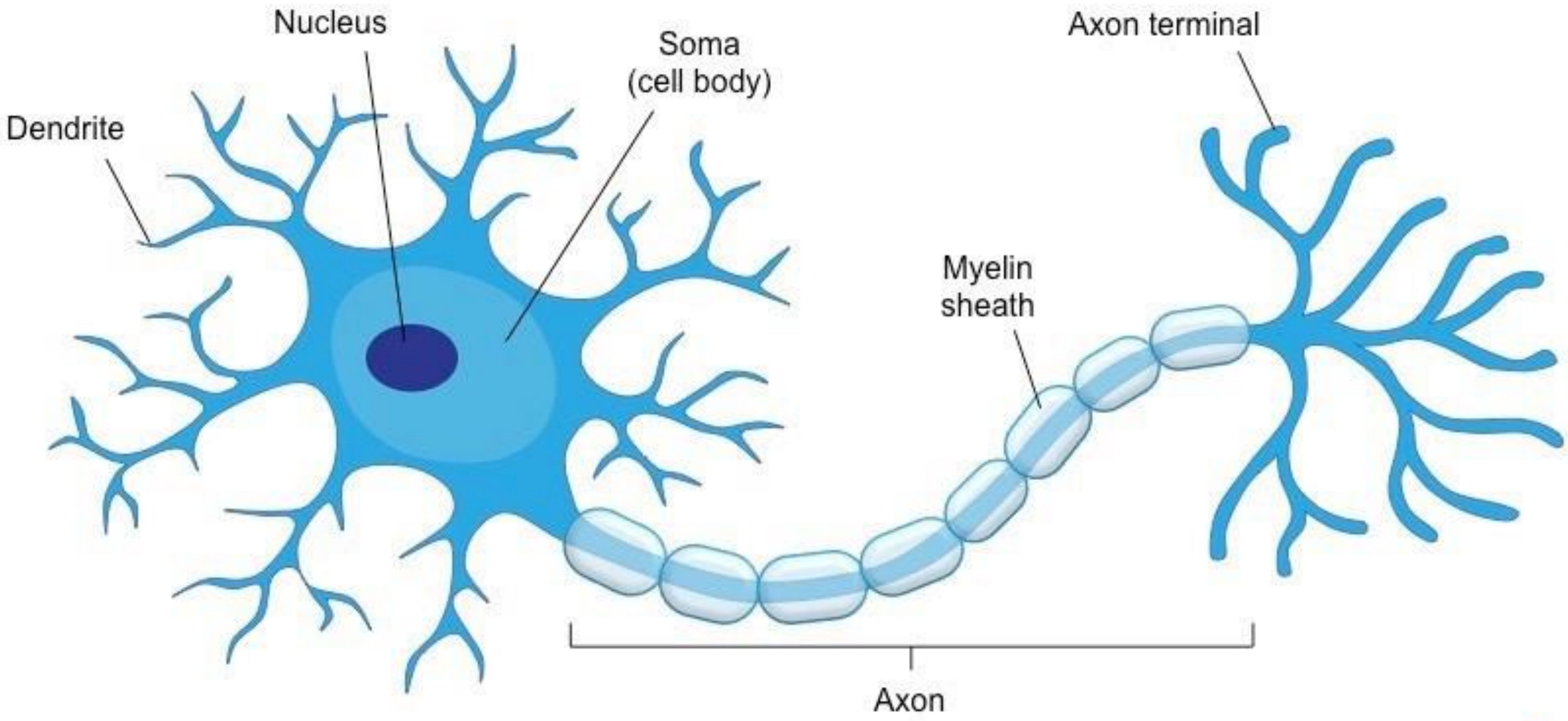
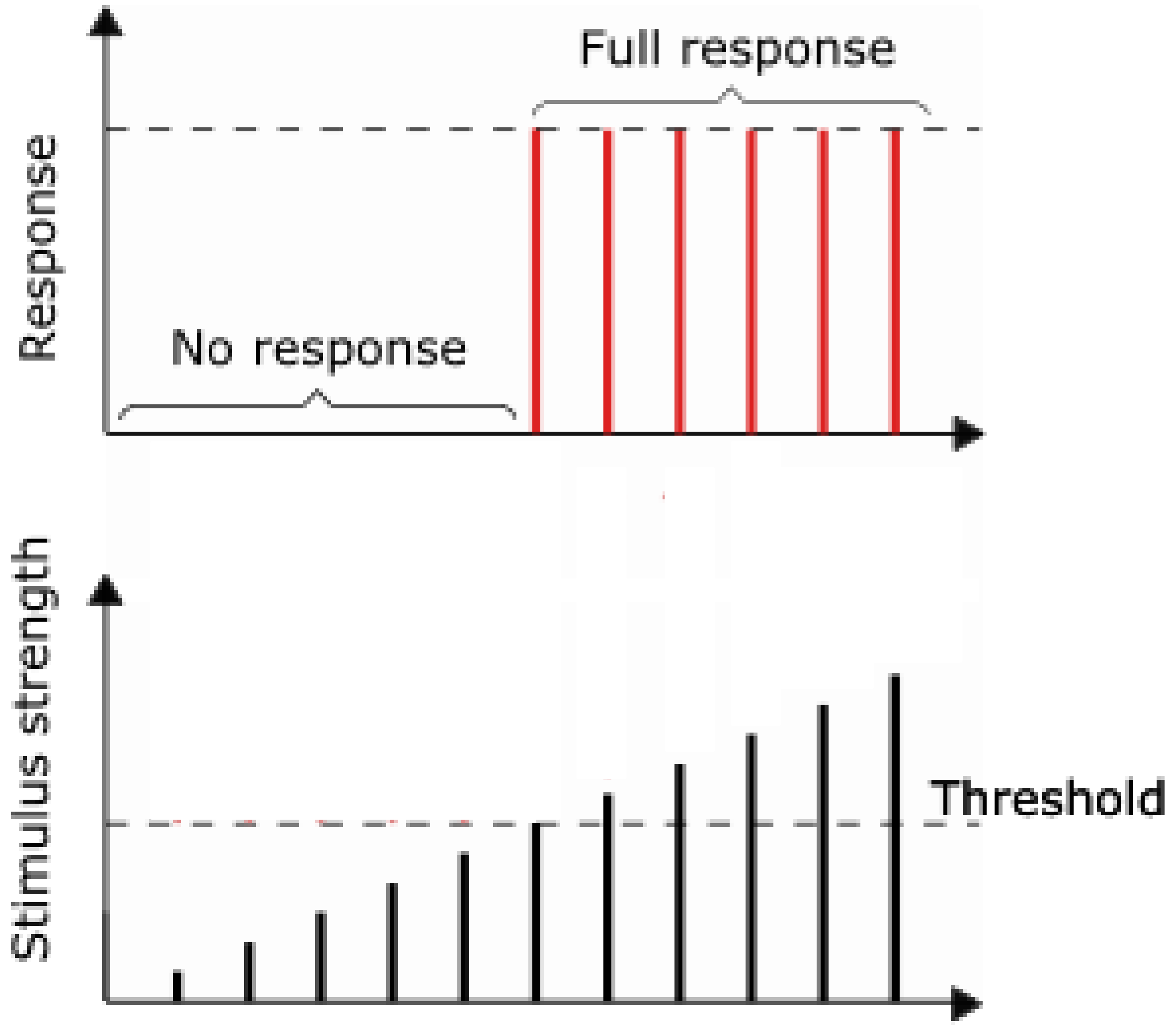
PERCEPTRON

THE DADDY OF NEURAL NETWORKS

FELIPE BUCHBINDER


Nucleus
Soma
(cell body)
Axon terminal
Dendrite
Myelin
sheath
Axon



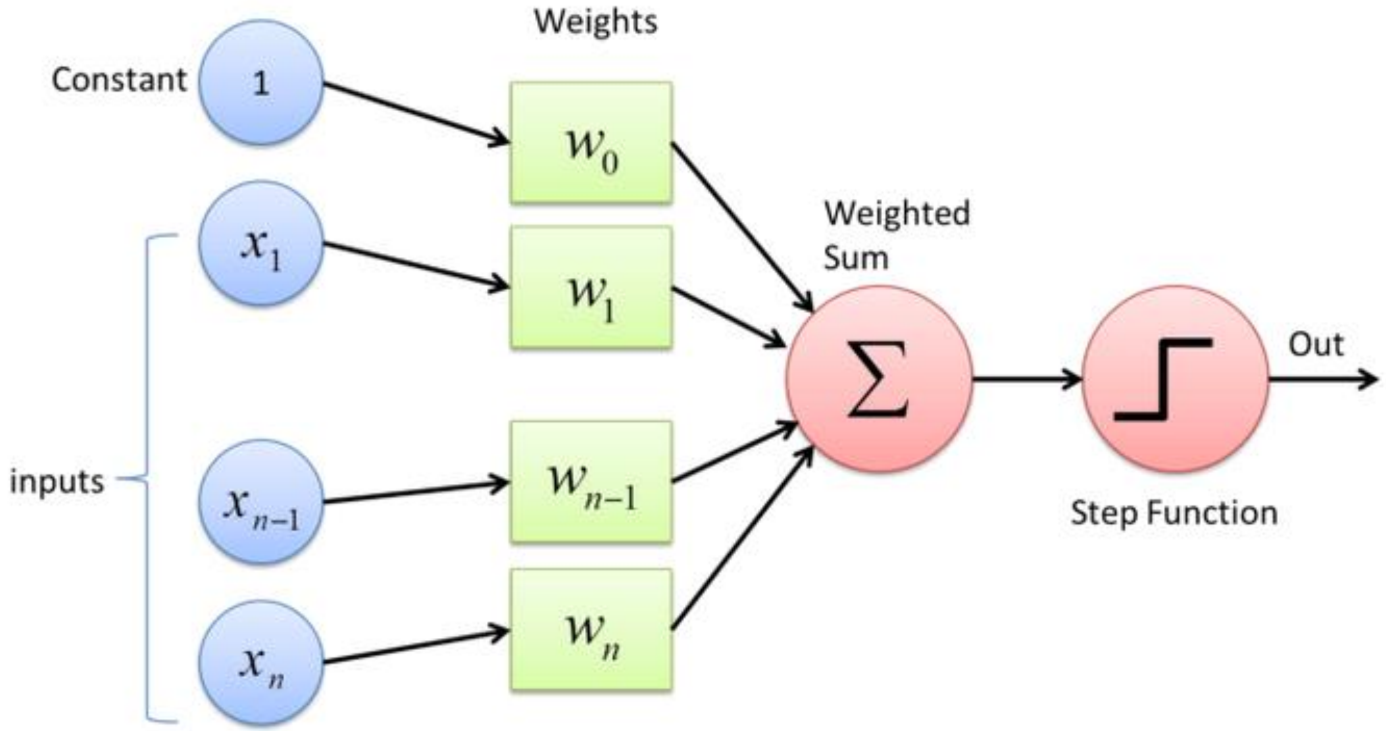
ALL-OR-NOTHING LAW OF NEURONAL ACTIVATION

(WHY IS IT, THEN, THAT WE SOMETIMES FEEL MORE PAIN OR LESS PAIN?)

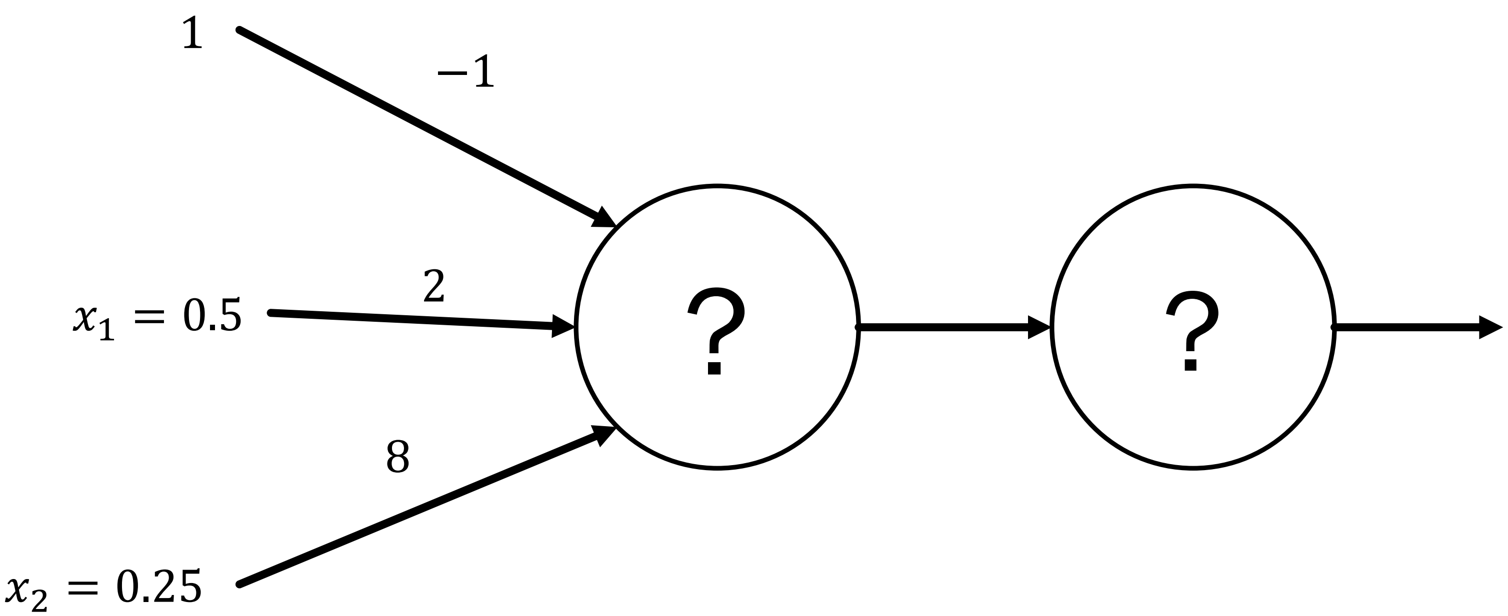
ALL-OR-NOTHING LAW OF NEURONAL ACTIVATION

(WHY IS IT, THEN, THAT WE SOMETIMES FEEL MORE PAIN OR LESS PAIN?)

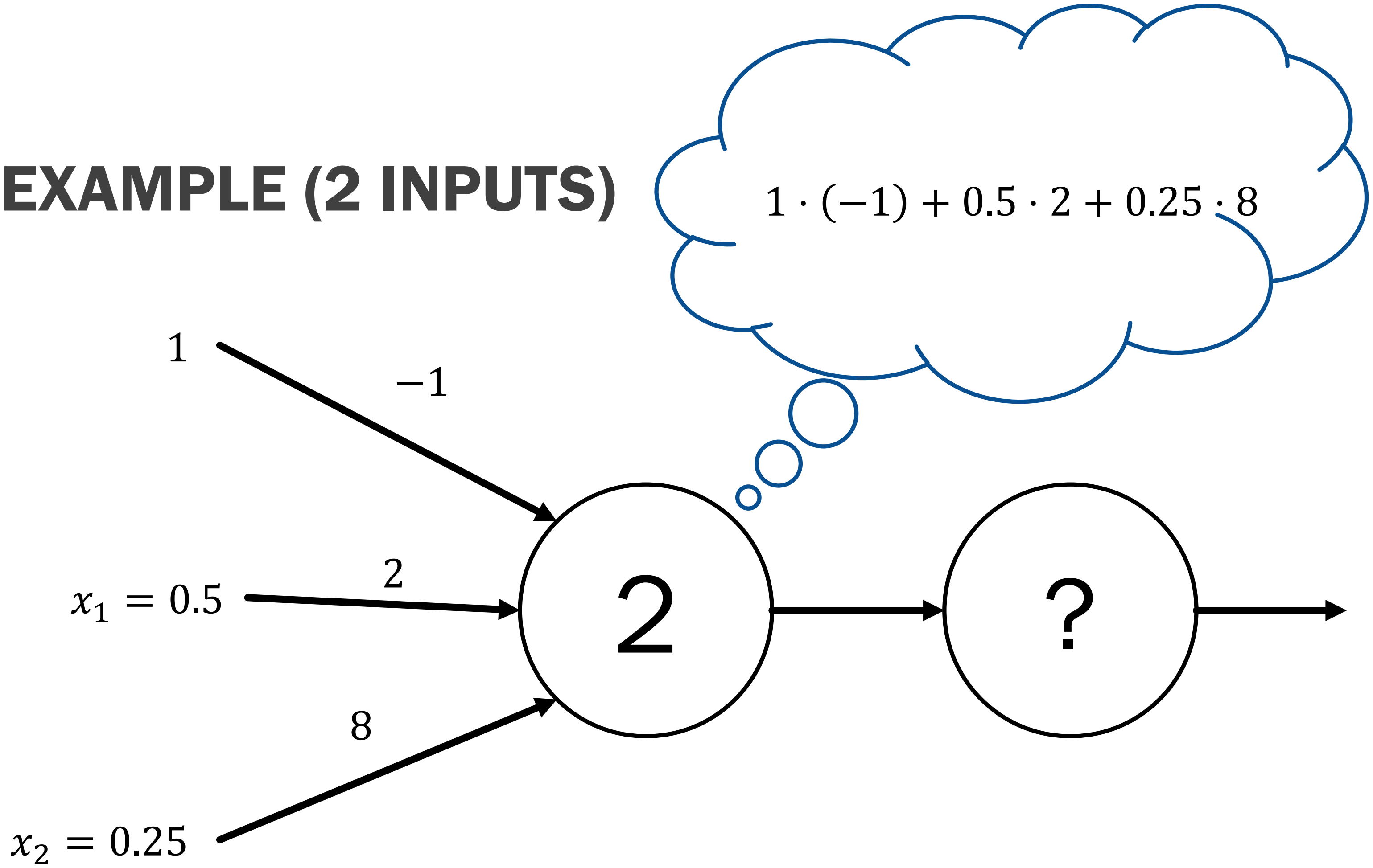
THE PERCEPTRON



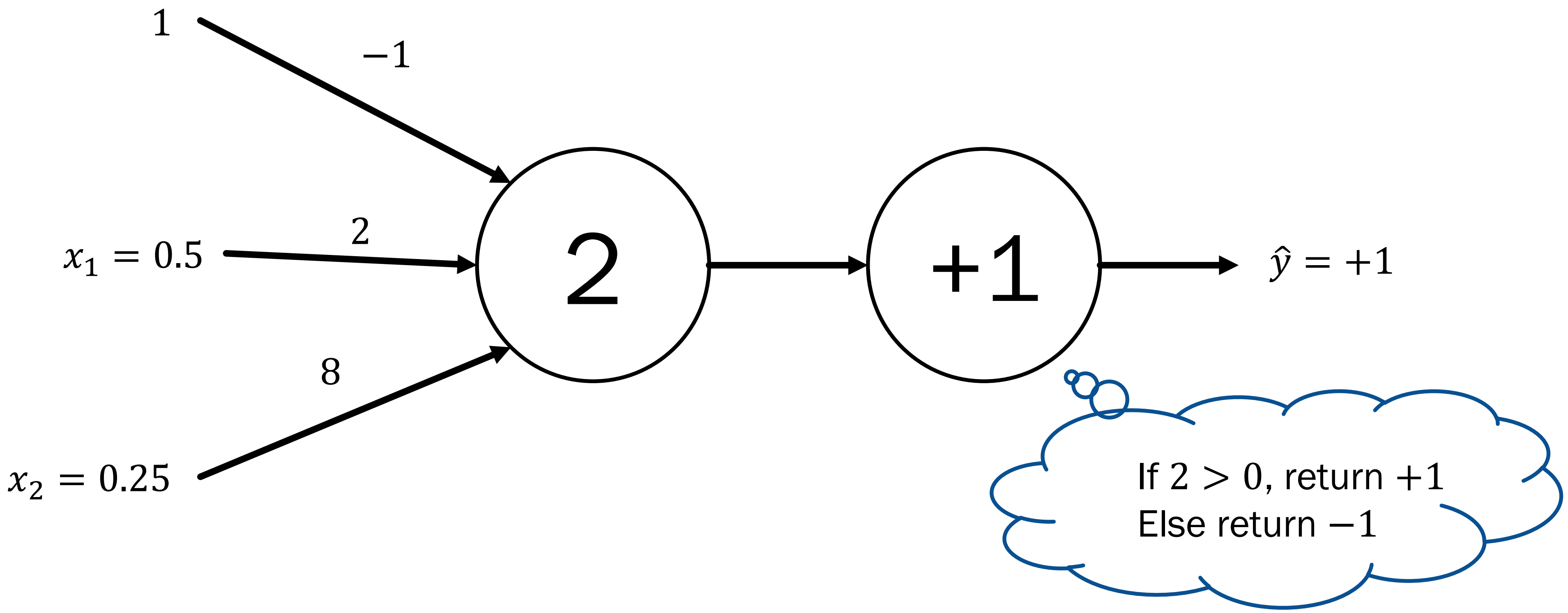
SIMPLE EXAMPLE (2 INPUTS)



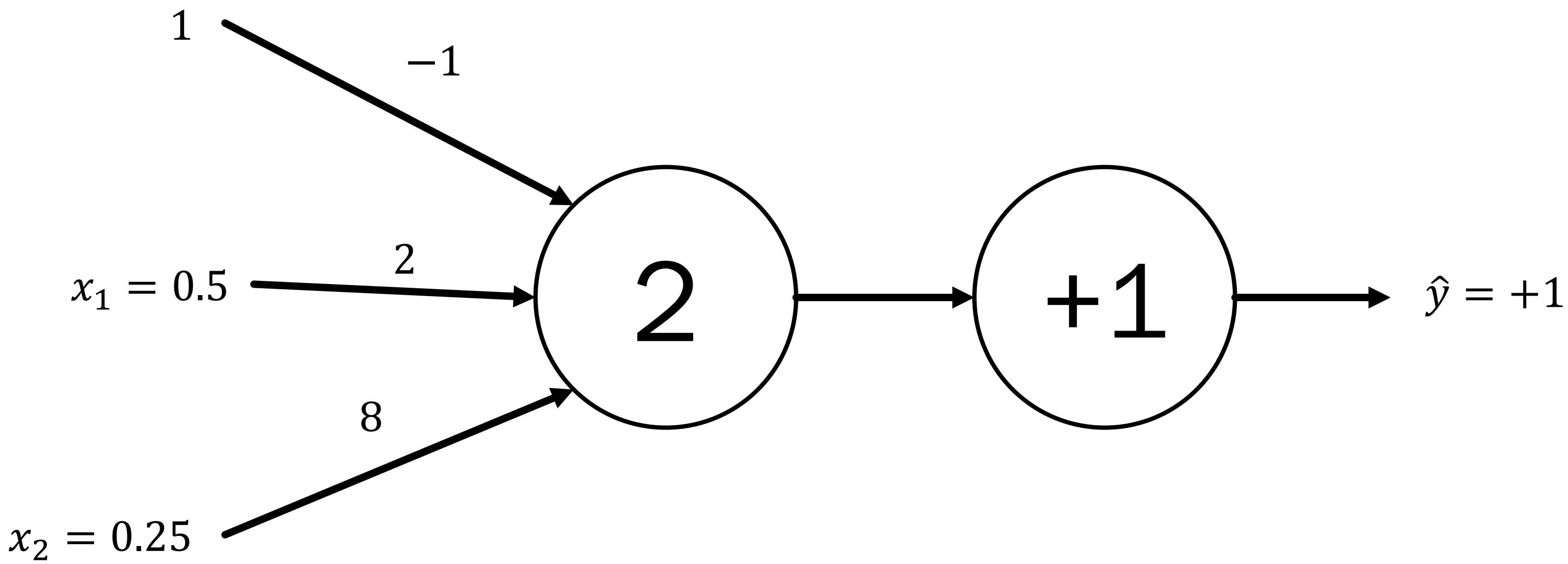
SIMPLE EXAMPLE (2 INPUTS)



SIMPLE EXAMPLE (2 INPUTS)



SIMPLE EXAMPLE (2 INPUTS)

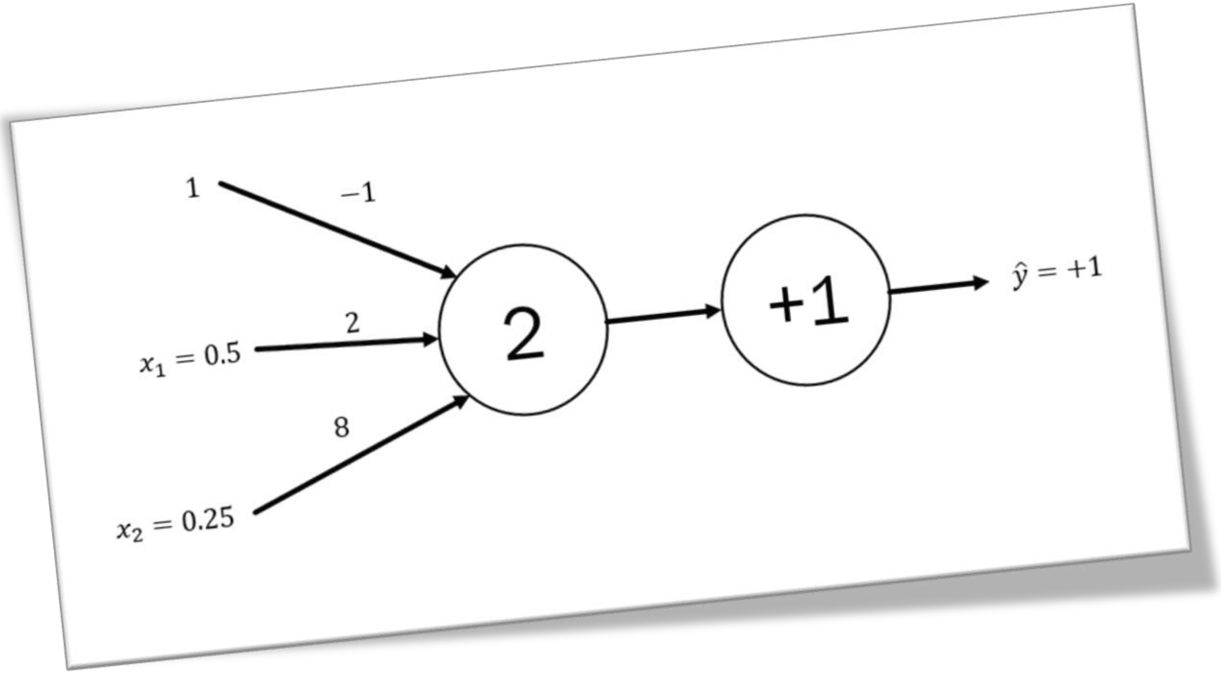


OUR PERCEPTRON'S PREDICTION CAN BE WRITTEN IN A SINGLE LINE

$$\boxed{\hat{y} = \mathrm{sign}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)}$$

OUR PERCEPTRON'S PREDICTION CAN BE WRITTEN IN A SINGLE LINE

$$\boxed{\hat{y} = \text{sign}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)}$$



$$\begin{aligned}\hat{y} &= \text{sign}\big((-1)\cdot 1 + 2\cdot 0.5 + 8\cdot 0.25\big)\\ &= \text{sign}(2)\\ &= +1\end{aligned}$$

PERCEPTRON'S CAN HAVE DIFFERENT ACTIVATION FUNCTIONS

$$\boxed{\hat{y} = \mathrm{sign}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)}$$

Heaviside (step) function

$$\boxed{\hat{y} = \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)}$$

Sigmoid function

$$\boxed{\hat{y} = \tanh\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)}$$

Hyperbolic tangent function

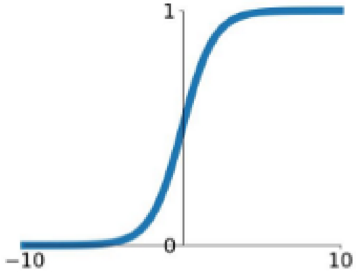
$$\boxed{\hat{y} = \mathrm{ReLU}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)}$$

Rectified Linear Unit

Activation Functions

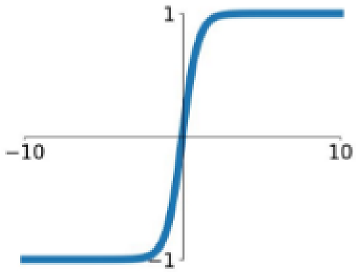
Sigmoid

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



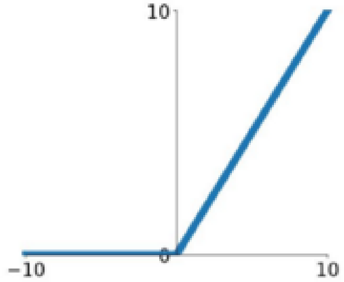
tanh

$\tanh(x)$



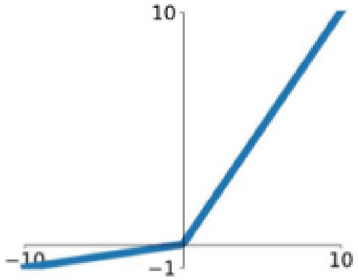
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

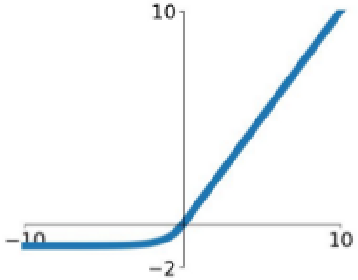


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

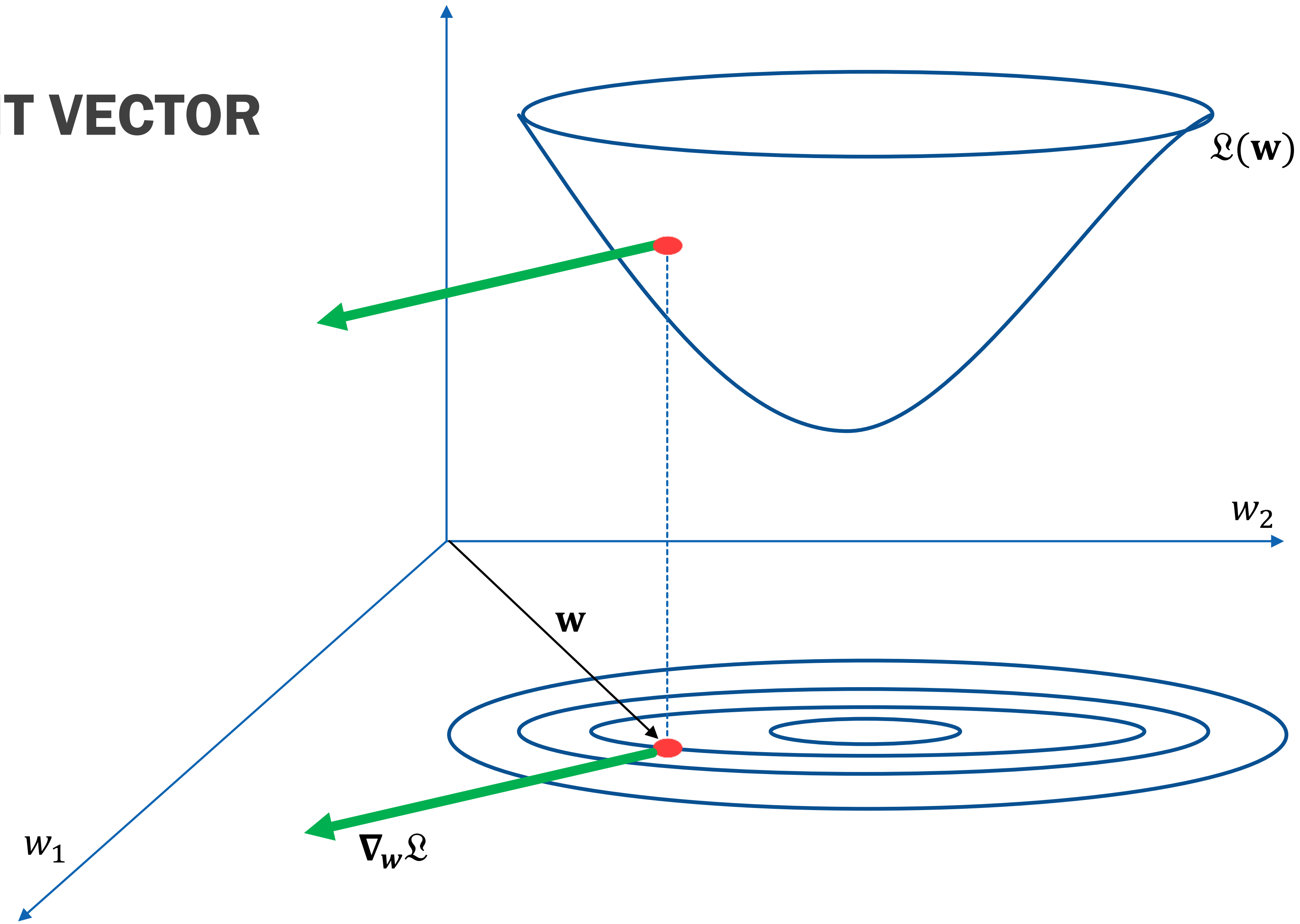
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



HOW CAN THE PERCEPTRON LEARN WHICH WEIGHTS TO USE?

CHOOSE WEIGHTS TO MINIMIZE SOME LOSS FUNCTION

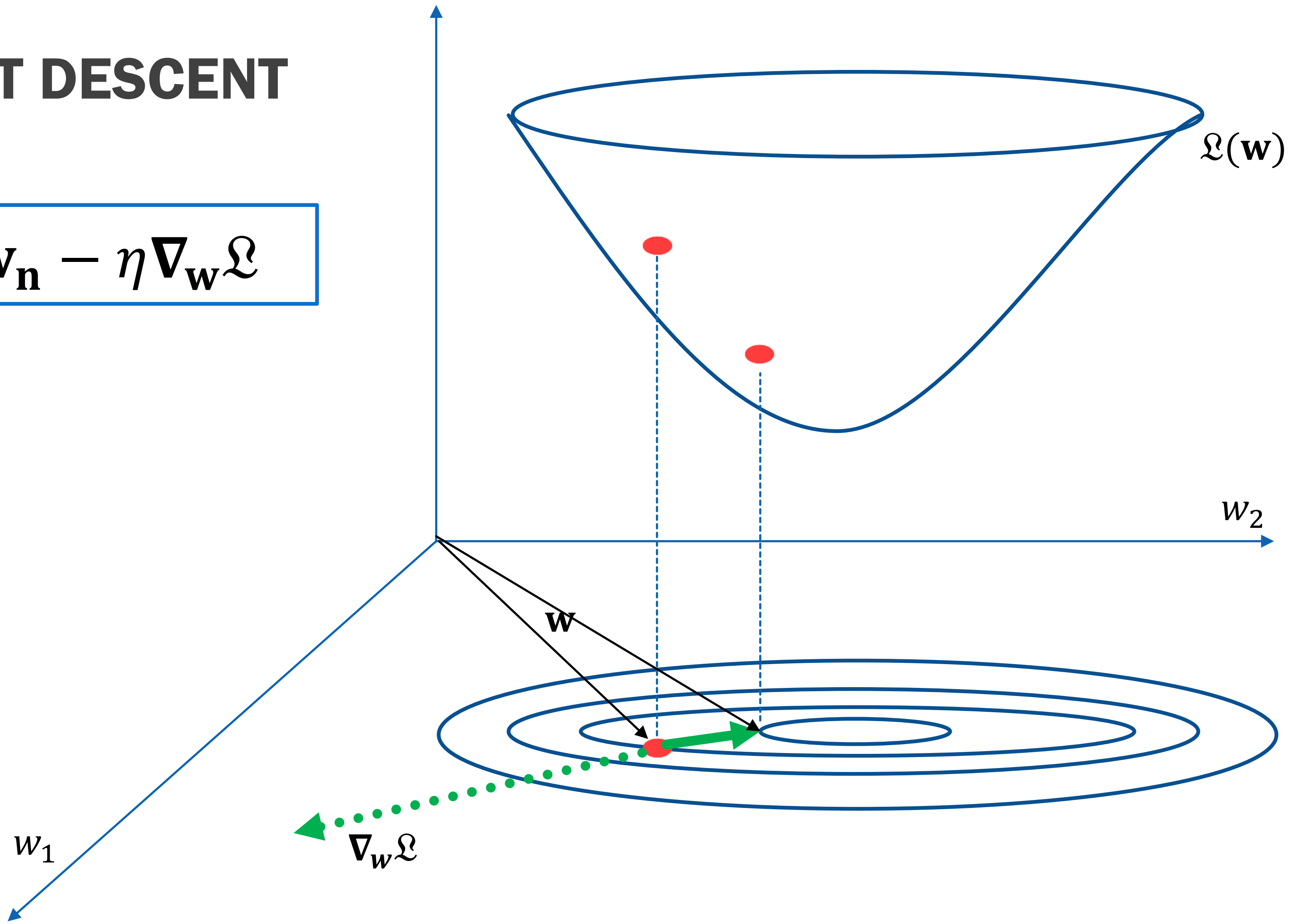
GRADIENT VECTOR



GRADIENT DESCENT

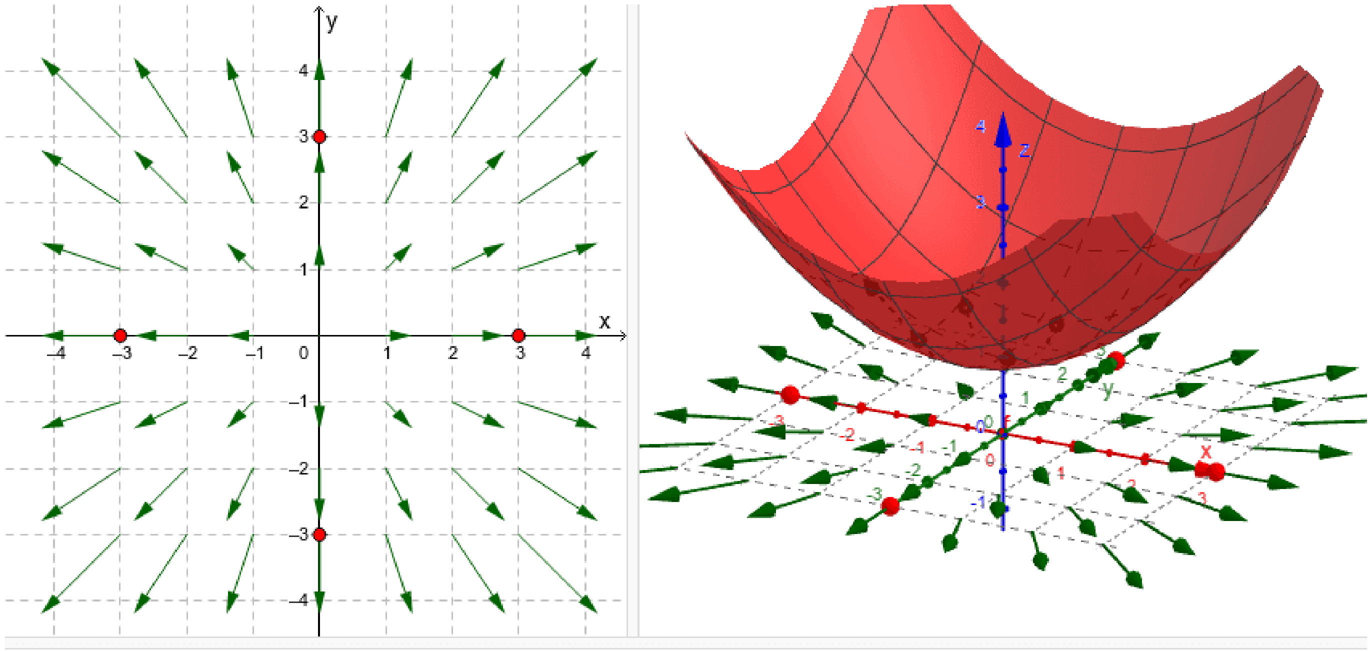
(WITH 1 POINT)

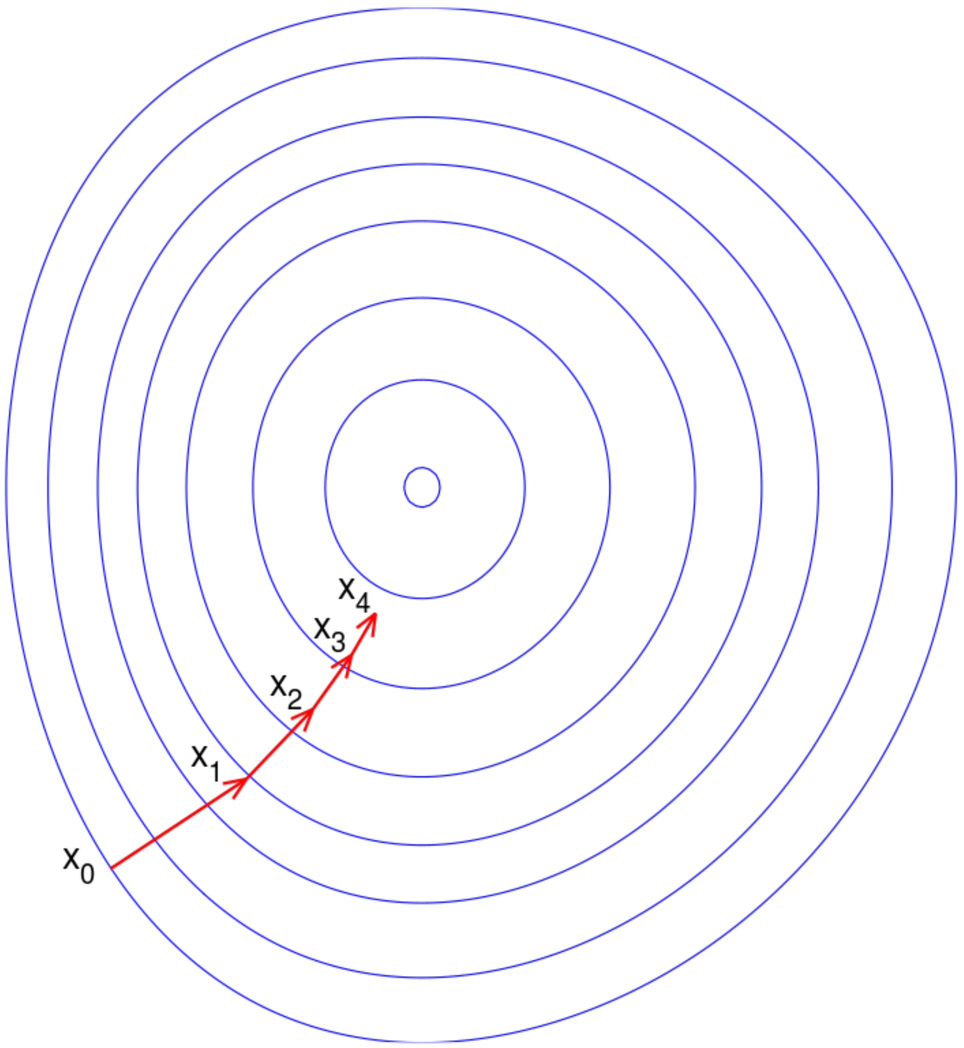
$$\mathbf{w_{n+1}} \leftarrow \mathbf{w_n} - \eta \boldsymbol{\nabla}_{\mathbf{w}} \mathfrak{L}$$



GRADIENT DESCENT

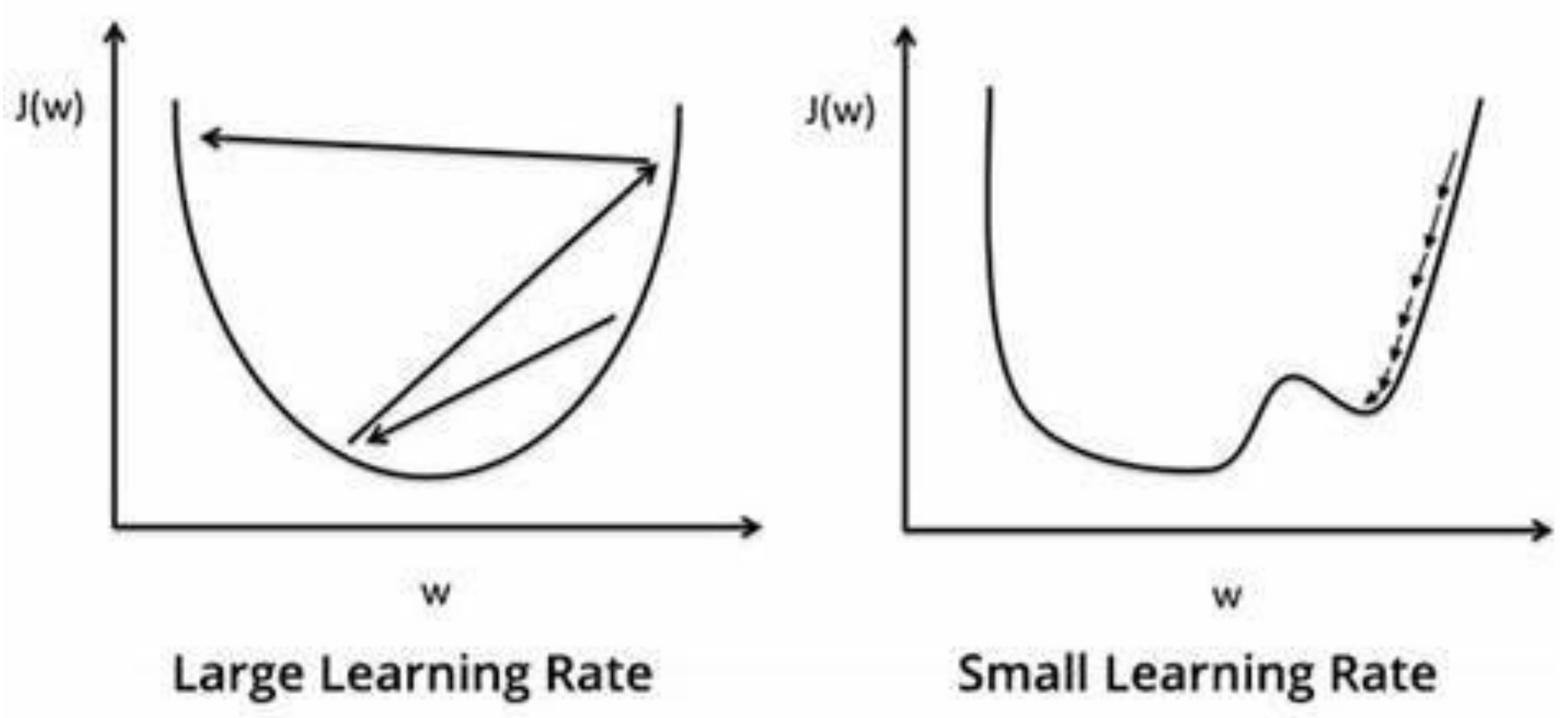
$$\boxed{\mathbf{w_{n+1}} \leftarrow \mathbf{w_n} - \eta \sum_i \nabla_{\mathbf{w}} \mathfrak{L}}$$





x0
x1
x2
x3
x4

EFFECT OF THE LEARNING RATE (η)



EXAMPLE: STEP ACTIVATION FUNCTION WITH HINGE LOSS

$$\mathfrak{L} = \max(0; 1 - y\hat{y})$$
$$\hat{y} = \mathrm{sign}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)$$
$$\mathbf{w_{n+1}} = \mathbf{w_n} - \eta \boldsymbol{\nabla}_{\mathbf{w}} \boldsymbol{\mathfrak{L}}$$

$$\nabla_{\mathbf{w}} \mathfrak{L} = y\mathbf{x}$$

$$\therefore$$

$$\mathbf{w_{n+1}} \leftarrow \mathbf{w_n} - \eta y \mathbf{x}$$

EXAMPLE: LOGISTIC ACTIVATION FUNCTION WITH HINGE LOSS

$$\mathfrak{L} = \max(0; 1 - y\hat{y})$$
$$\hat{y} = \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x})$$
$$\mathbf{w}_{\mathbf{n+1}} = \mathbf{w}_{\mathbf{n}} - \eta \boldsymbol{\nabla}_{\mathbf{w}} \mathfrak{L}$$

$$\nabla_{\mathbf{w}} \mathfrak{L} = y\sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x})[\mathbf{1} - \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x})]\mathbf{x}$$

$\therefore$

$$\mathbf{w}_{\mathbf{n+1}} \leftarrow \mathbf{w}_{\mathbf{n}} - \eta y\sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x})[\mathbf{1} - \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x})]\mathbf{x}$$

EXAMPLE: LOGISTIC ACTIVATION FUNCTION WITH HINGE LOSS

$$\mathfrak{L} = \max(0; 1 - y\hat{y})$$
$$\hat{y} = \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x})$$
$$\mathbf{w_{n+1}} = \mathbf{w_n} - \eta\nabla_{\mathbf{w}}\mathfrak{L}$$

This tells us about the importance of normalizing the inputs. Why?

$$\nabla_{\mathbf{w}}\mathfrak{L} = y\sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x})[\mathbf{1} - \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x})]\mathbf{x}$$

$$\therefore$$

$$\mathbf{w_{n+1}} \leftarrow \mathbf{w_n} - \eta y\sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x})[\mathbf{1} - \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x})]\mathbf{x}$$

EXAMPLE: LOGISTIC ACTIVATION FUNCTION WITH HINGE LOSS

$$\mathfrak{L} = \max(0; 1 - y\hat{y})$$
$$\hat{y} = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$
$$\mathbf{w_{n+1}} = \mathbf{w_n} - \eta\nabla_{\mathbf{w}}\mathfrak{L}$$

This tells us about the importance of normalizing the inputs. Why?

$$\nabla_{\mathbf{w}}\mathfrak{L} = y\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})[\mathbf{1} - \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})]\mathbf{x}$$

$$\therefore$$

$$\mathbf{w_{n+1}} \leftarrow \mathbf{w_n} - \eta y\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})[\mathbf{1} - \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})]\mathbf{x}$$

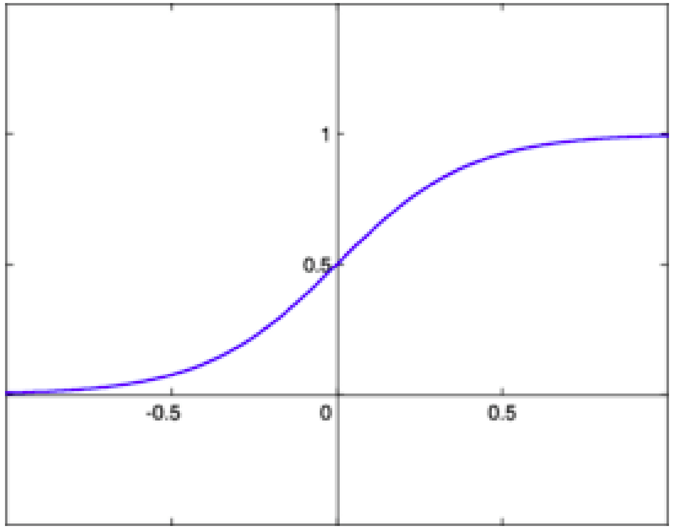
This also offers a nice illustration of the vanishing gradient problem. Why? – and what to do about it?

THE VANISHING GRADIENT PROBLEM

$$\mathbf{w_{n+1}} \leftarrow \mathbf{w_n} - \eta \sum_i \boldsymbol{\nabla}_{\mathbf{w}} \mathfrak{L}$$

When $\nabla_{\mathbf{w}} \mathfrak{L} \rightarrow 0$, learning stops ($\mathbf{w_{n+1}} \approx \mathbf{w_n}$)

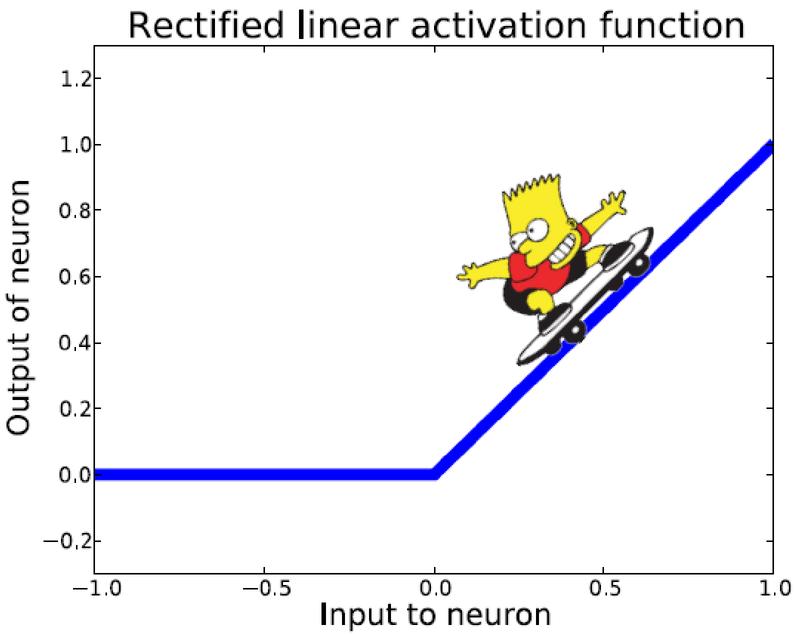
What can we do about it?



SOLUTIONS TO THE VANISHING GRADIENT PROBLEM

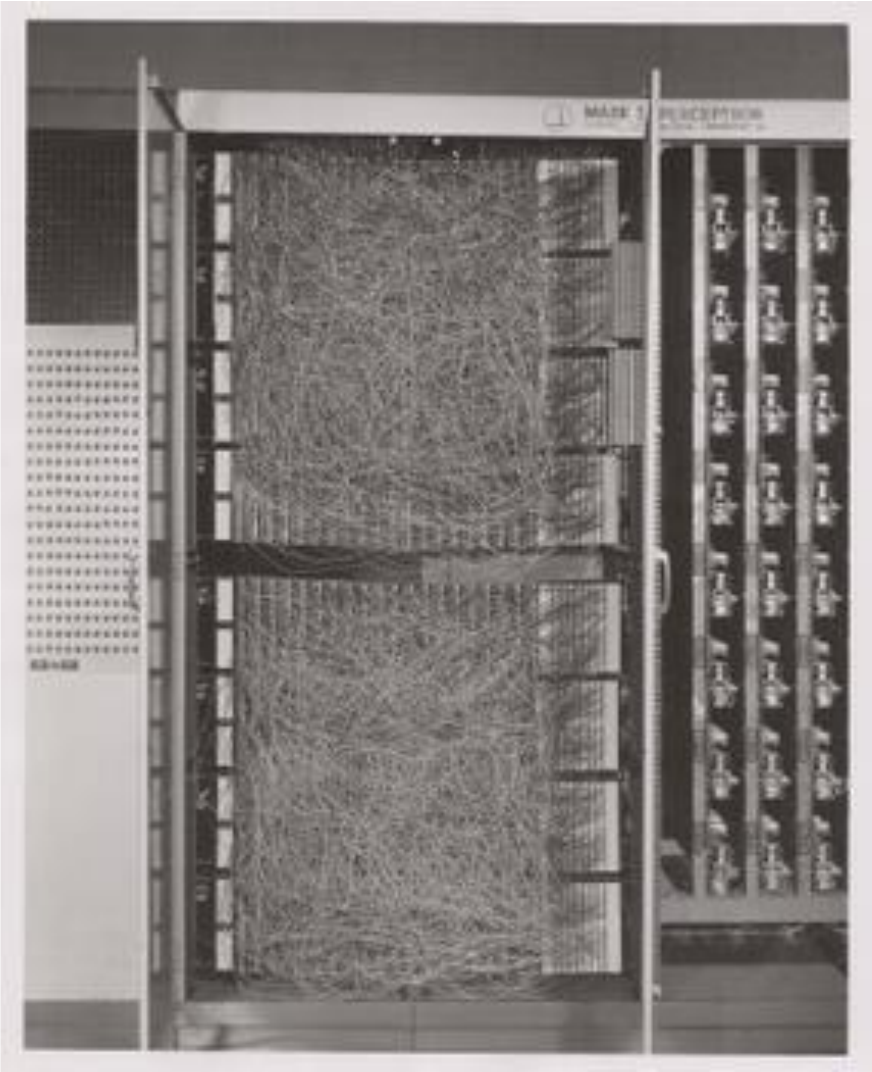
$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Normalization



Changing the activation function

and other more advanced (or pricy) options…



THE FIRST PERCEPTRON (IBM, 1958)

COMING UP NEXT:

NEURAL NETWORKS