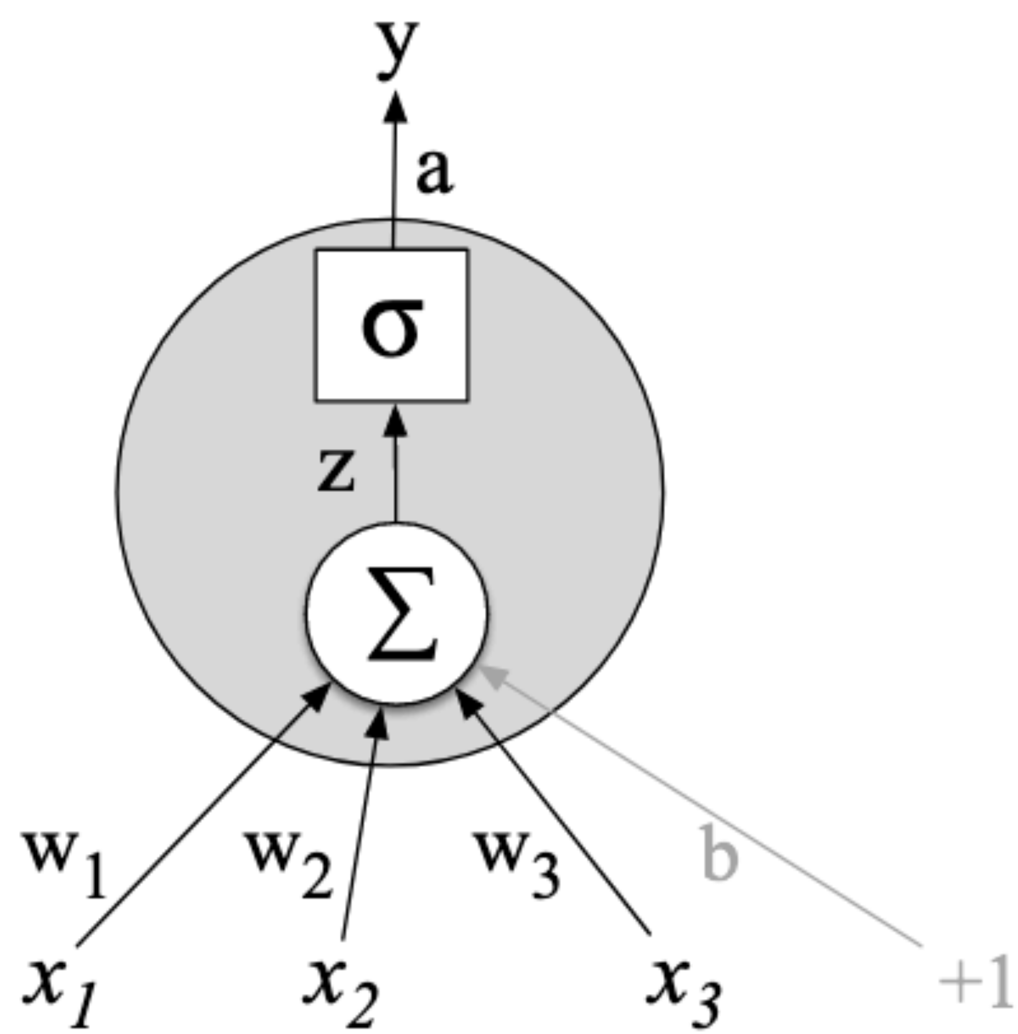
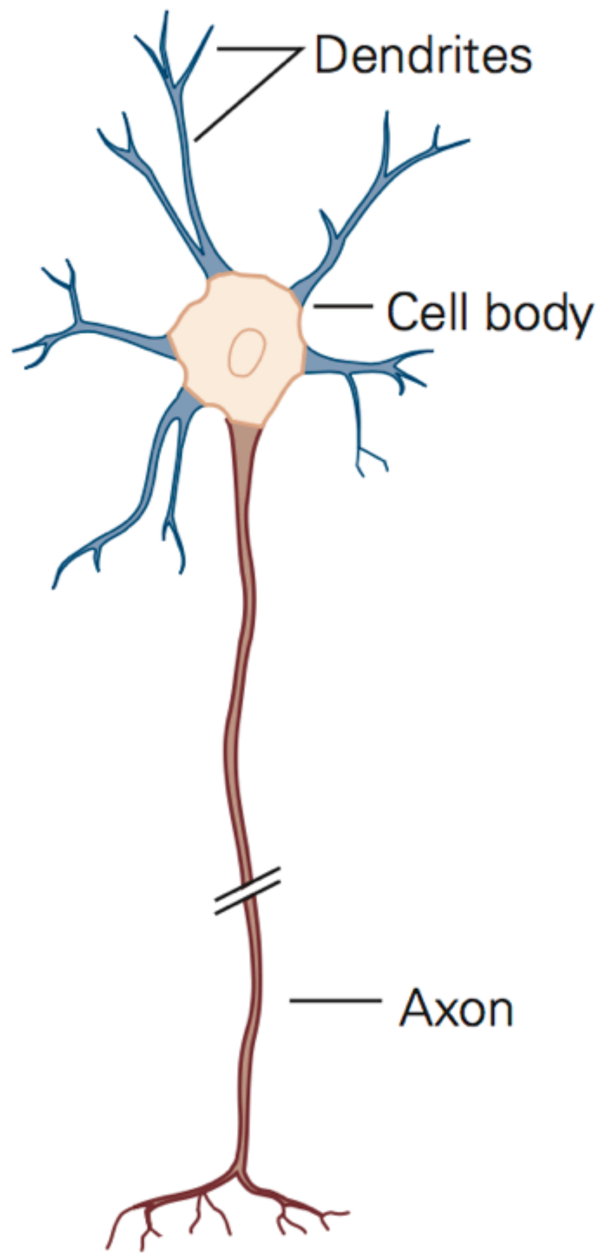
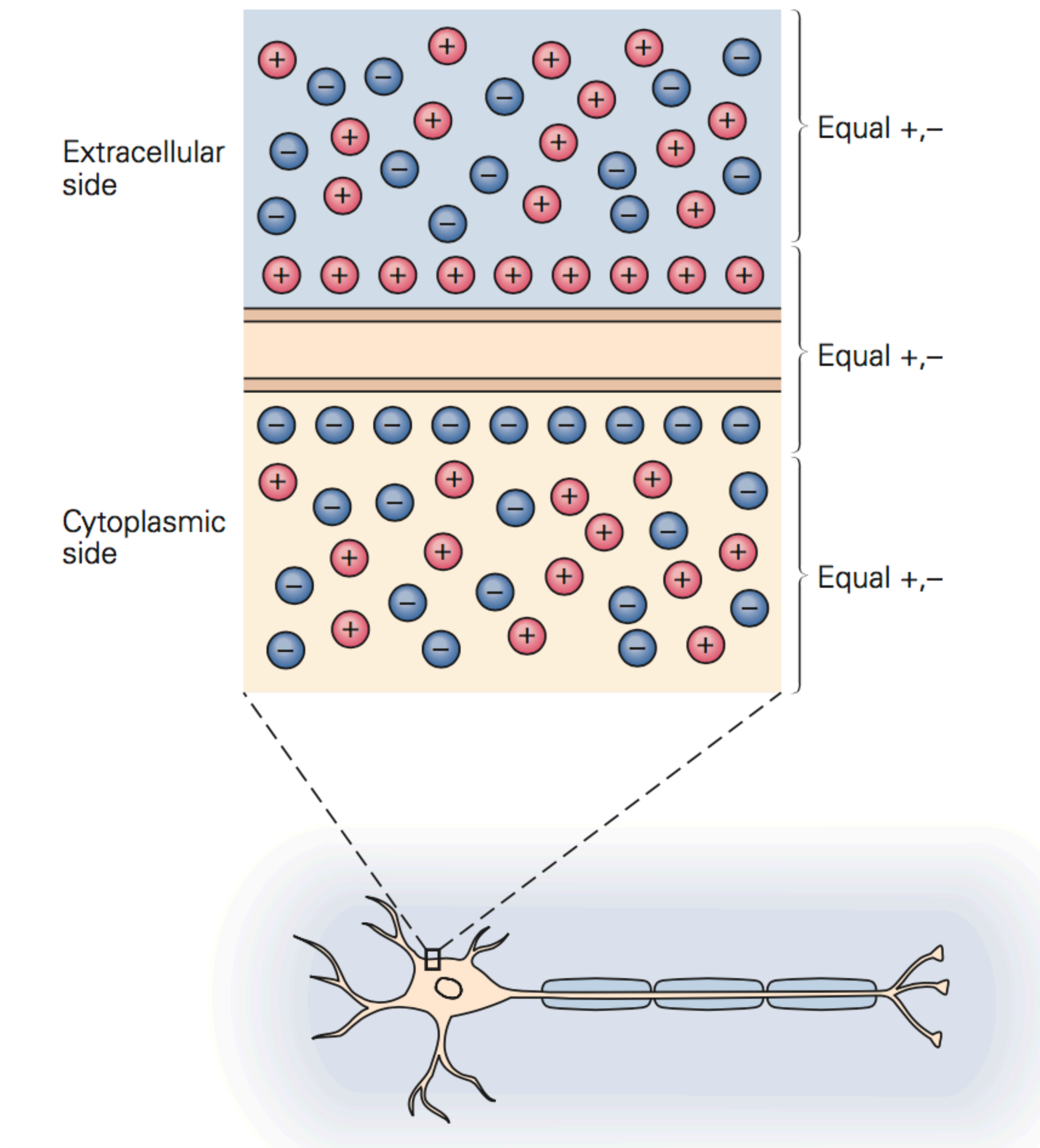
Artificial neuron



Neural inspiration

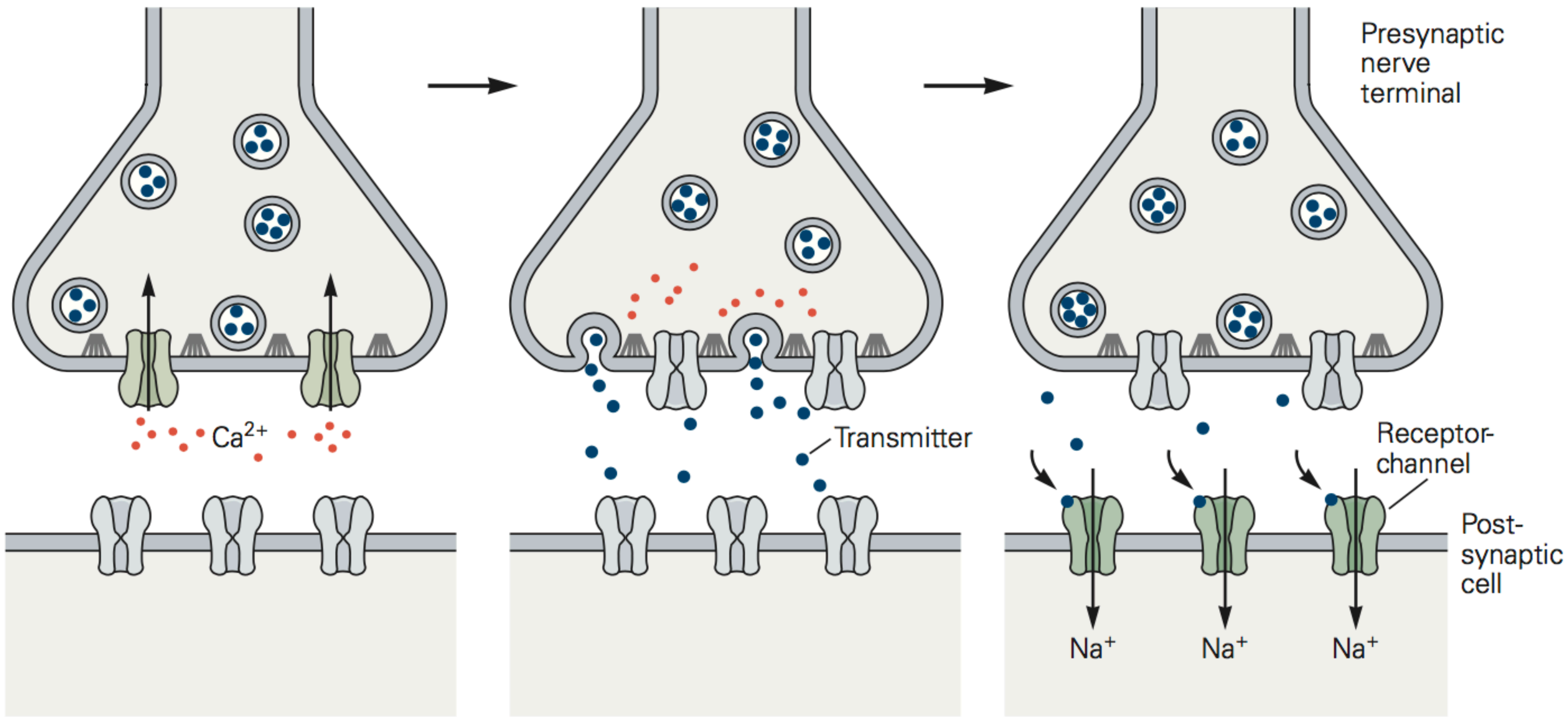


Neural inspiration

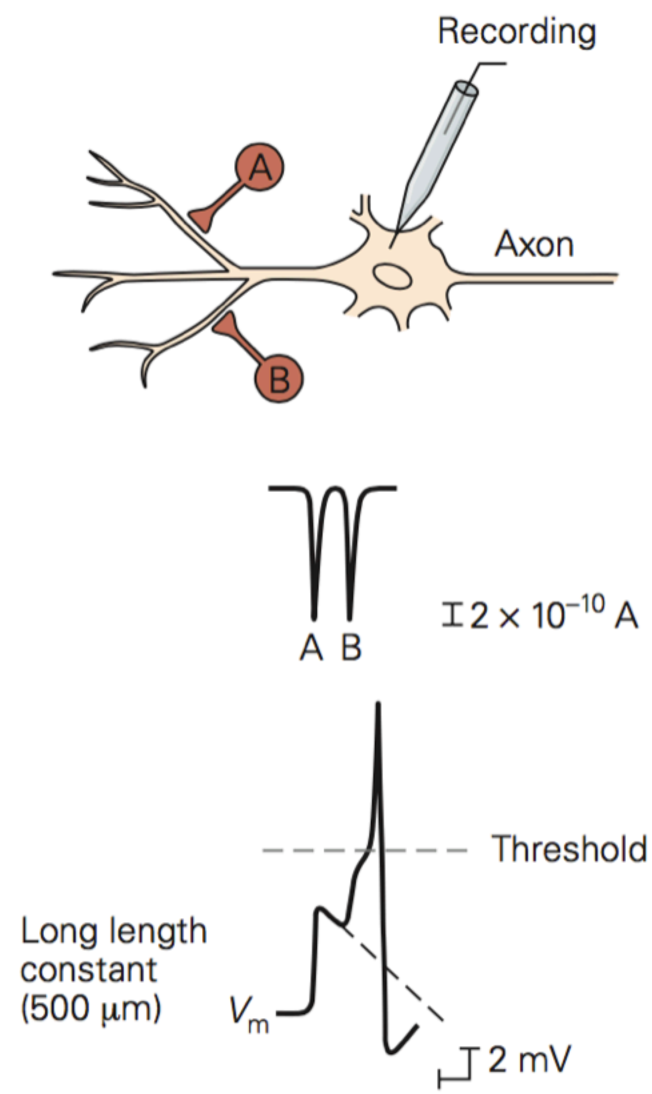


Synapses

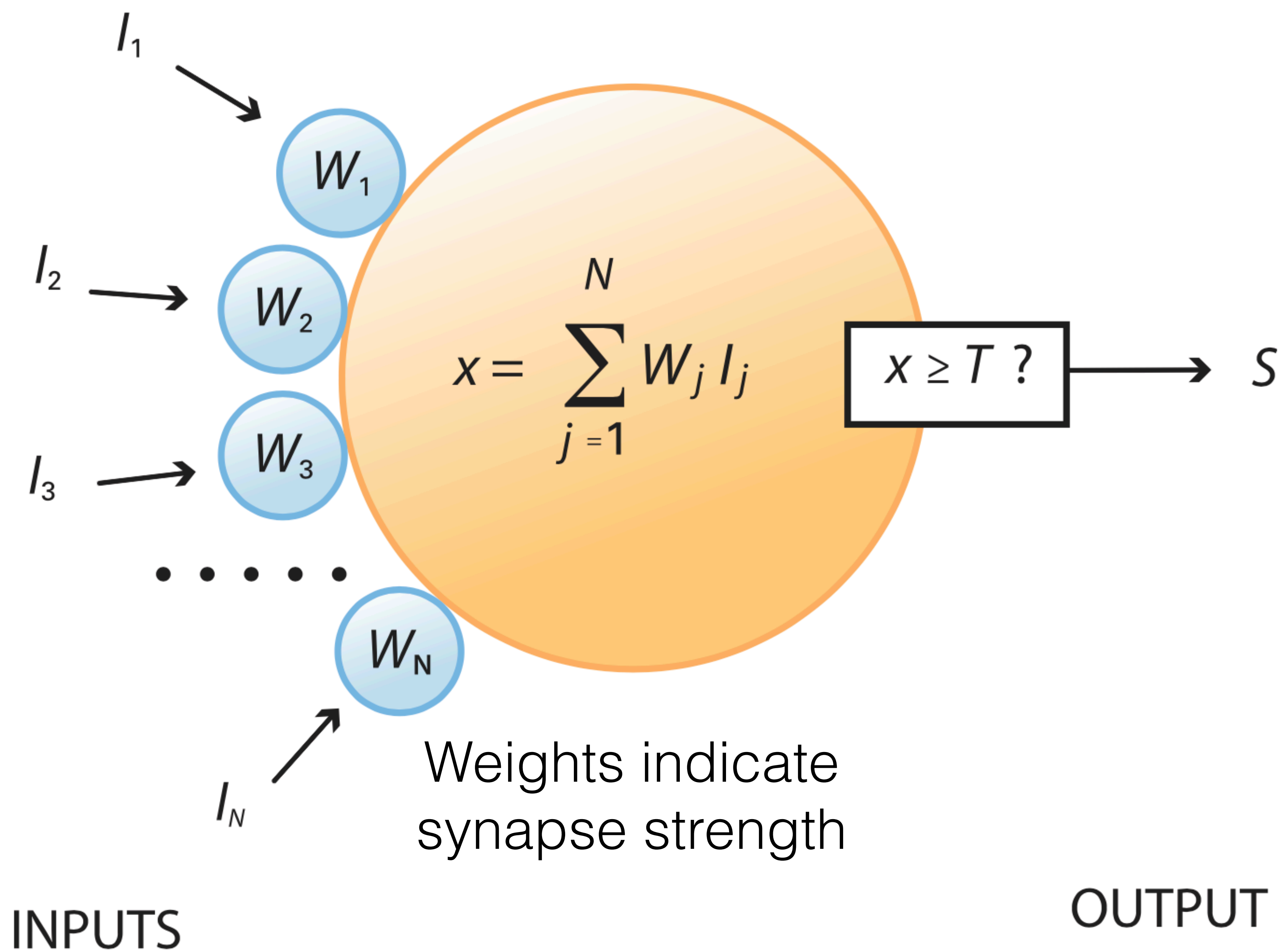
- On the input side, information is usually transmitted chemically, using a neurotransmitter (such as dopamine or serotonin)



Neural inspiration

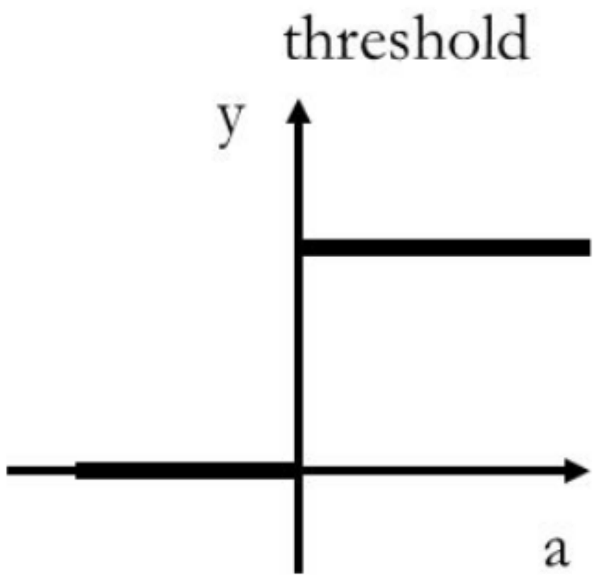


Perceptron (idealization of a biological neuron)



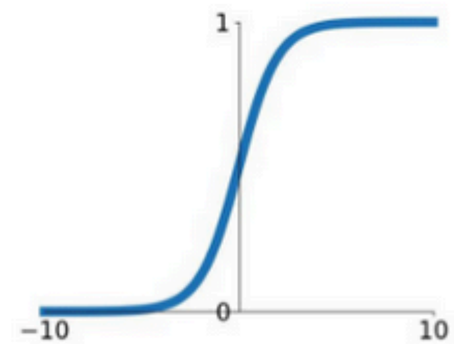
Weights indicate synapse strength

ANN activation functions



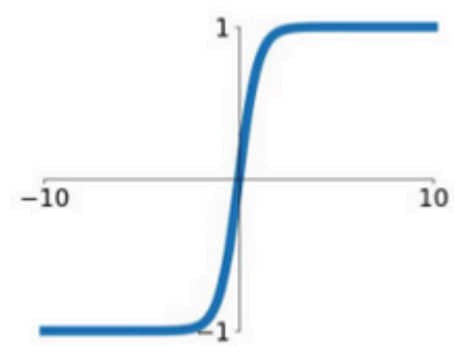
Sigmoid

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



tanh

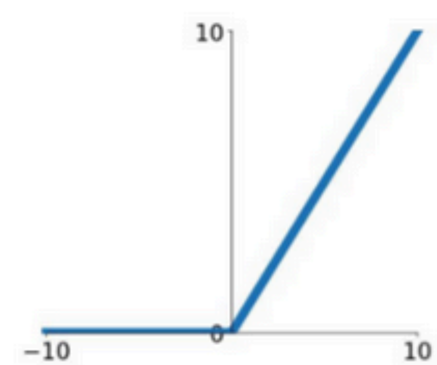
$$\tanh(x)$$



$$\frac{e^x - e^{-x}}{e^x + e^{-x}}$$

ReLU

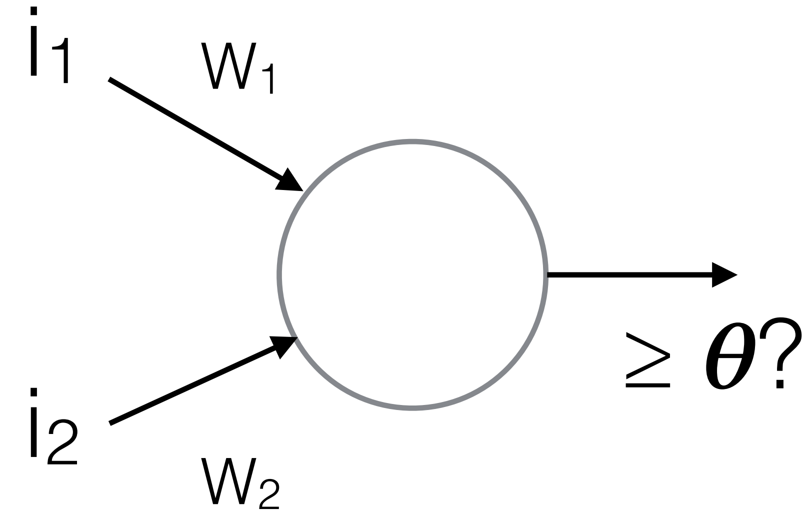
$$\max(0, x)$$



credit

Logical functions using perceptrons

- Suppose that our input features indicate light at a two points in space (0 = no light; 1 = light)



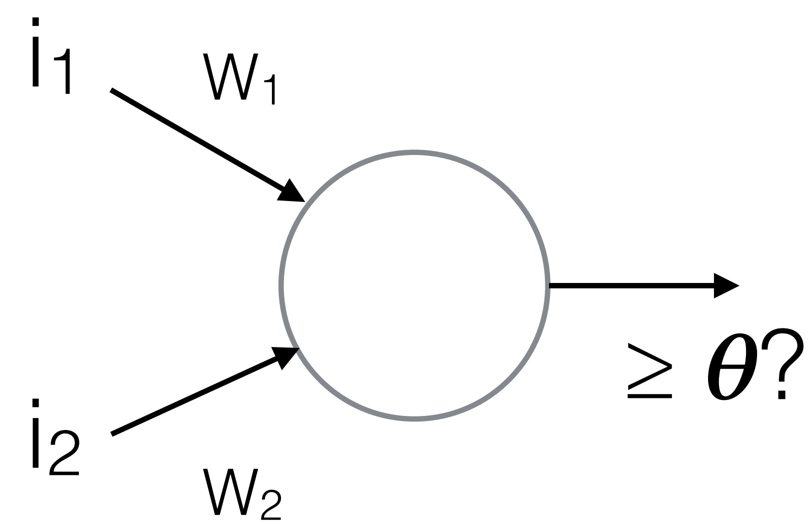
- How can we build a perceptron that detects when there is light in both locations?

$$w_1 = 1, w_2 = 1, \theta = 2$$

i_1	i_2	$w_1i_1+w_2i_2$
0	0	0
0	1	1
1	0	1
1	1	2

Limitations of perceptrons

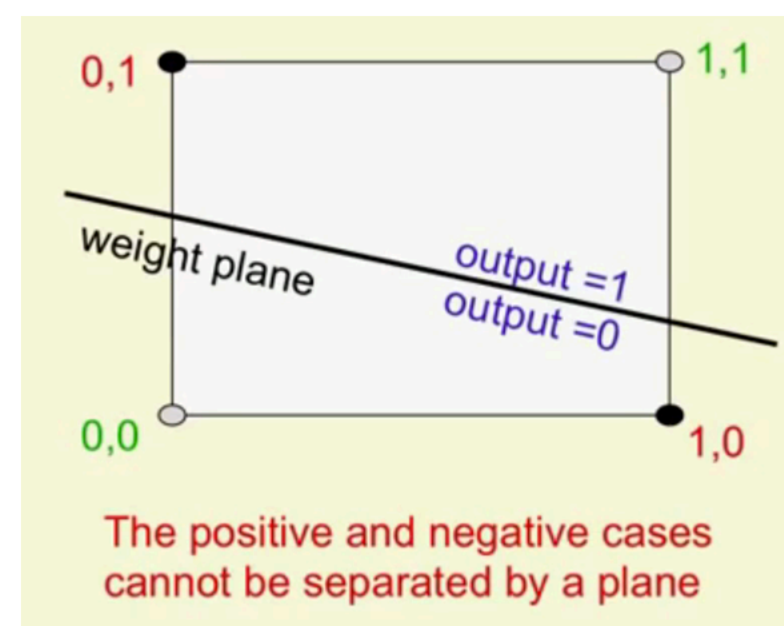
- Can we build a perceptron that fires when the two pixels have the same value ($i_1 = i_2$)?



Positive: (1, 1) (0, 0)

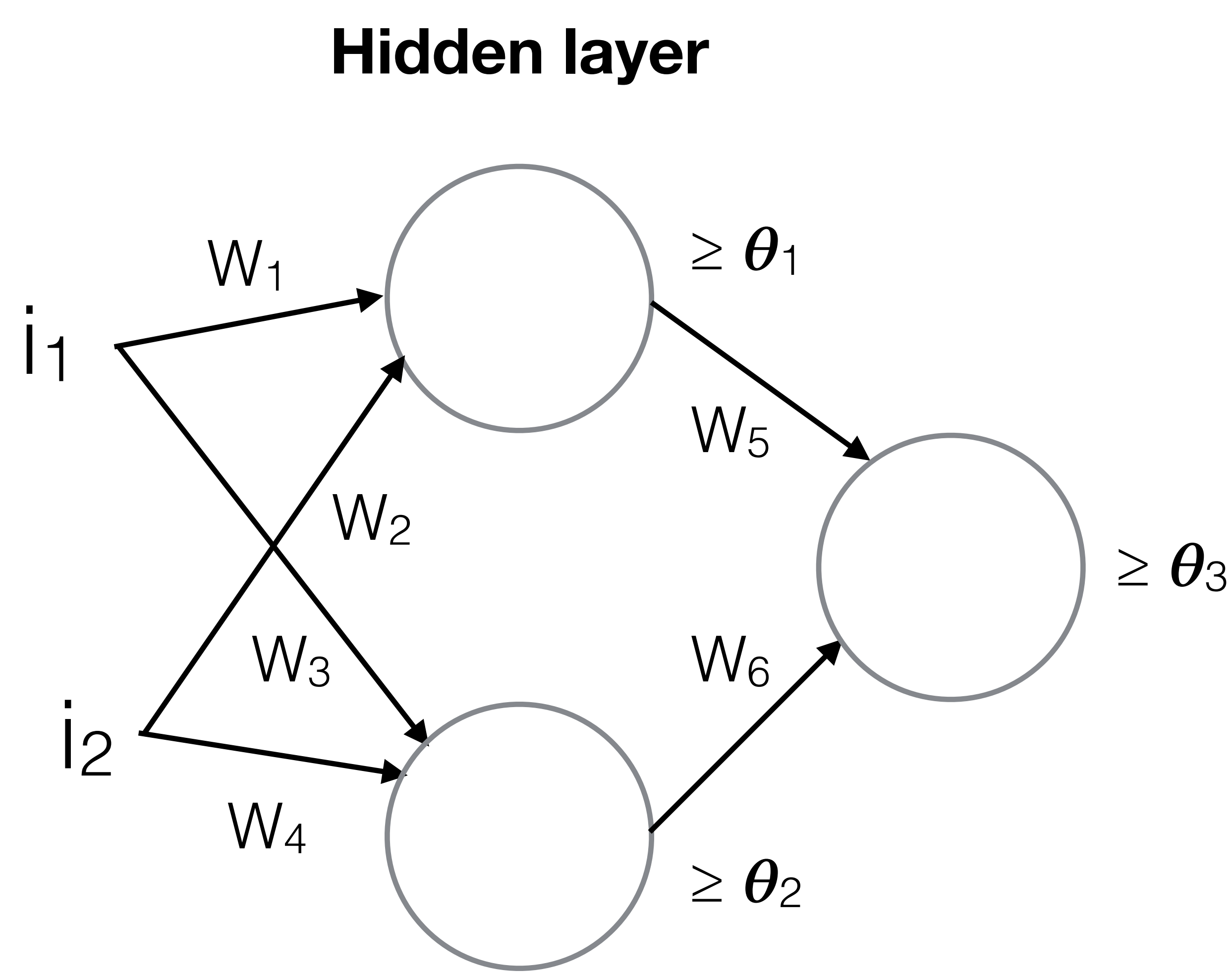
$$w_1 + w_2 \geq \theta, \quad 0 \geq \theta$$
$$w_1 < \theta, \quad w_2 < \theta$$

Negative: (1, 0) (0, 1)



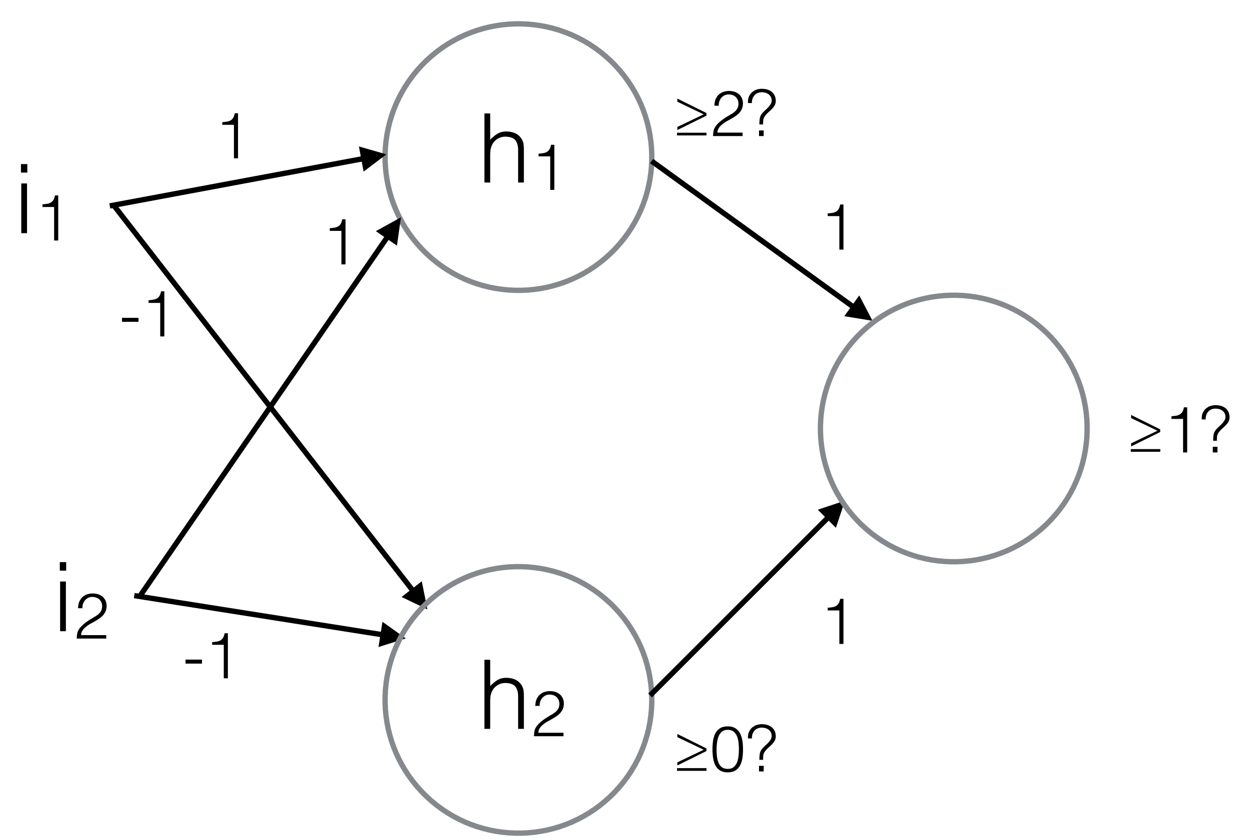
Multilayer perceptron

- Fire when the two pixels have the same value ($i_1 = i_2$)
- How can we set the weights?



Representation learning

- Fire when the two pixels have the same value ($i_1 = i_2$)

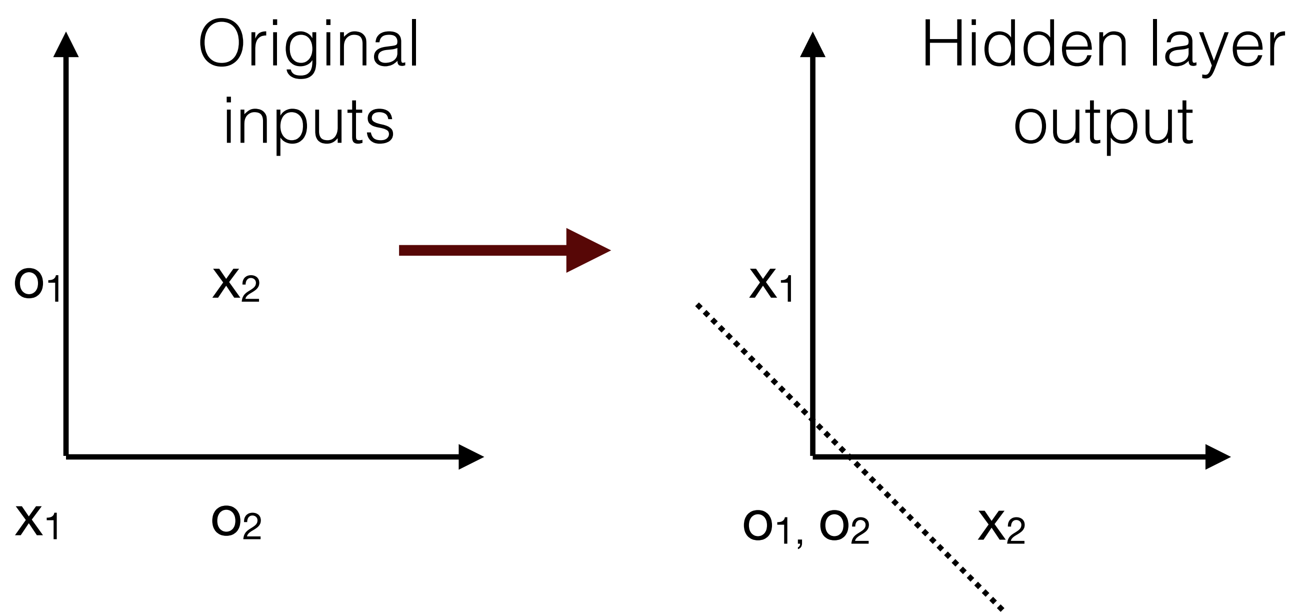


			Hidden layer input		Hidden layer output		
	i_1	i_2	h_1	h_2	h_1	h_2	o
x_1	0	0	0	0	0	1	1
o_1	0	1	1	-1	0	0	0
o_2	1	0	1	-1	0	0	0
x_2	1	1	2	-2	1	0	1

(for x_1 and x_2 the correct output is 1;
for o_1 and o_2 the correct output is 0)

Representation learning

- Recode the input: the hidden layer representations are now linearly separable



Not linearly separable

Linearly separable

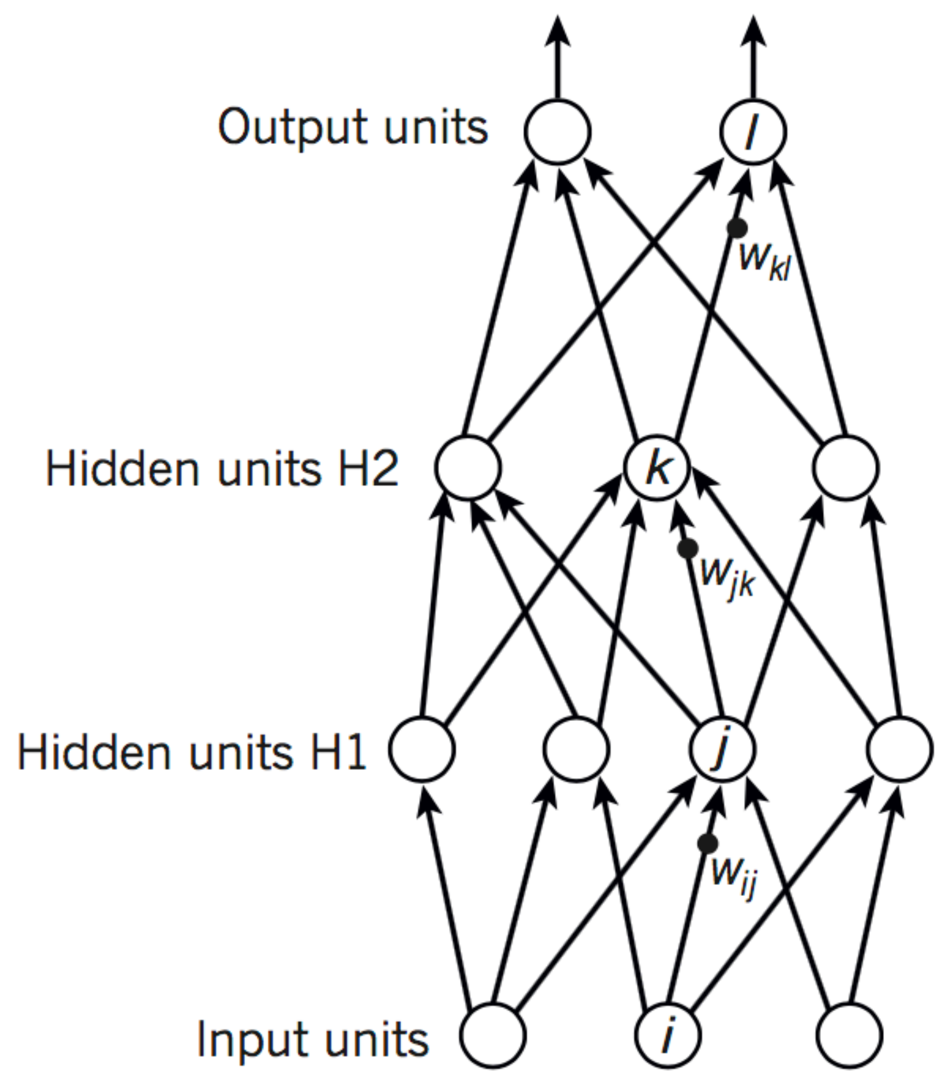
			Hidden layer input		Hidden layer output		
	i_1	i_2	h_1	h_2	h_1	h_2	o
x_1	0	0	0	0	0	1	1
o_1	0	1	1	-1	0	0	0
o_2	1	0	1	-1	0	0	0
x_2	1	1	2	-2	1	0	1

Universal approximation theorem

- Any “reasonable” function can be computed by a multilayer perceptron with a single hidden layer
- But there’s no guarantee that we can **learn** that function from examples
- Multiple hidden layers sometimes help

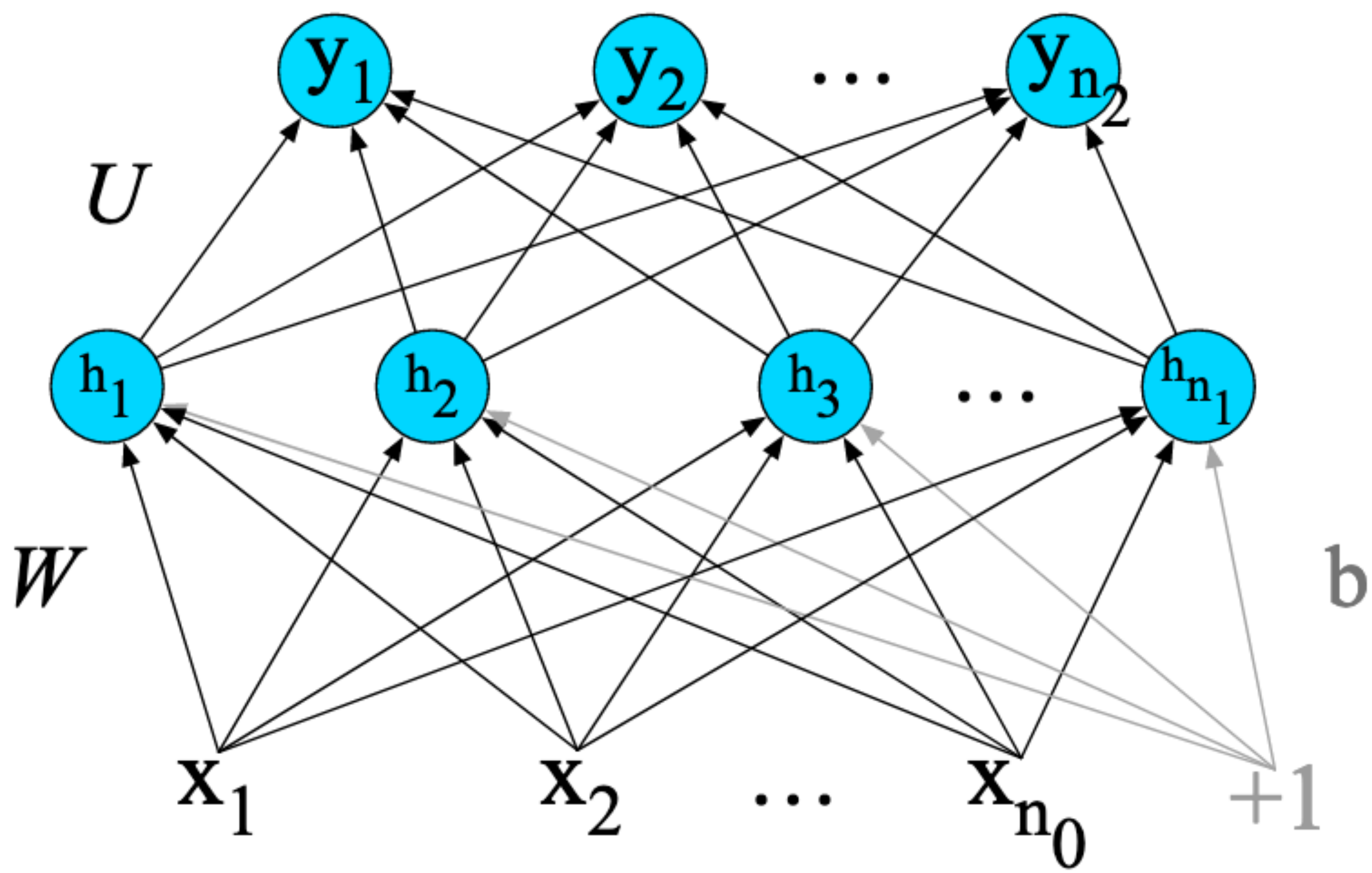
Deep learning

Multiple layers of hidden units:



(Figure from LeCun, Bengio and Hinton, 2015)

Multilayer perceptrons as classifiers

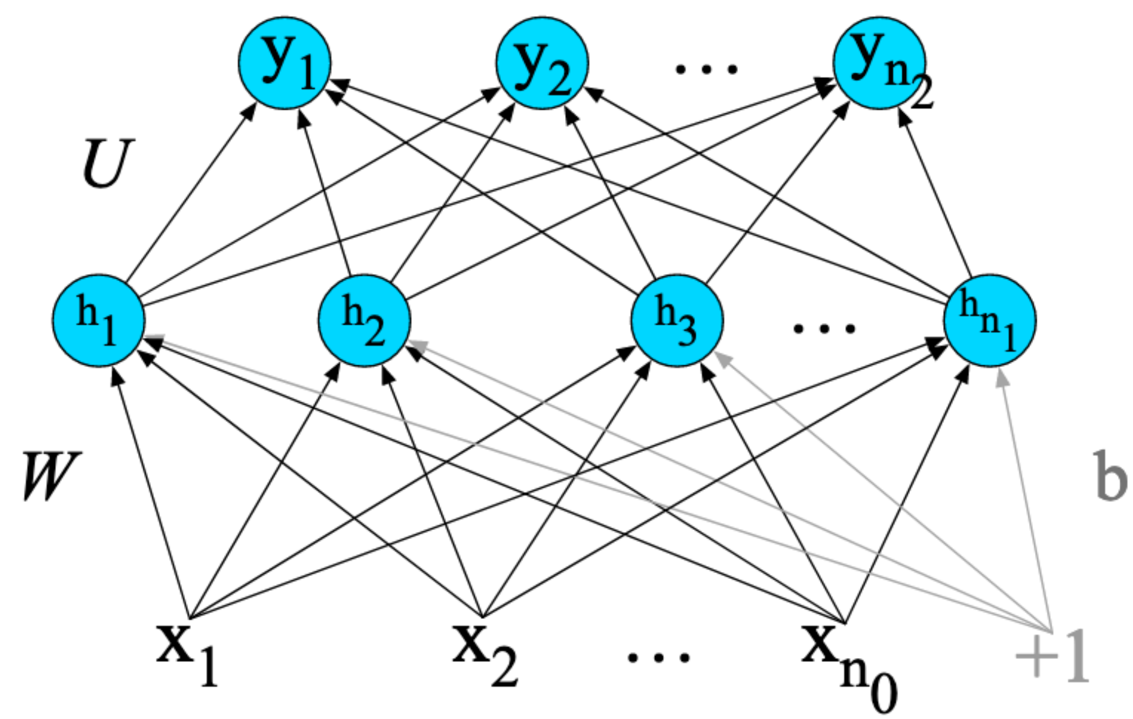


$$\mathbf{y} = \sigma(U\mathbf{h})$$

$$\mathbf{h} = \sigma(W\mathbf{x} + b)$$

(Applying $\boldsymbol{\sigma}$ elementwise)

Multilayer perceptrons as classifiers



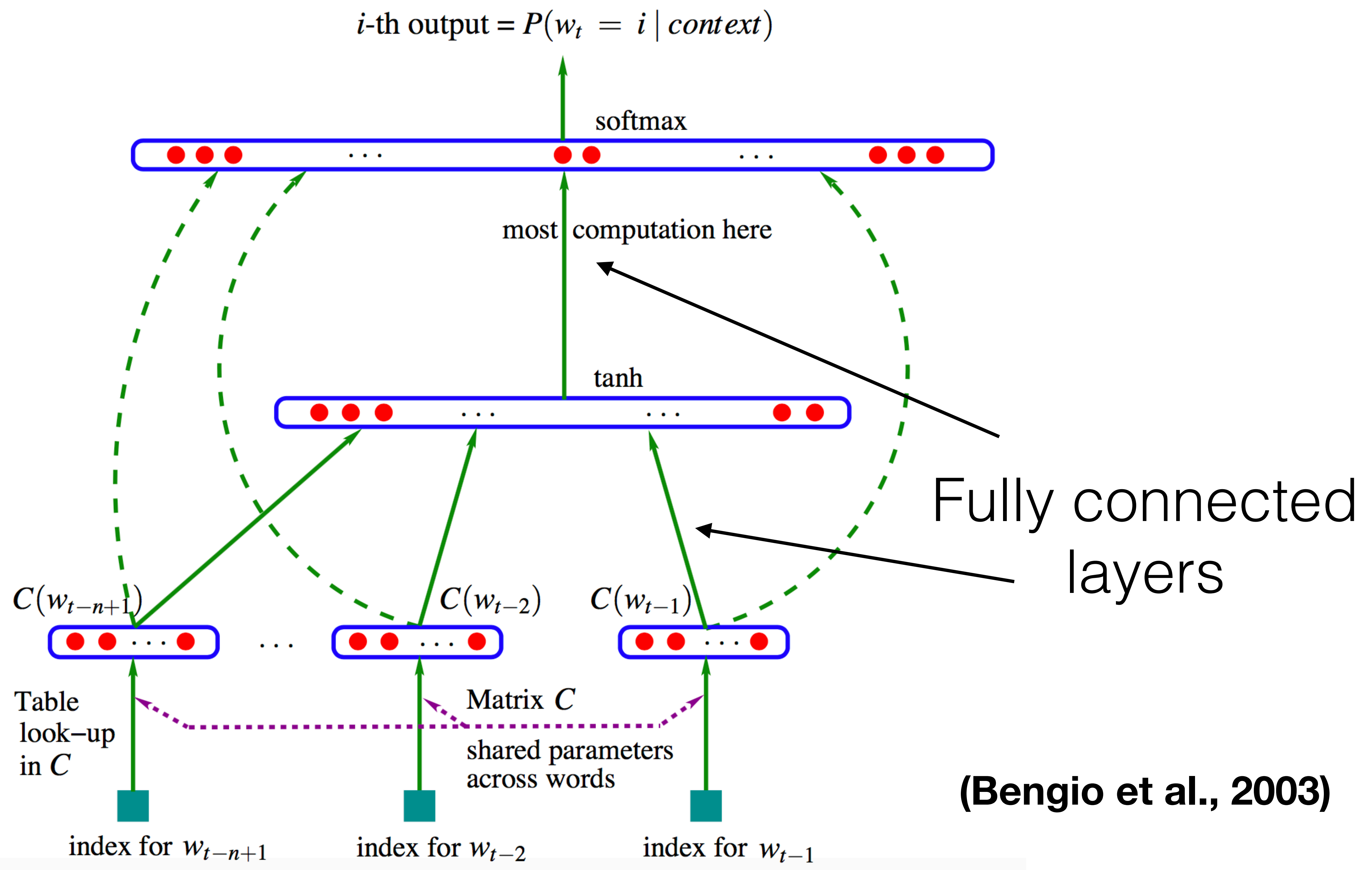
$$\mathbf{z} = \text{softmax}(\mathbf{y})$$

$$\mathbf{y} = \sigma(U\mathbf{h})$$

$$\mathbf{h} = \sigma(W\mathbf{x} + b)$$

$$\text{softmax}(\mathbf{y}) = \left(\frac{e^{y_1}}{\sum_{j=1}^{d} e^{y_j}}, \ldots, \frac{e^{y_d}}{\sum_{j=1}^{d} e^{y_j}} \right)$$

Neural language models



(Bengio et al., 2003)