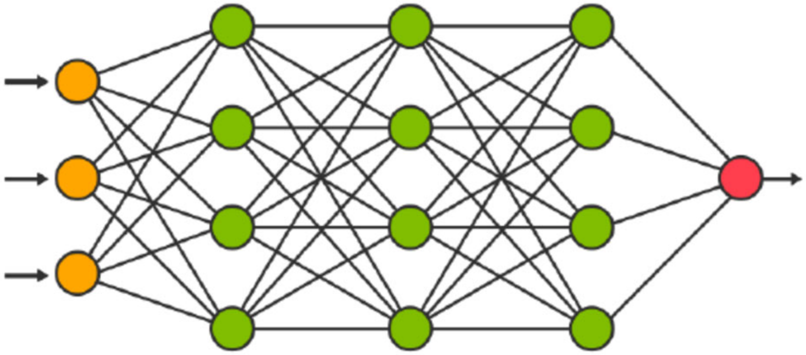
Neural Networks

Fatemeh Mansoori

University of Isfahan

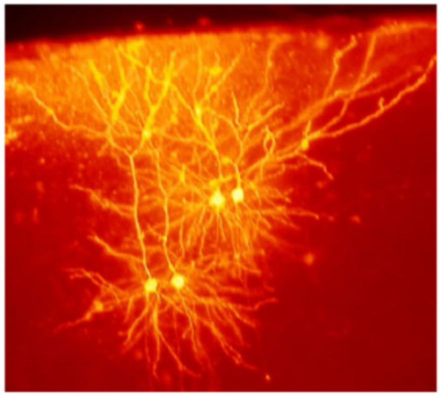
Introduction to Neural Network



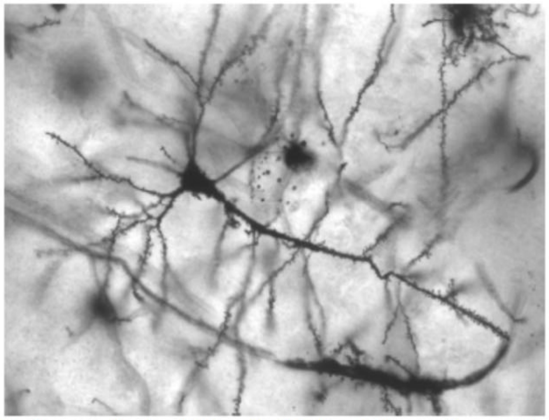
This slides are created based on the slide of the machine learning course at sharif university, Deap learning course Andeow Ng, Deap learning course by Sebastian rashka

Inspired by Biological Brains and Neurons

https://publicdomainpictures.net/en/view-image.php?image=130359&picture=human-brain



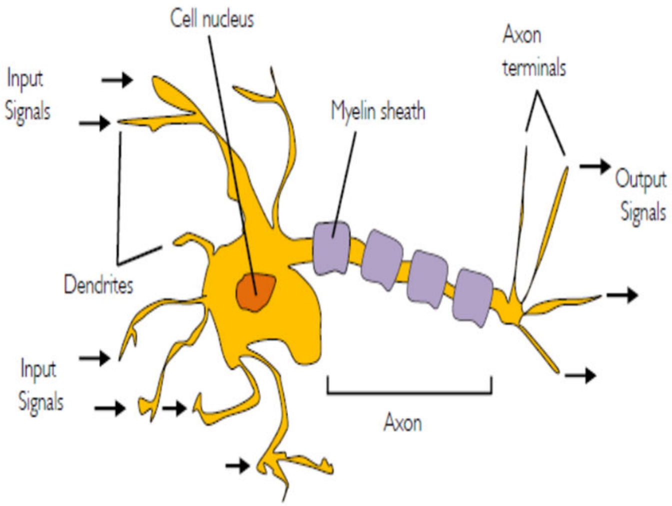
https://commons.wikimedia.org/wiki/Neuron#/media/File:Mouse_cingulate_cortex_neurons.jpg



https://commons.wikimedia.org/wiki/Neuron#/media/File:Pyramidal_hippocampal_neuron_40x.jpg

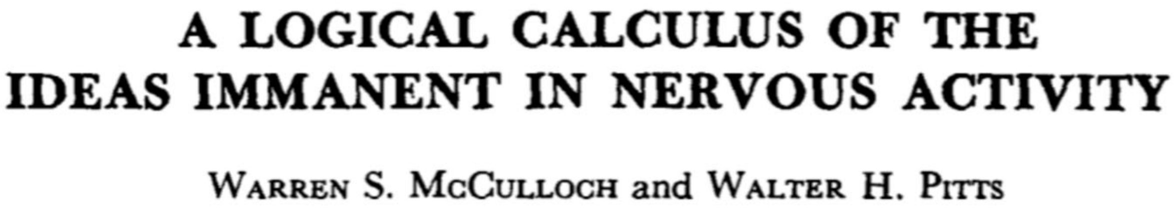
Biological Analogy

- A brain is a set of densely connected neurons
- Depending on the input signals, the neuron performs computations and decides to fire or not.



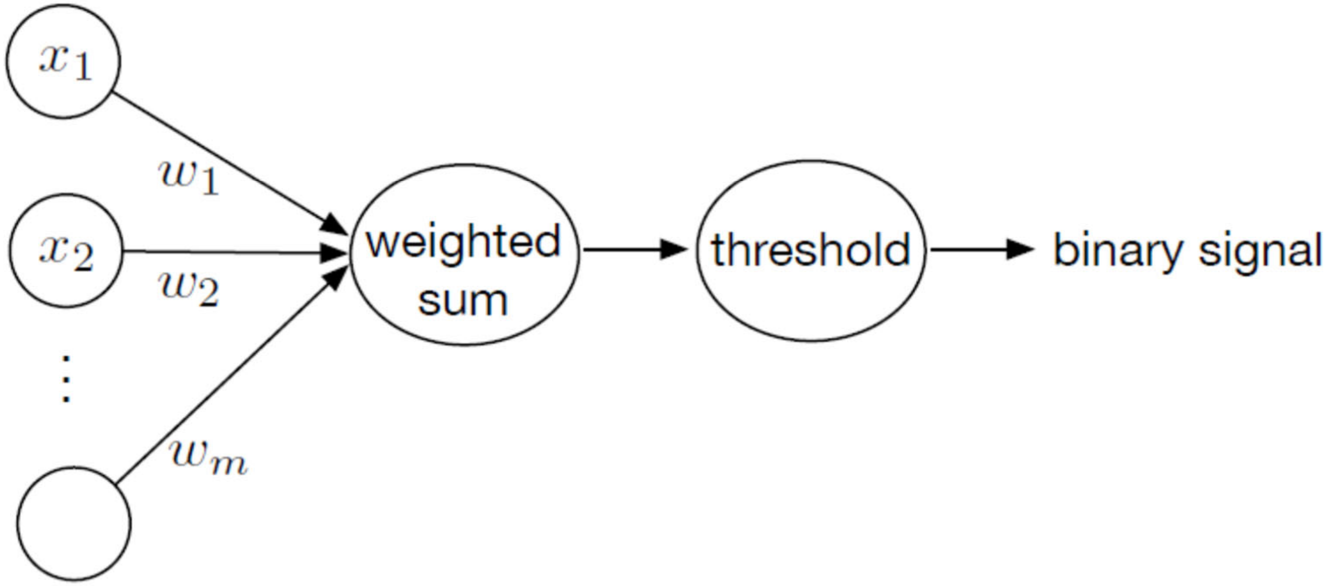
Mathematical formulation of a biological neuron,
could solve AND, OR, NOT problems

McCulloch and Pitts Neuron Model

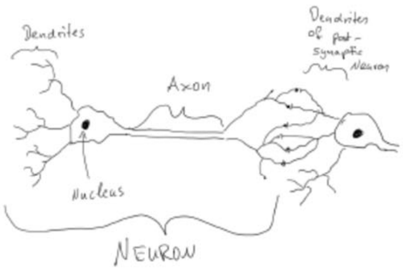


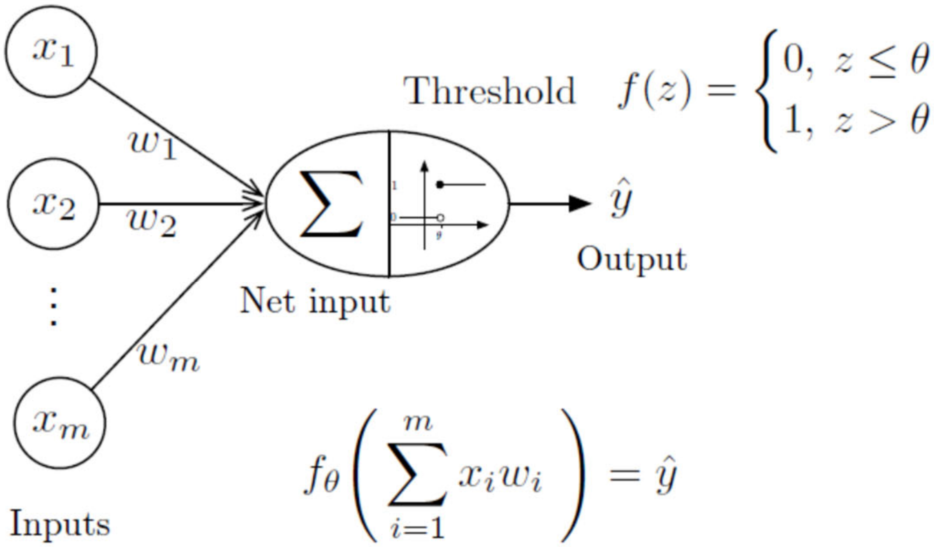
A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. MCCULLOCH and WALTER H. PITTS



A Computational Model of a Biological Neuron





$$f(z) = \begin{cases} 0, & z \leq \theta \\ 1, & z > \theta \end{cases}$$

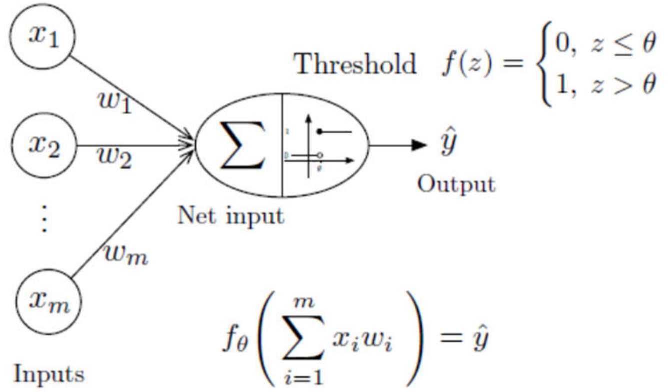
$$f_\theta\left(\sum_{i=1}^{m} x_i w_i\right) = \hat{y}$$

Terminalogy

- Net input = weighted inputs, z
- Activations = activation function(net input); $a = \sigma(z)$
- Label output = threshold(activations of last layer); $\hat{y} = f(a)$

<u>Special cases:</u>

- In perceptron: activation function = threshold function
- In linear regression: activation = net input = output


x_1
x_2
x_m
w_1
w_2
w_m
Inputs
$\sum$
Net input
Threshold $f(z) = \begin{cases} 0, & z \leq \theta \\ 1, & z > \theta \end{cases}$
$\hat{y}$
Output
$f_\theta\left(\sum_{i=1}^{m} x_i w_i\right) = \hat{y}$

Perceptron Output

$$\hat{y} = \begin{cases} 0, & z \leq \theta \\ 1, & z > \theta \end{cases}$$

More convenient to re-arrange:

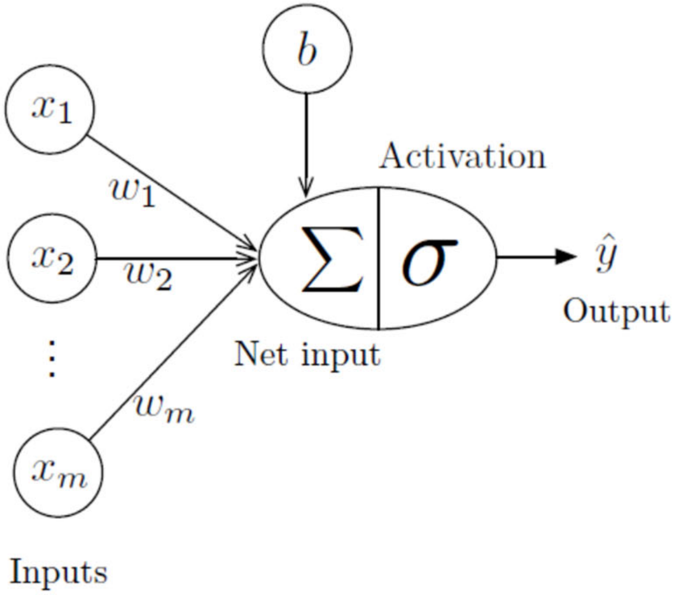
$$\hat{y} = \begin{cases} 0, & z - \theta \leq 0 \\ 1, & z - \theta > 0 \end{cases}$$

negative threshold

$$\boxed{-\theta = \text{"bias"}}$$

General Notation for Single-Layer Neural Networks

- Common notation (in most modern texts): define the bias unit separately
- However, often inconvenient for mathematical notation



"separate" bias unit

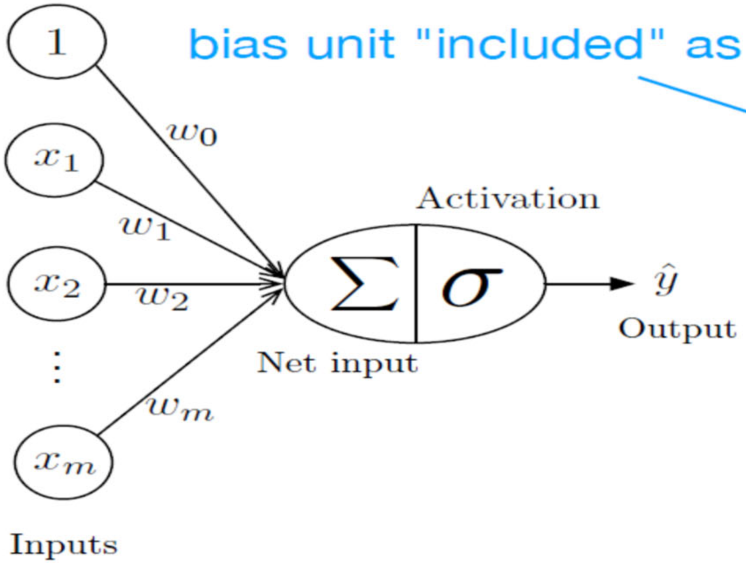
$$\sigma\left(\sum_{i=1}^{m} x_i w_i + b\right) = \sigma\left(\mathbf{x}^T\mathbf{w} + b\right) = \hat{y}$$

$$\sigma(z) = \begin{cases} 0, & z \leq 0 \\ 1, & z > 0 \end{cases}$$

$$b = -\theta$$

General Notation for Single-Layer Neural Networks

- Often more convenient notation: define bias unit as w_0 and prepend a 1 to <u>each</u> input vector as an additional "feature" value
- Modifying input vectors is more inconvenient/inefficient coding-wise, though



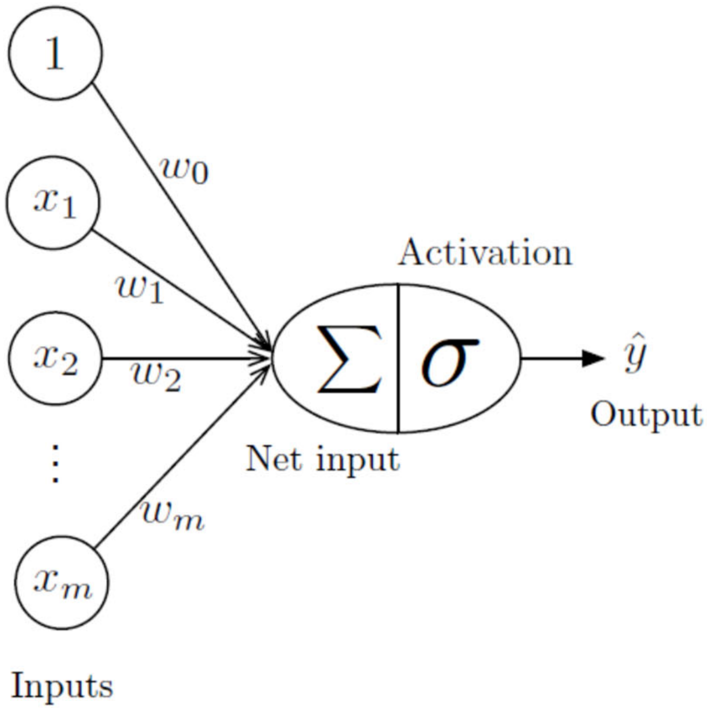
bias unit "included" as w_0

$$\sigma\left(\sum_{i=0}^{m} x_i w_i\right) = \sigma\left(\mathbf{x}^\top \mathbf{w}\right) = \hat{y}$$

$$\sigma(z) = \begin{cases} 0, & z \leq 0 \\ 1, & z > 0 \end{cases}$$

$$w_0 = -\theta$$

General Notation for Single-Layer Neural Networks



Vector dot product

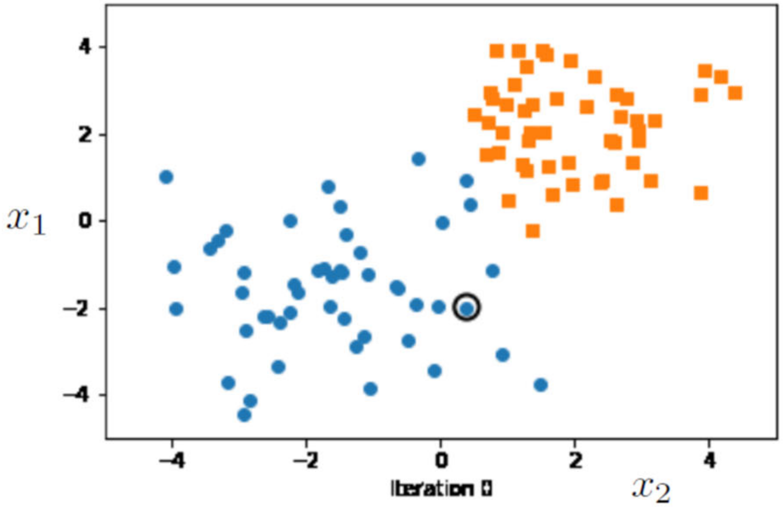
$$\sigma\left(\sum_{i=0}^{m} x_i w_i\right) = \sigma\boxed{(\mathbf{x}^T\mathbf{w})} = \hat{y}$$

$$\sigma(z) = \begin{cases} 0, & z - \theta \leq 0 \\ 1, & z - \theta > 0 \end{cases}$$

$$w_0 = -\theta$$

Perceptron Learning Rule

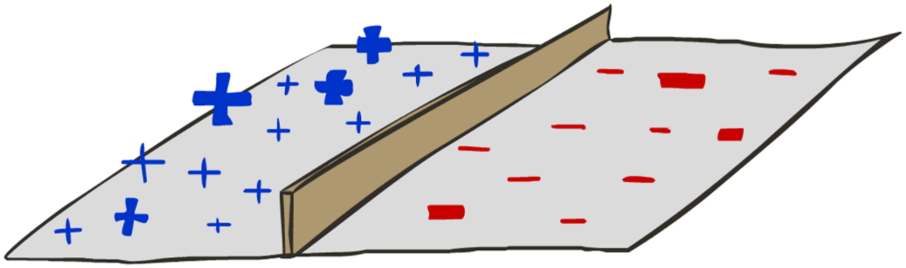
Assume binary classification task, Perceptron finds decision boundary if classes are separable



[animated GIF]

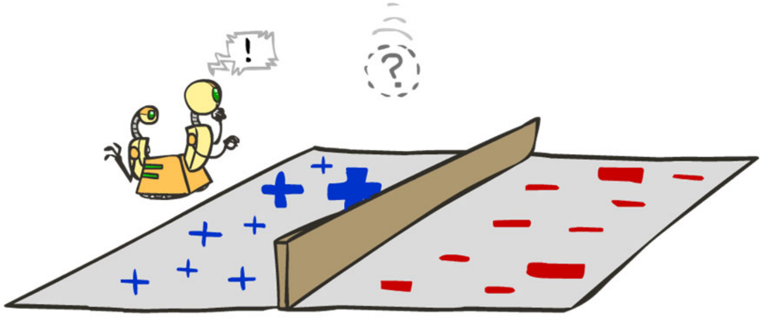
Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1

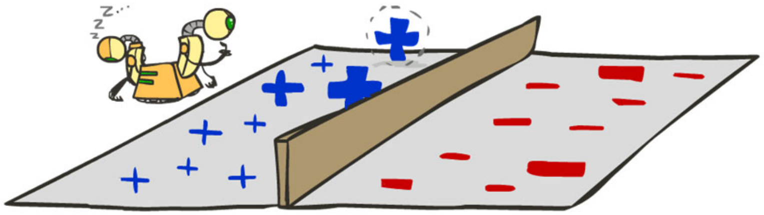


Learning

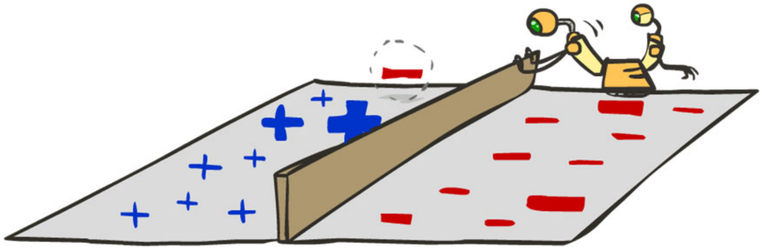
- Start with weights = 0
- For each training instance:
  - Classify with current weights



  - If correct (i.e., y=y*), no change!

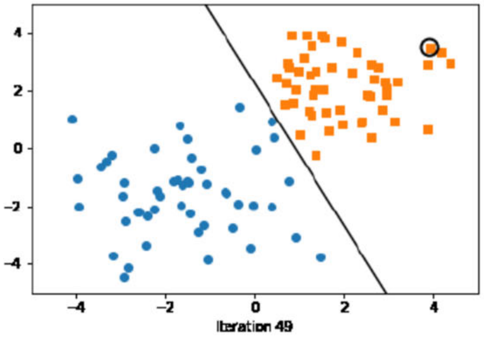


  - If wrong: adjust the weight vector



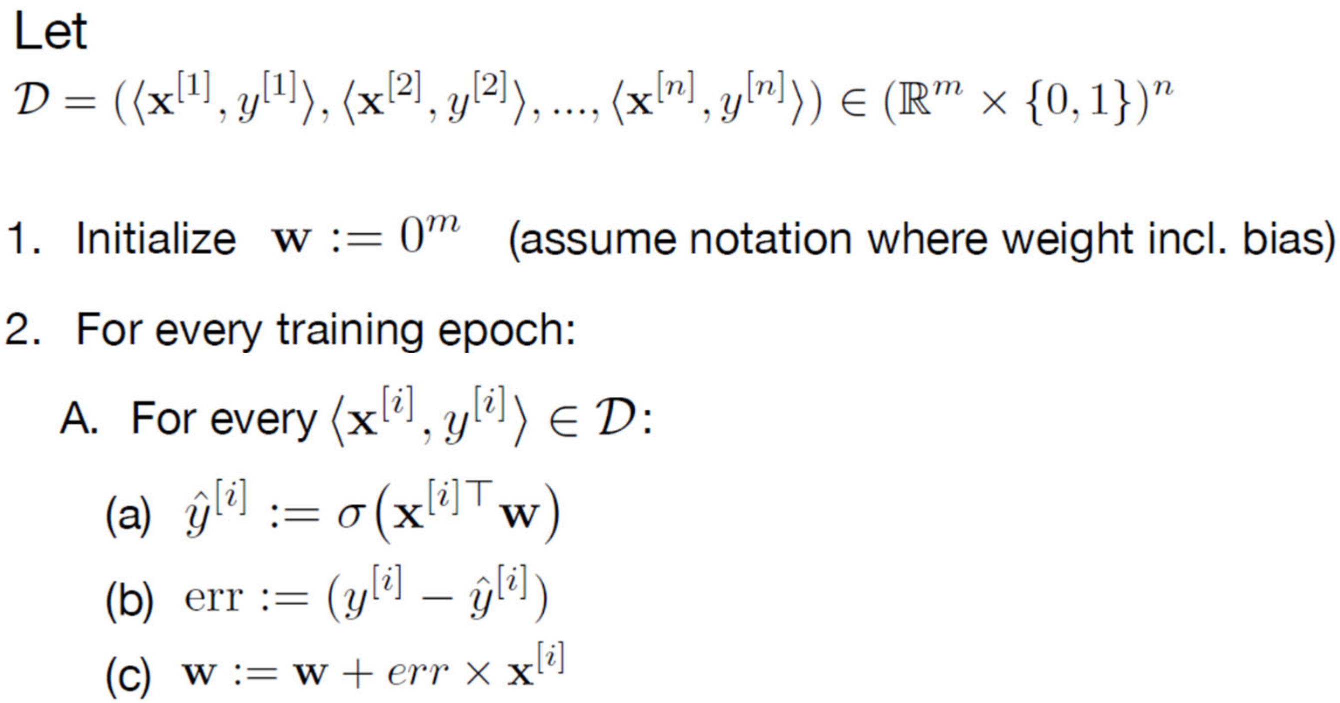
The Perceptron Learning Algorithm

- If correct: Do nothing if the prediction if output is equal to the target
- If incorrect, scenario **a)**:
  If output is 0 and target is 1, add input vector to weight vector
- If incorrect, scenario **b)**:
  If output is 1 and target is 0, subtract input vector from weight vector



Guaranteed to converge if a solution exists
(more about that later...)

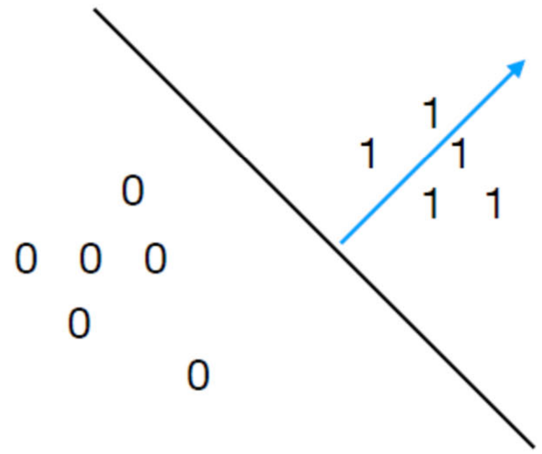
The Perceptron Learning Algorithm



Let
$\mathcal{D} = (\langle \mathbf{x}^{[1]}, y^{[1]} \rangle, \langle \mathbf{x}^{[2]}, y^{[2]} \rangle, ..., \langle \mathbf{x}^{[n]}, y^{[n]} \rangle) \in (\mathbb{R}^m \times \{0,1\})^n$

```
1. Initialize  $\mathbf{w} := 0^m$   (assume notation where weight incl. bias)
2. For every training epoch:
   A. For every $\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathcal{D}$:
      (a) $\hat{y}^{[i]} := \sigma\left(\mathbf{x}^{[i]\top}\mathbf{w}\right)$
      (b) $\mathrm{err} := (y^{[i]} - \hat{y}^{[i]})$
      (c) $\mathbf{w} := \mathbf{w} + err \times \mathbf{x}^{[i]}$
```

Geometric Intuition

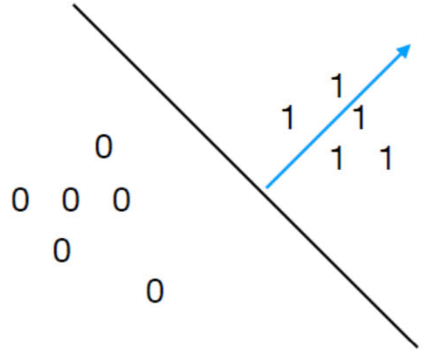


Decision boundary

Weight vector is perpendicular to the boundary. Why?

Geometric Intuation

Decision boundary



Weight vector is perpendicular to the boundary. Why?

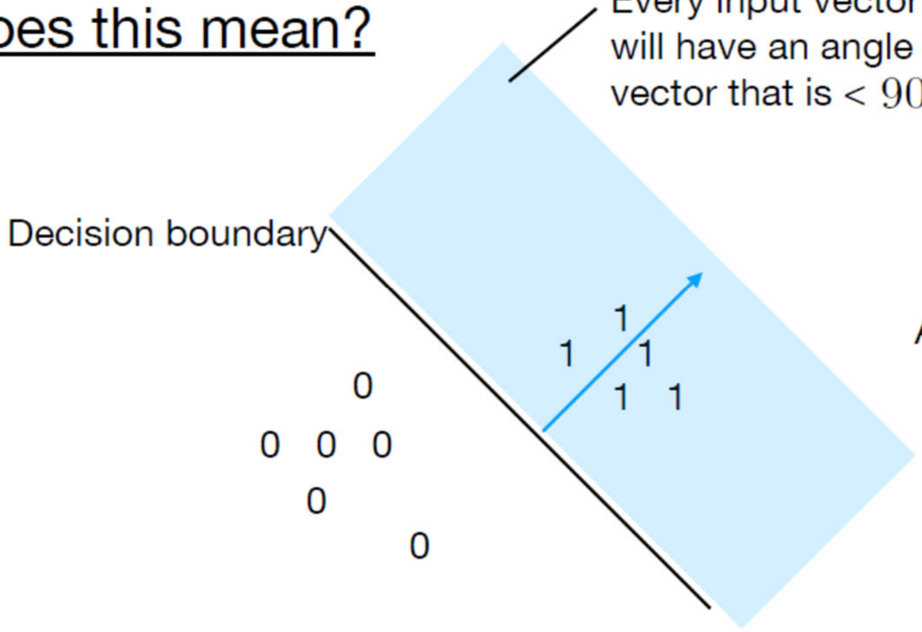
Remember,

$$\hat{y} = \begin{cases} 0, & \mathbf{w}^T\mathbf{x} \leq 0 \\ 1, & \mathbf{w}^T\mathbf{x} > 0 \end{cases}$$

$$\mathbf{w}^T\mathbf{x} = ||\mathbf{w}|| \cdot ||\mathbf{x}|| \cdot \underbrace{\cos(\theta)}$$

So this needs to be 0 at the boundary, and it is zero at 90°

What else does this mean?

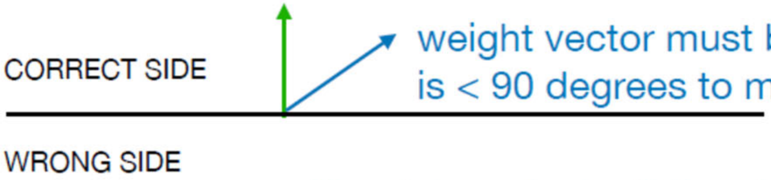


Every input vector on this side will have an angle with the weight vector that is $< 90^{\circ}$

Decision boundary

Assume origin (0, 0)
and no bias

input vector for an example with label 1

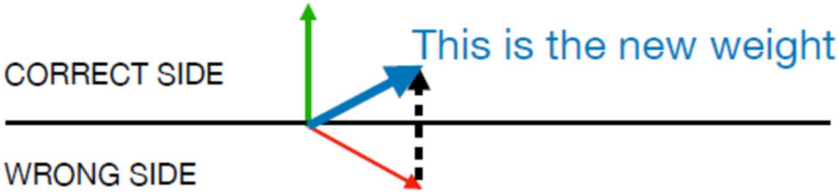


weight vector must be somewhere such that the angle is < 90 degrees to make a correct prediction

The dot product will then be positive, i.e., > 0, since

$$\mathbf{w}^T\mathbf{x} = ||\mathbf{w}|| \cdot ||\mathbf{x}|| \cdot \cos(\theta)$$

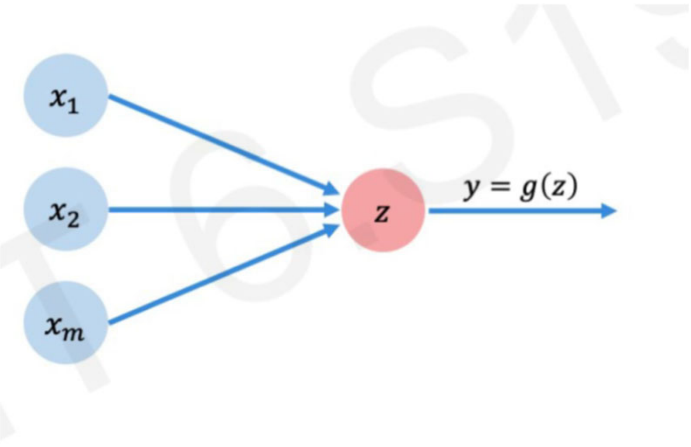
input vector for an example with label 1



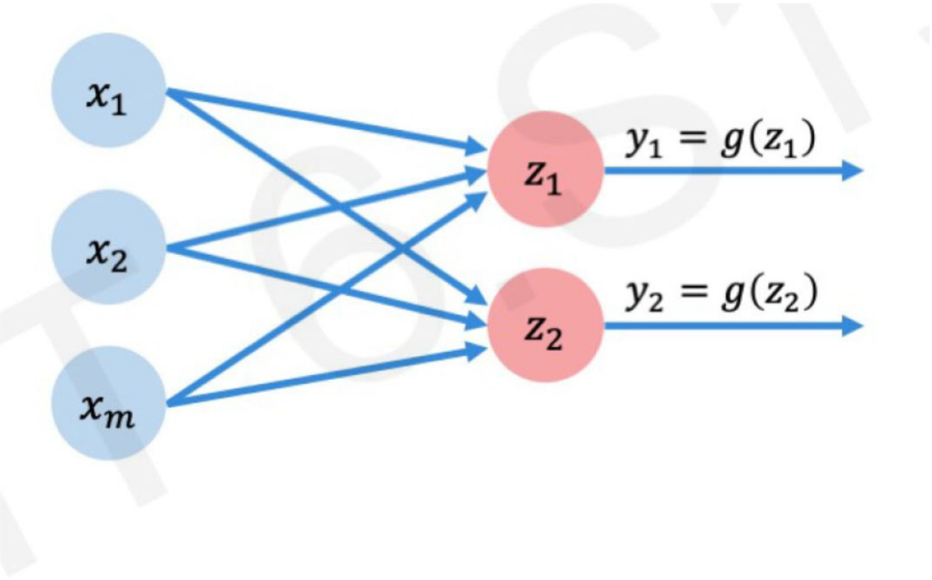
This is the new weight vector

For this weight vector, we make a wrong prediction; hence, we update

Multi Output Perceptron

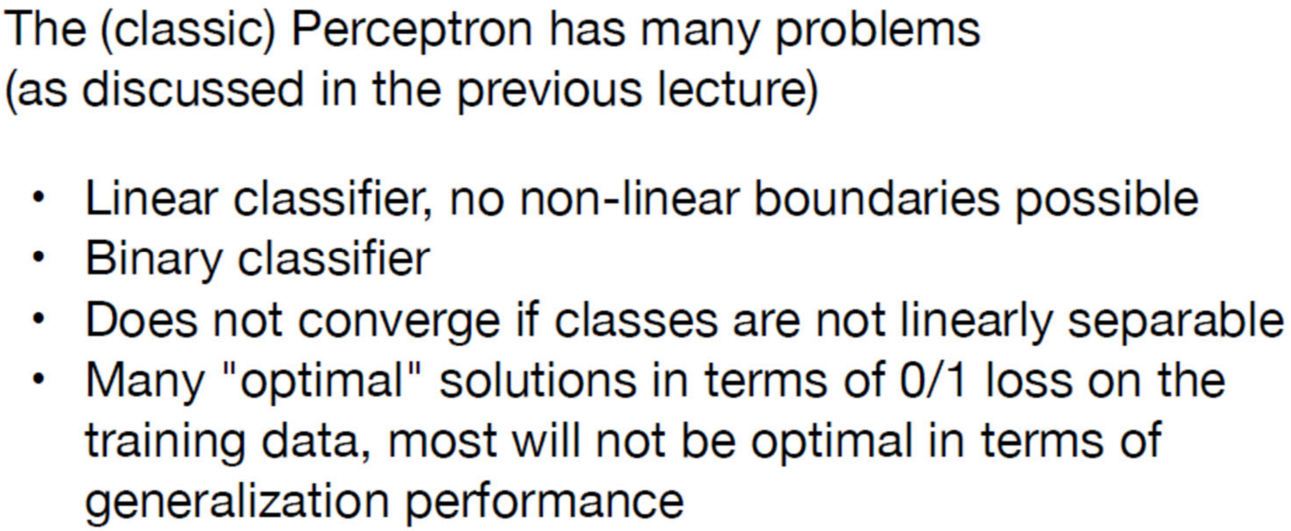


$$z = w_0 + \sum_{j=1}^{m} x_j \, w_j$$



$$z_i = w_{0,i} + \sum_{j=1}^{m} x_j \, w_{j,i}$$

Perceptron Conlcultion



The (classic) Perceptron has many problems
(as discussed in the previous lecture)

- Linear classifier, no non-linear boundaries possible
- Binary classifier
- Does not converge if classes are not linearly separable
- Many "optimal" solutions in terms of 0/1 loss on the training data, most will not be optimal in terms of generalization performance