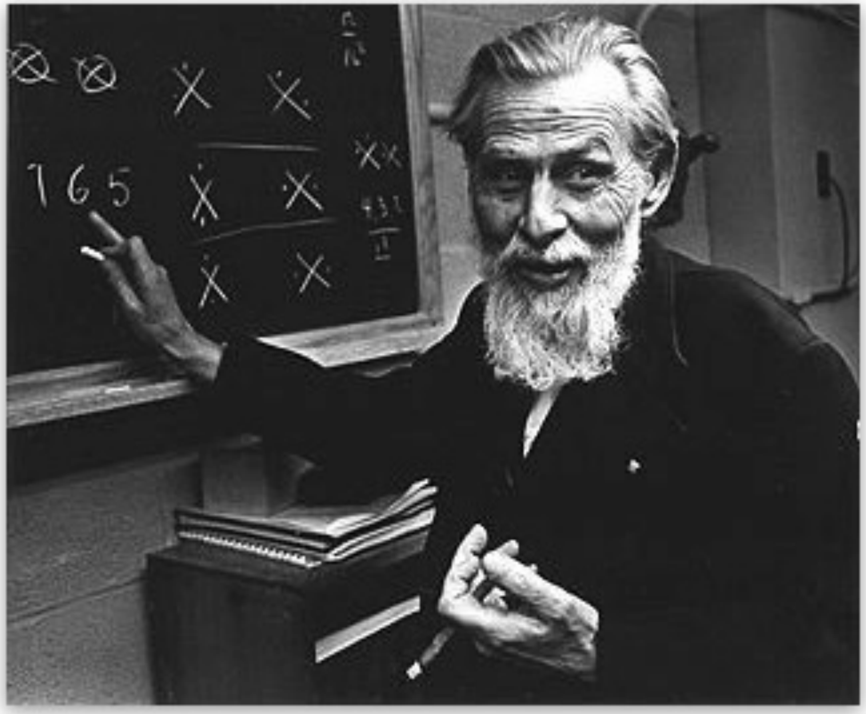
Perceptron – Intro to Artificial Neural Networks

Diagrams and content from A. Géron Hands on Machine Learning

https://www.amazon.fr/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1491962291

History of ANNs

- Artificial Neural networks have been around for a while…
- First introduced in 1943 by the neurophysiologist Warren McCulloch and the mathematician Walter Pitts.

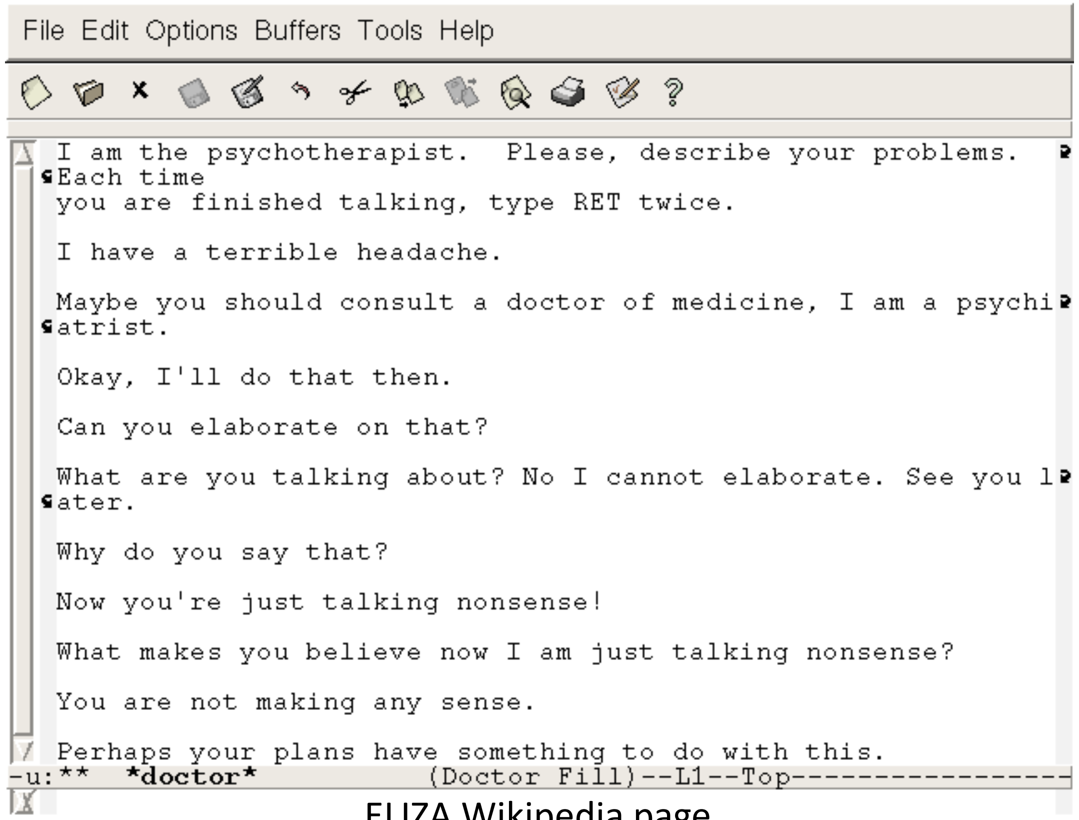


https://alchetron.com/Warren-Sturgis-McCulloch

History of ANNs

- Interest in ANNs came in waves :
  - 1960s : people thought we would soon be conversing with truly intelligent machines.
    - ELIZA, Weisenbaum 1964 :
  - When this wasn’t the case,

  funding went elsewhere…



ELIZA Wikipedia page

History of ANNs

- 1980s : revival of interest, new architectures invented, better training techniques, but progress was slow…
- by the 90s, other machine learning techniques were invented (SVMs) => seemed to offer better results and stronger theoretical foundations.

History of ANNs

- Present day : another wave of interest, but this one seems to be different :
  - Now a **huge quantity of data** available to train neural networks, and ANNs frequently **outperform** other ML techniques on very large and complex problems.
  - Tremendous increase in **computing power** since the 1990s now makes it possible to train large neural networks in a reasonable amount of time.
  - Training algorithms have been improved.
  - ANNs seem to have entered a virtuous circle of funding and progress.

Biological Neurons

- Biological neurons receive short electrical impulses called signals from other neurons via synapses.
- When a neuron **receives a sufficient number** of signals from other neurons, it **fires** its own signals.

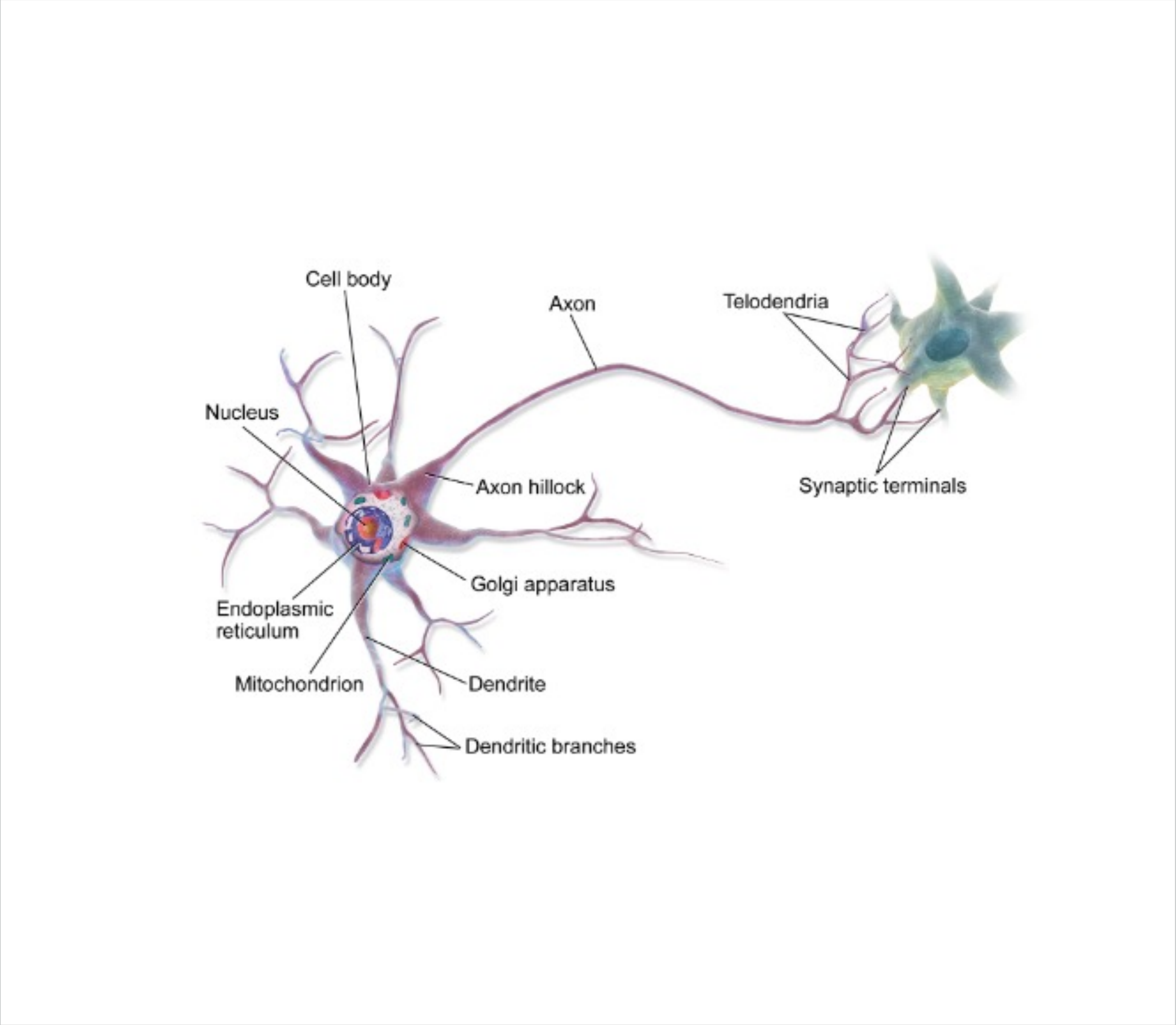
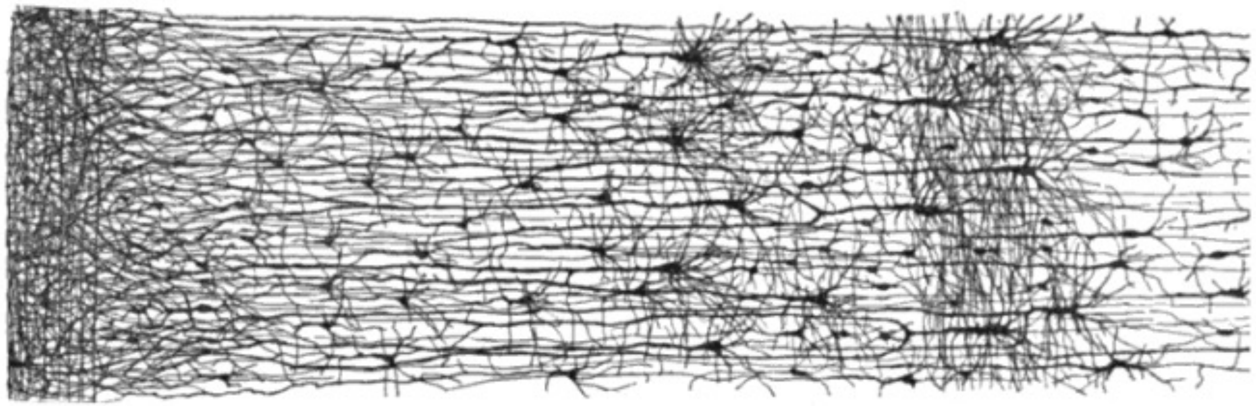


Image by Bruce Blaus (Creative Commons 3.0).
Reproduced from https://en.wikipedia.org/wiki/Neuron

Biological Neurons

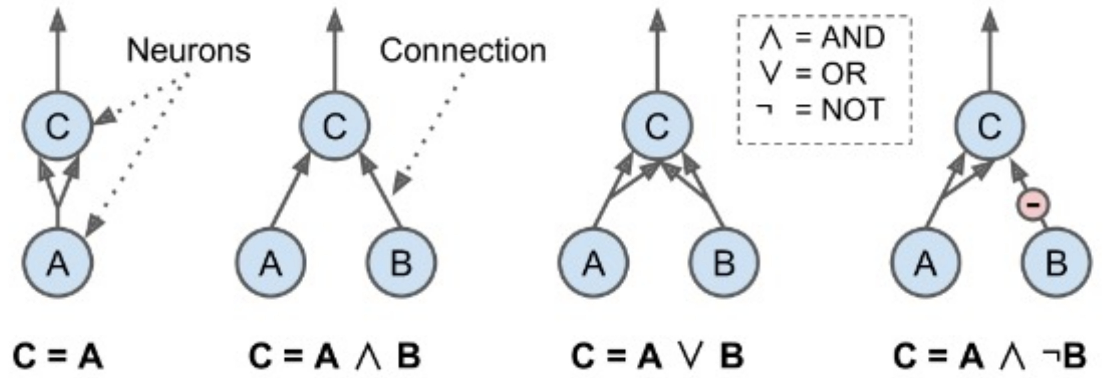
- Organized in a vast network of billions of neurons, each neuron is typically connected to thousands of other neurons.
- Research suggests that neurons are often organized in consecutive layers



Drawing of a cortical lamination by S. Ramon y Cajal (public domain)

Logical Computations with Neurons

- Warren McCulloch and Walter Pitts proposed a very simple model of the biological neuron
- It has one or more binary (on/off) inputs and one binary output.

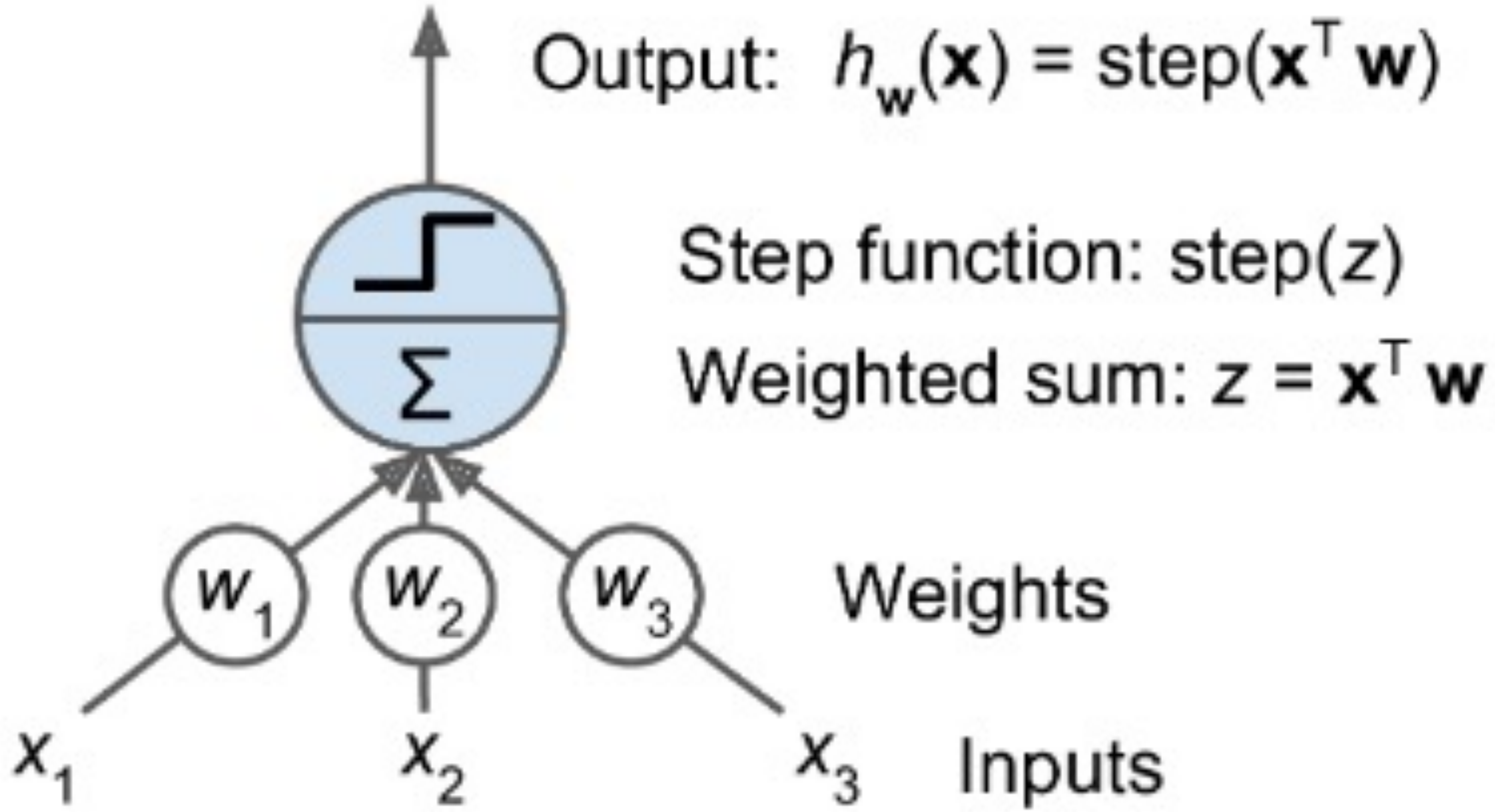


The Perceptron

- One of the simplest ANN architectures, invented in 1957 by Frank Rosenblatt

- These artificial neurons are slightly different and are called *Threshold Logic Units.* We will see why.

Threshold Logic Unit (TLU)



The Step Function

- Most commonly used is the *Heaviside Step Function :*

$$\text{heaviside}\ (z) = \begin{cases} 0 & \text{if}\ \ z < 0 \\ 1 & \text{if}\ \ z \ge 0 \end{cases}$$

- How does this single unit compare to logistic regression ?

Binary Classification

- A single TLU can be used for simple linear binary classification.

- It computes a linear combination of the inputs and if the result exceeds a threshold, it outputs the positive class or else outputs the negative class.

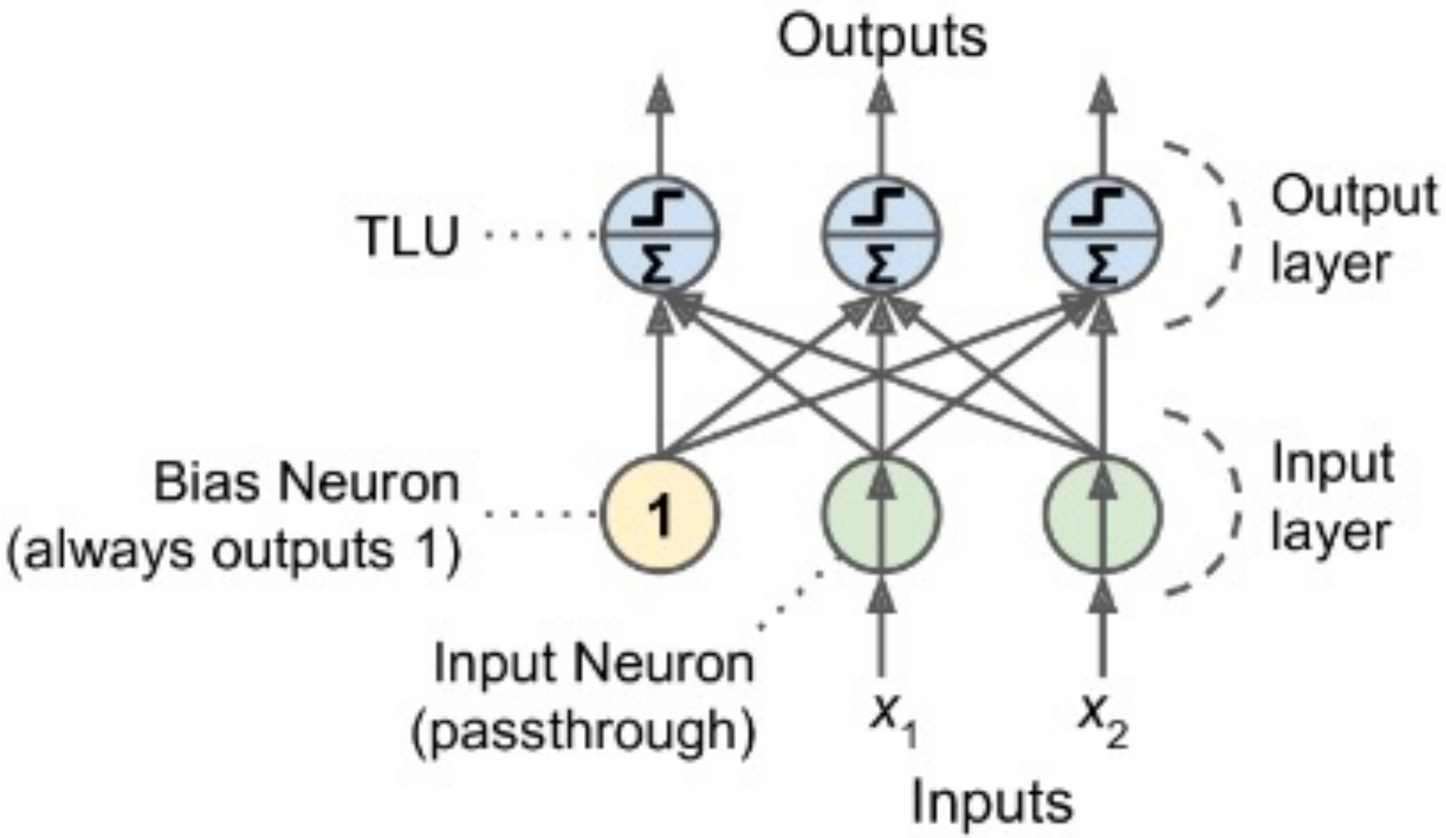
- Training a TLU in this case means finding the right values for w0, w1, and w2 (we will see the training algo in a few slides.)

Perceptron

- The Perceptron can refer to this single TLU, but also can refer to a single *layer* of TLUs.

- When all the neurons in a layer are **connected to every neuron** in the previous layer, it is called a ***fully connected*** *layer* or a ***dense*** *layer*.

Perceptron Diagram



Perceptron

- Common to draw special passthrough neurons called *input neurons*

- Also, an extra bias feature is usually added.

- This Perceptron can classify instances simultaneously into three different binary classes, which makes it a **multi-output** classifier

Computing the outputs of a fully connected layer

$$h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{X}) = activation(\boldsymbol{XW} + \boldsymbol{b})$$

- **X** represents the matrix of input features. It has one row per instance, one column per feature.

- The weight matrix **W** contains all the connection weights except for the ones from the bias neuron. It has one row per input neuron and one column per artificial neuron in the layer.

- The bias vector **b** contains all the connection weights between the bias neuron and the artificial neurons. It has one bias term per artificial neuron.

- The **activation** function: when the artificial neurons are TLUs, it is a step function. We will discuss other activation functions, the sigmoid function being one of them.

How is the Perceptron trained ?

- In his book *The Organization of Behavior*, published in 1949, Donald Hebb suggested that *when a biological neuron* ***often triggers another neuron***, the ***connection*** between these two neurons ***grows stronger*****.**
- “**Cells that fire together, wire together**.” (Siegrid Löwel)
- Perceptrons are trained using a variant of this rule that takes into account the error made by the network
- For every output neuron that produced a wrong prediction, it *reinforces the connection weights from the inputs that would have contributed to the correct prediction*.

Perceptron Learning Rule

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta\left(y_j - \hat{y}_j\right)x_i$$

- $w_{i,j}$ is the connection weight between the i-th input neuron and the j-th output neuron.
- x_i is the i-th input value of the current training instance.
- $\hat{y}_j$ is the output of the j-th output neuron for the current training instance.
- y_j is the target output of the j-th output neuron for the current training instance.
- η is the learning rate.

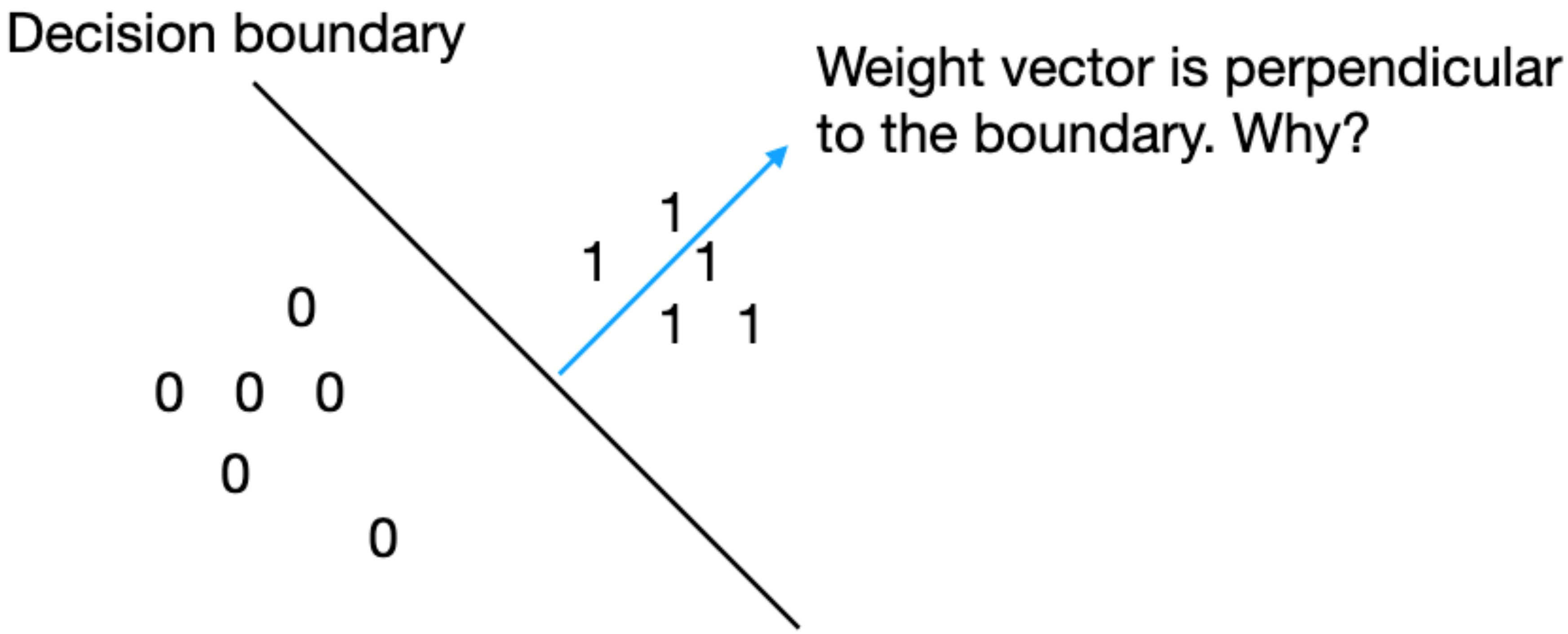
What can the Perceptron Learn ?

- Cannot learn complex patterns in the data

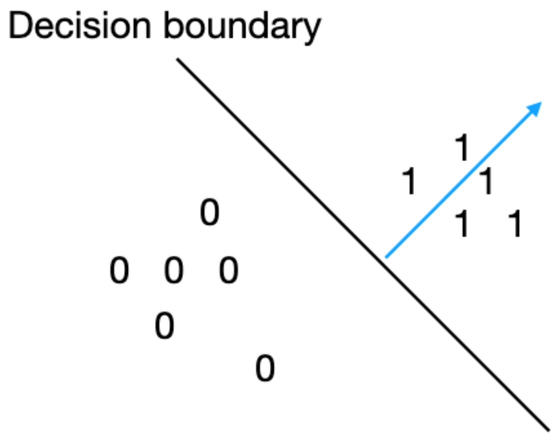
- However, if the training instances are linearly separable, Rosenblatt demonstrated that this algorithm would converge to a solution.

Geometric Intuition of the Learning Rule

(Sebastian Raschka : https://www.youtube.com/watch?v=Fj7BgxI73TA)



Geometric Intuition of the Learning Rule



Weight vector is perpendicular to the boundary. Why?

Remember,

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

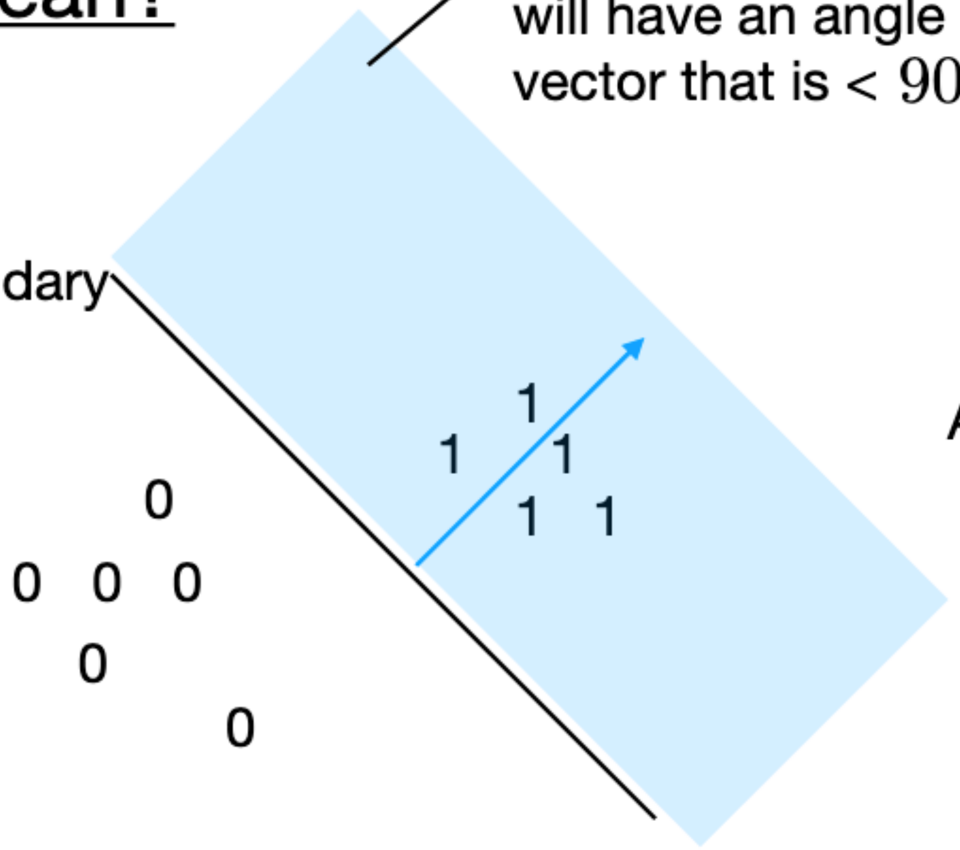
$$\mathbf{w}^T\mathbf{x} = ||\mathbf{w}|| \cdot ||\mathbf{x}|| \cdot \underbrace{\cos(\theta)}$$

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

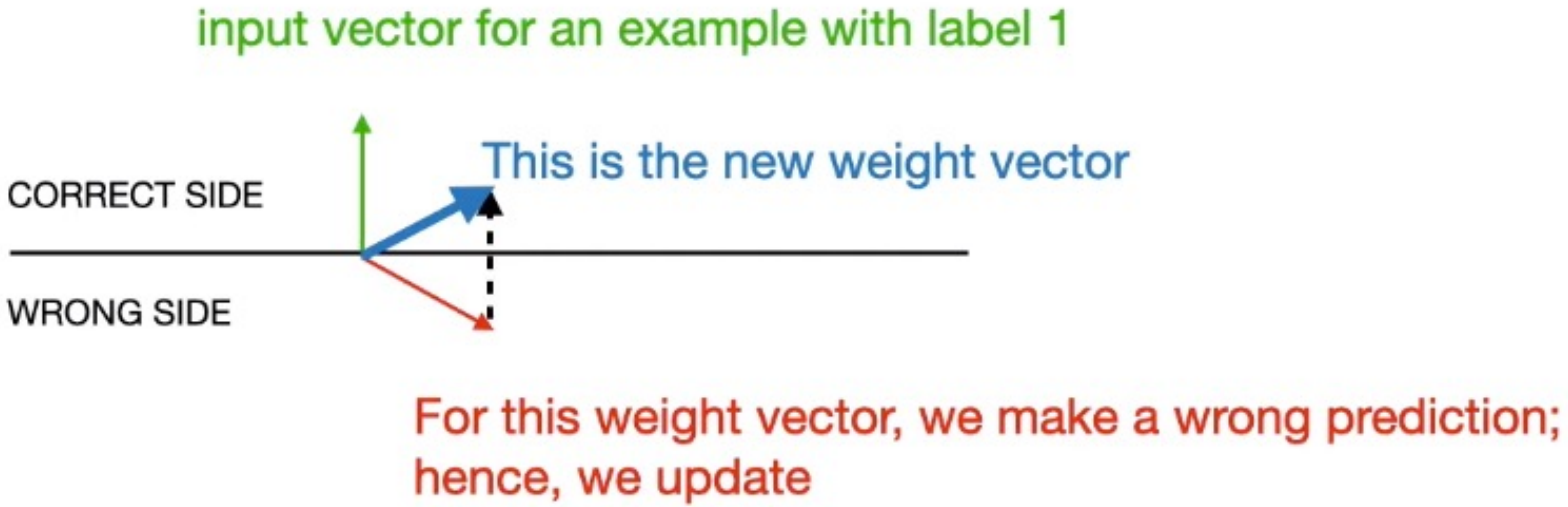
What else does this mean?

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
This is the new weight vector
CORRECT SIDE
WRONG SIDE
For this weight vector, we make a wrong prediction;
hence, we update

Sklearn Perceptron

- Scikit-Learn provides a Perceptron class that implements a **single TLU** network (iris dataset example):

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron

iris = load_iris()
X = iris.data[:, (2, 3)]  # petal length, petal width
y = (iris.target == 0).astype(np.int)  # Iris Setosa?

per_clf = Perceptron()
per_clf.fit(X, y)


y_pred = per_clf.predict([[2, 0.5]])
```