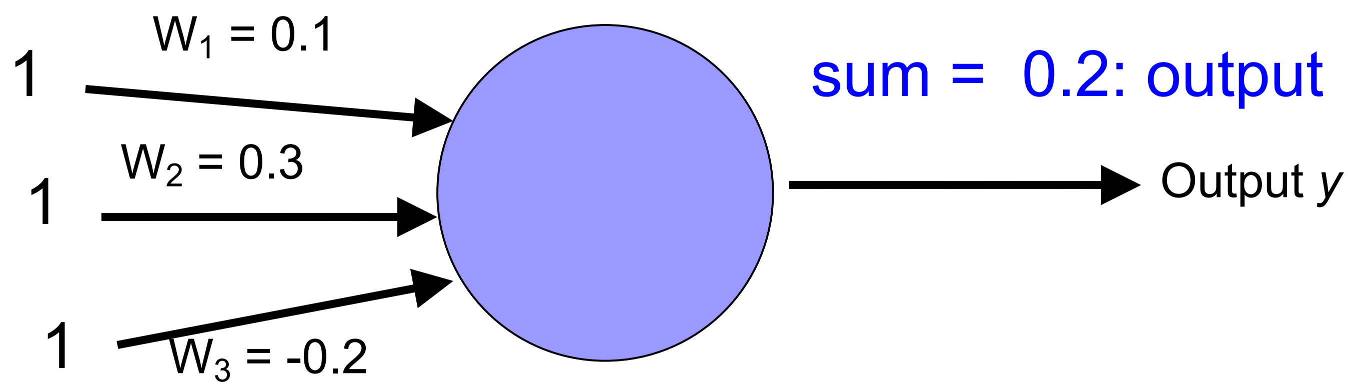
decrease (0-1=-1) all non-zero x_i by 0.1

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual - predicted}) * x_i$$



sum = 0.2: output 1

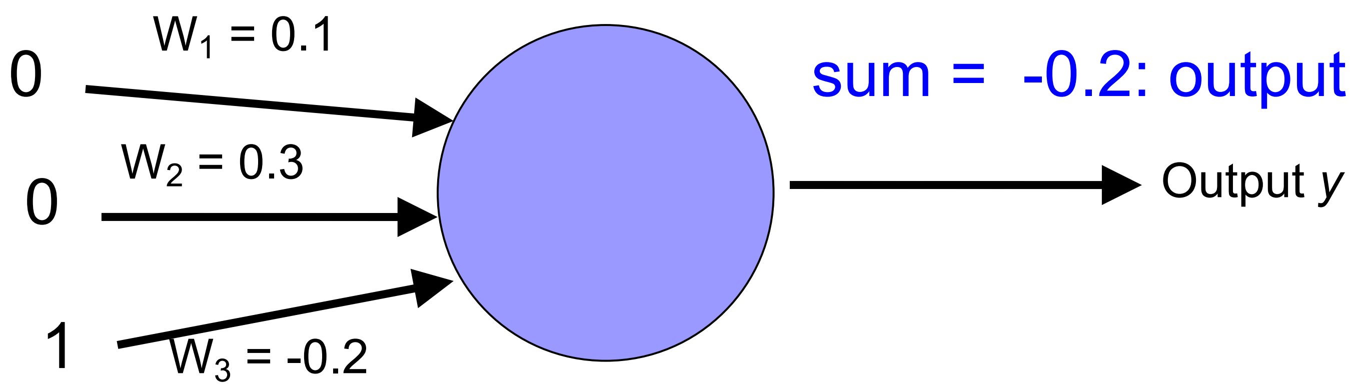
Right. No update!

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



sum = -0.2: output 0

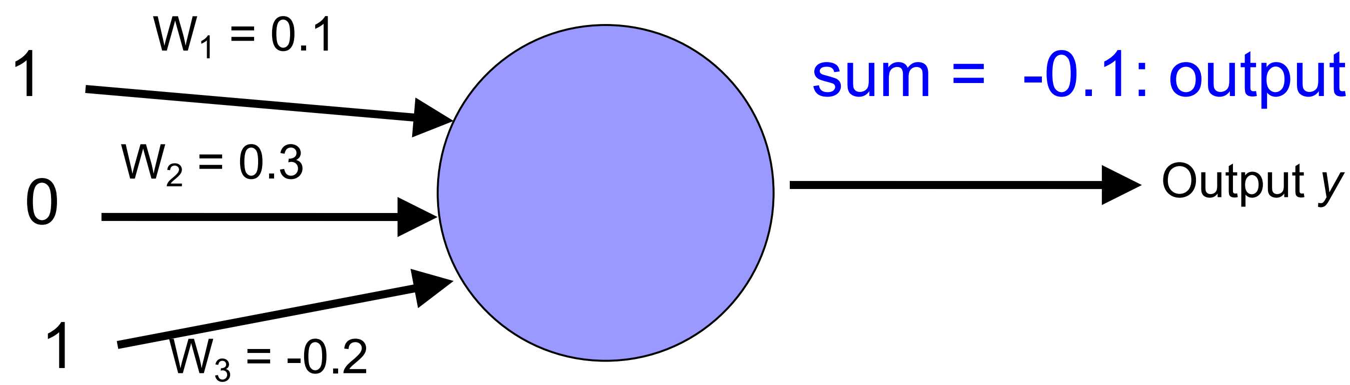
Right. No update!

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



sum = -0.1: output 0

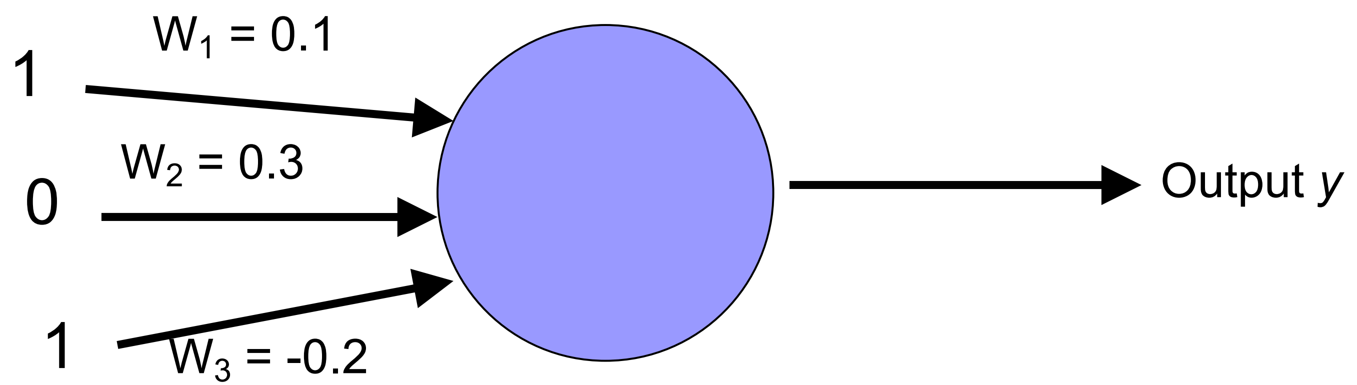
Right. No update!

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual - predicted}) * x_i$$



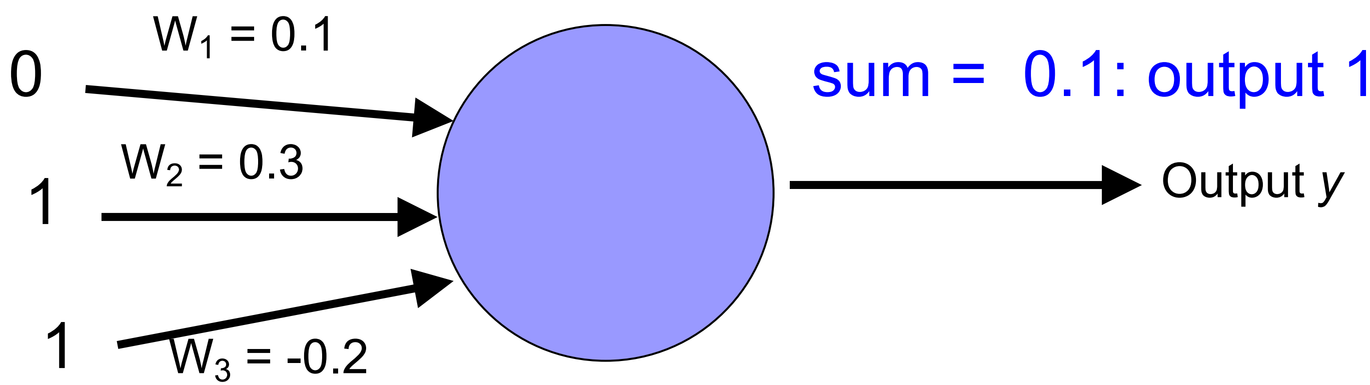
Are they all right?

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



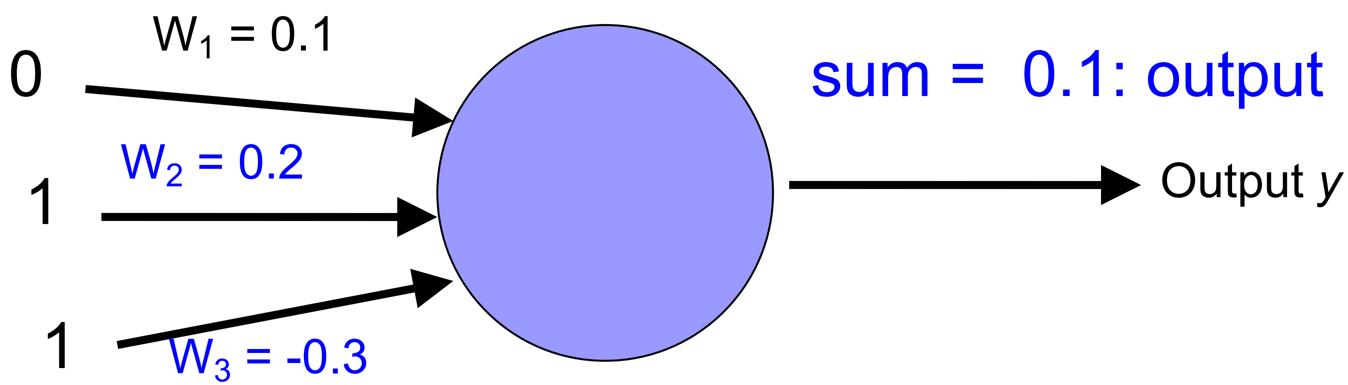
Wrong

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual} - \text{predicted}) * x_i$$



sum = 0.1: output 1

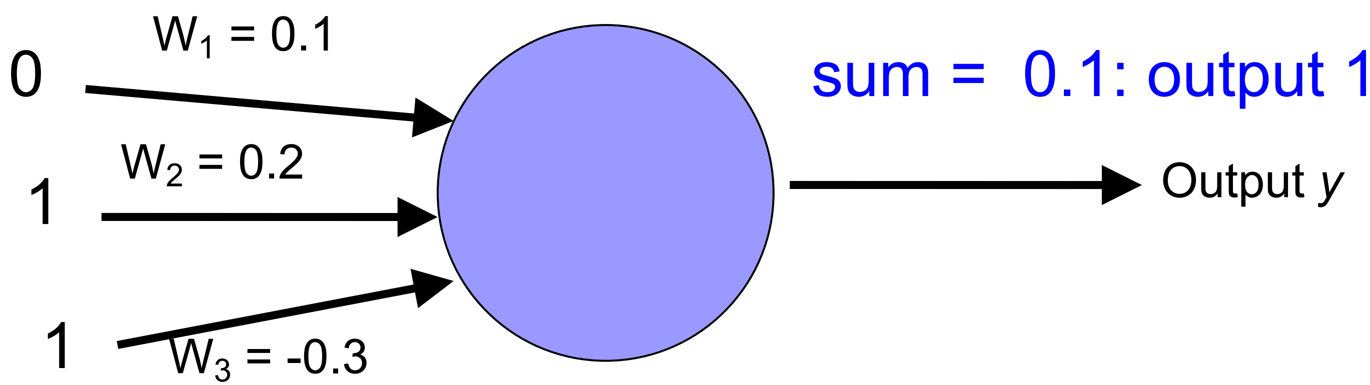
decrease (0-1=-1) all non-zero x_i by 0.1

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual - predicted}) * x_i$$



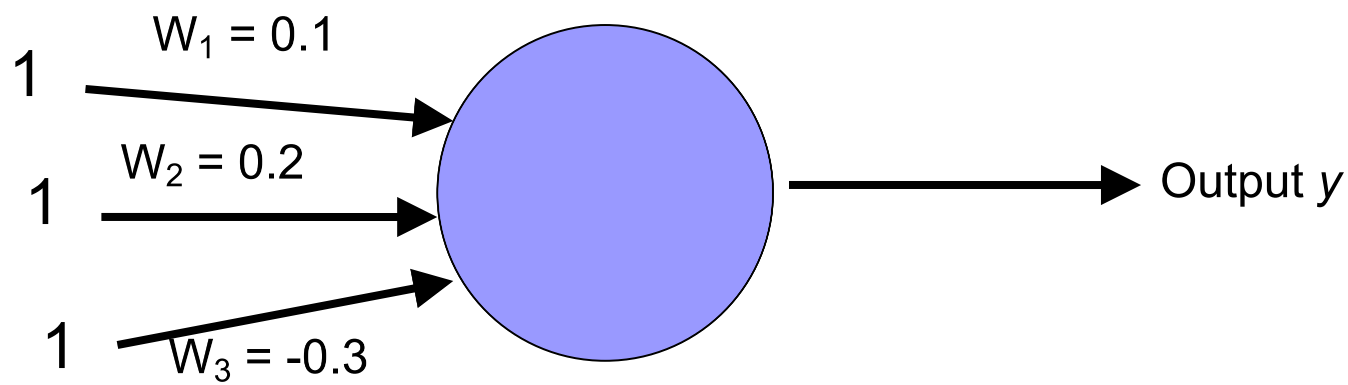
Are they all right?

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

$\lambda = 0.1$

if wrong:

$$w_i = w_i + \lambda * (\text{actual - predicted}) * x_i$$



We've learned AND!

Perceptron learning

A few missing details, but not much more than this

Keeps adjusting weights as long as it makes mistakes

If the training data is linearly separable, the perceptron learning algorithm is guaranteed to converge to the "correct" solution (where it gets all examples right)