ML Training

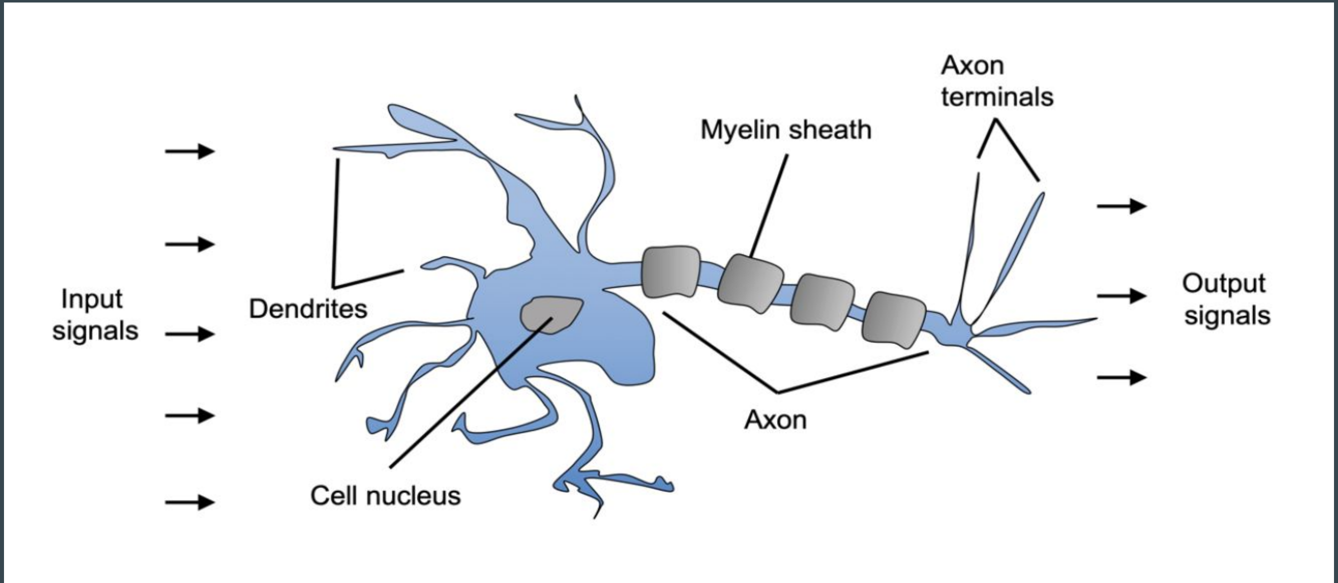
Classification

Algorithms

- Perceptron
- Adaptive Linear Neurons

Artificial Neurons

Warren McCulloch and Walter Pitts published the first concept of a simplified brain cell, the so-called McCulloch-Pitts (MCP) neuron, in 1943 (*A Logical Calculus of the Ideas Immanent in Nervous Activity* by *W. S. McCulloch* and *W. Pitts*, *Bulletin of Mathematical Biophysics*, 5(4): 115-133, 1943).



Biological Neuron

McCulloch and Pitts described nerve cell as

- A simple logic gate
- Binary outputs
- Multiple signals arrive at the dendrites
- Integrated into the cell body

If the accumulated signal **exceeds a certain threshold** , an output signal is generated that will be passed on by the axon.

The Perceptron

F. Rosenblatt, 1957

$z = w_1x_1 + w_2x_2 + \ldots + w_mx_m$

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

$$\sigma(z) = \begin{cases} 1 & \text{if } z \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

Linear Algebra

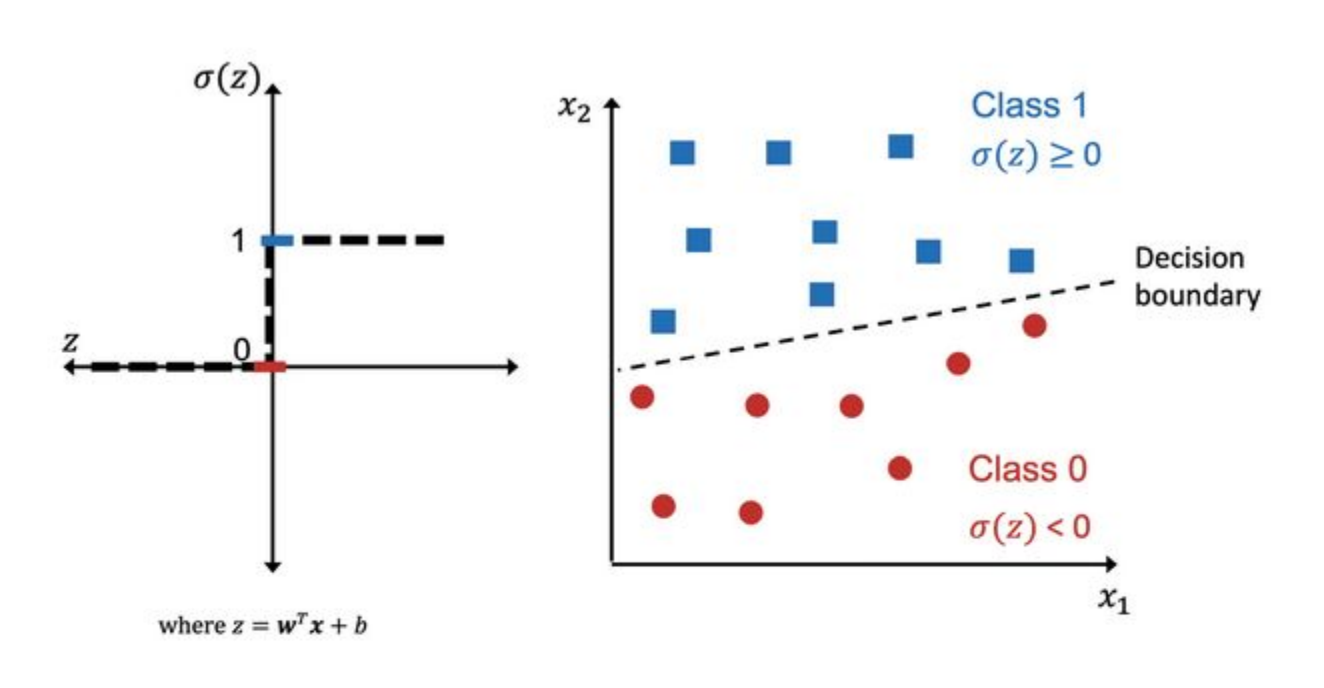
$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

$$\boldsymbol{a}^T \boldsymbol{b} = \sum_i a_i b_i = a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$$

http://www.cs.cmu.edu/~zkolter/course/linalg/linalg_notes.pdf (Linear Algebra Review and Reference)

The Bias Term



The Perceptron Learning Rule

- Initialize the weights and bias unit to 0 or small random numbers
- For each training example, $x^{(i)}$
  - Compute the output value ($\hat{Y}(i)$)
  - Update the weights and the bias unit

$$w_j := w_j + \Delta w_j$$
$$\text{and} \quad b := b + \Delta b$$

$$\Delta w_j = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)x_j^{(i)}$$
$$\text{and} \quad \Delta b = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)$$

η = Learning Rate (> 0.0, <= 1.0)

Which model to use?

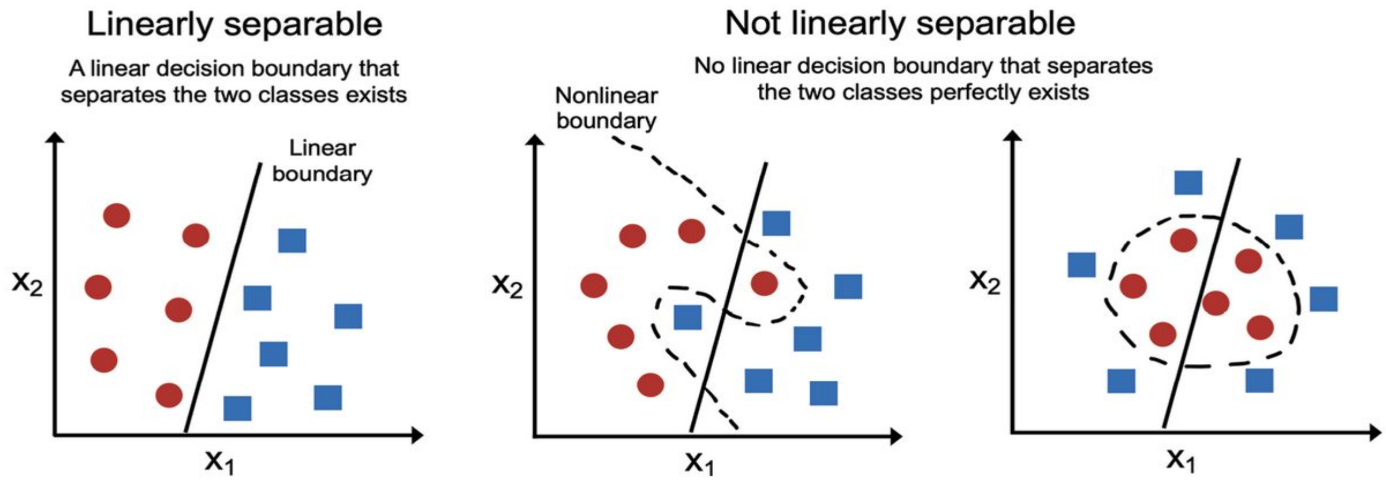
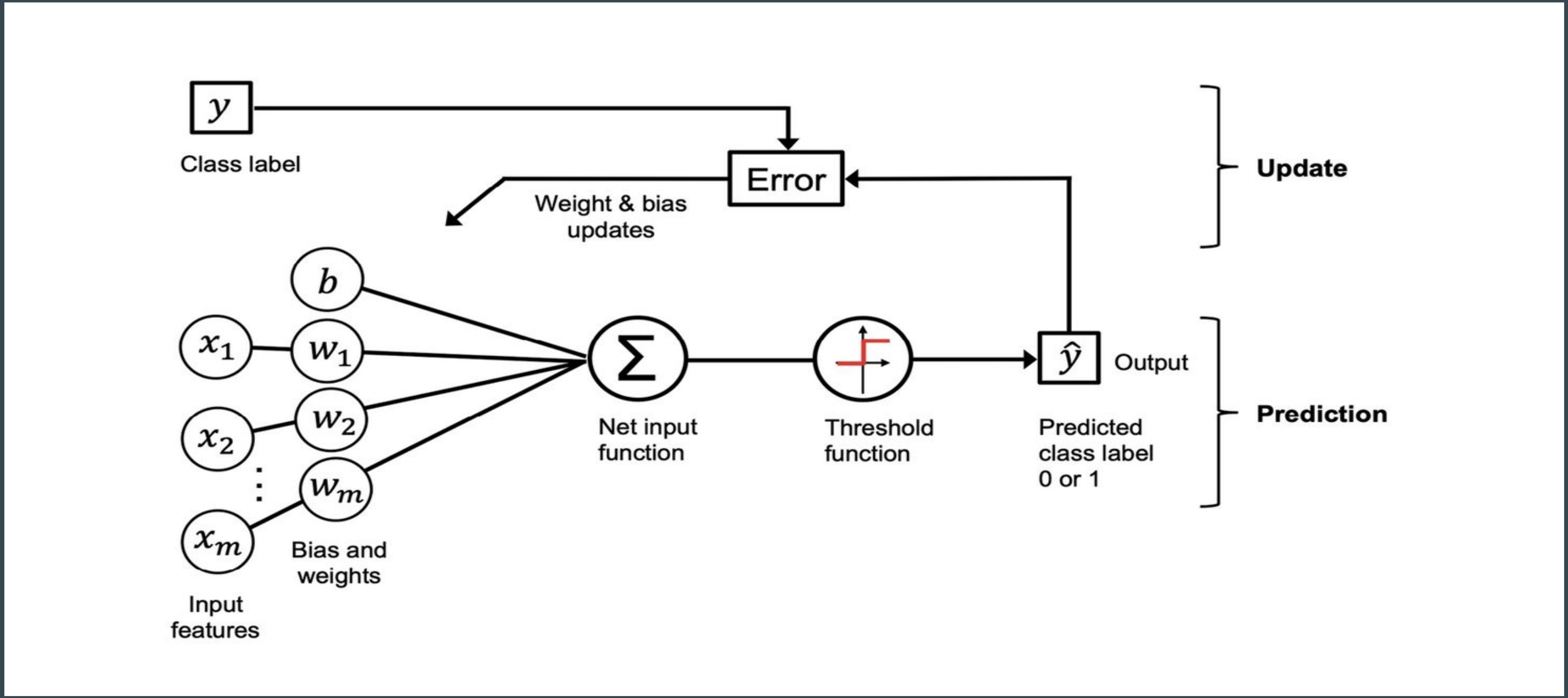


Figure 2.3: Examples of linearly and nonlinearly separable classes

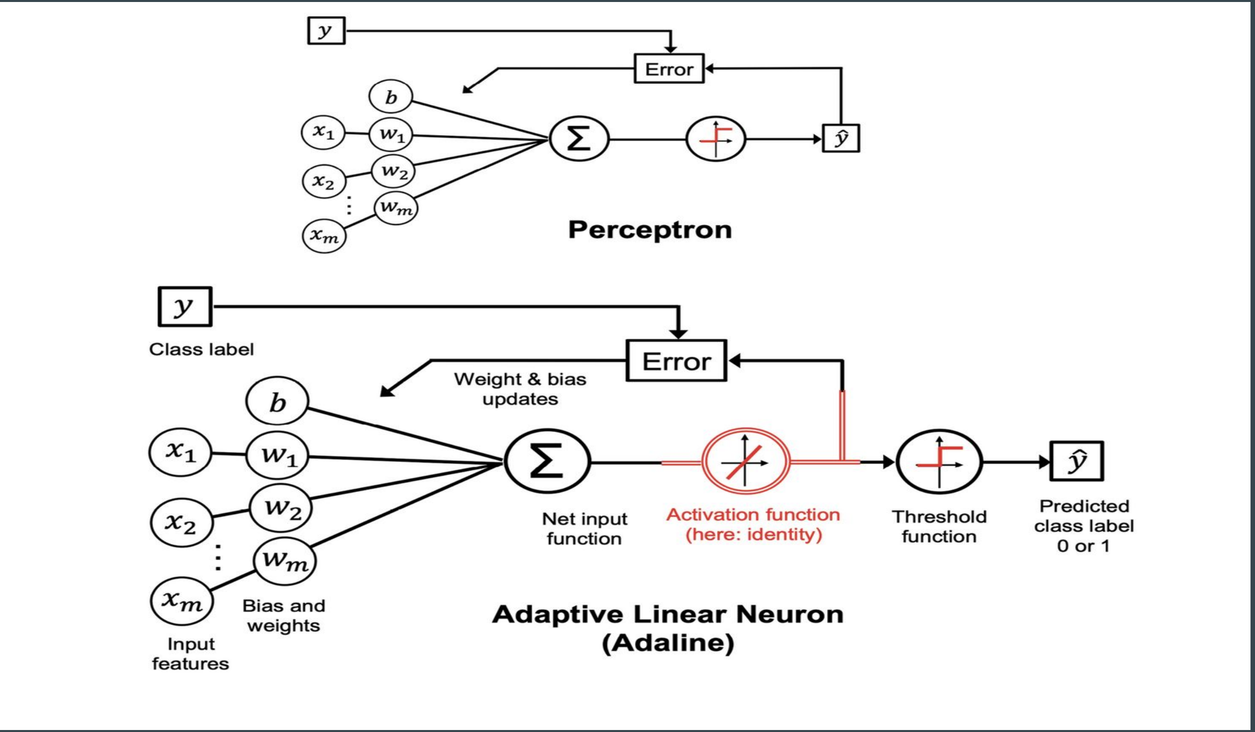
Training a Perceptron



NumPy Review

https://sebastianraschka.com/blog/2020/numpy-intro.html

ADaptive LInear NEuron (ADLINE)



Learning Rule

$$L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \sigma\left(z^{(i)}\right)\right)^2$$

$$\frac{\partial L}{\partial w_j} = -\frac{2}{n}\sum_{i}\left(y^{(i)} - \sigma\left(z^{(i)}\right)\right) x_j^{(i)}$$