Practical AI and ML

Francesco Ponzio, Department of Control and Computer Engineering, PoliTO
francesco.ponzio@polito.it

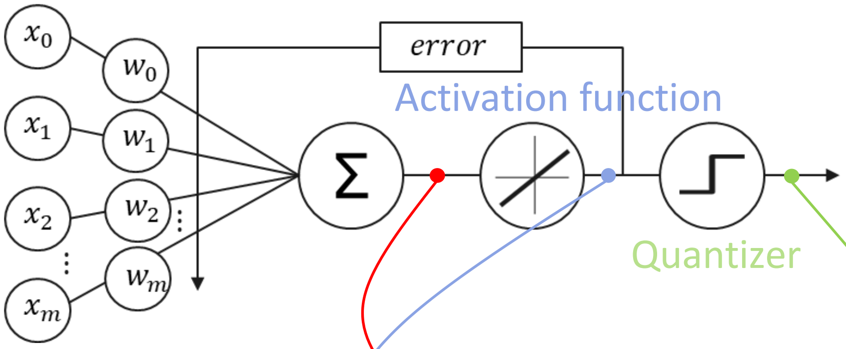
POLITECNICO
DI TORINO

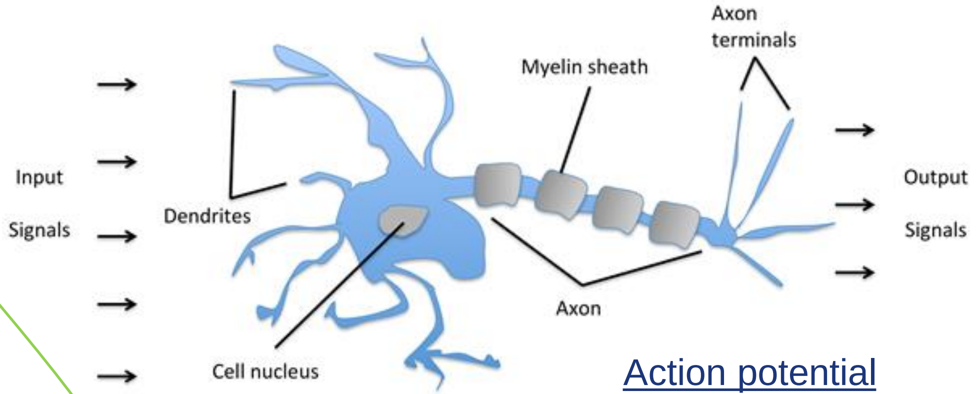
Perceptron

POLITECNICO
DI TORINO

Perceptron (Adaline) - Intro

- Multiple signals arrive at the dendrites and are integrated into the cell body.
- If the accumulated signal is >= a threshold θ, an output signal is generated.



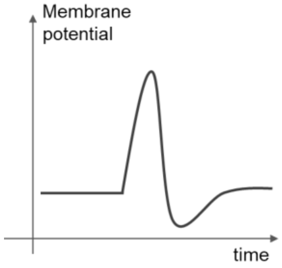


$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}^i$$

$$\varphi(z) = z$$

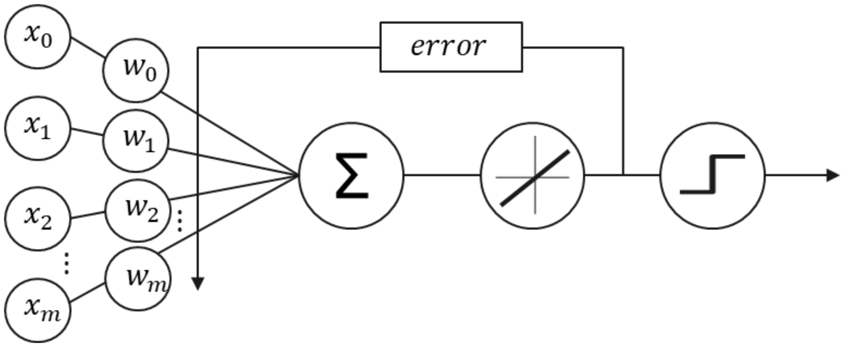
$$q(z) = \begin{cases} 1 \; if \; z \geq \theta \\ -1 \; otherwise \end{cases}$$

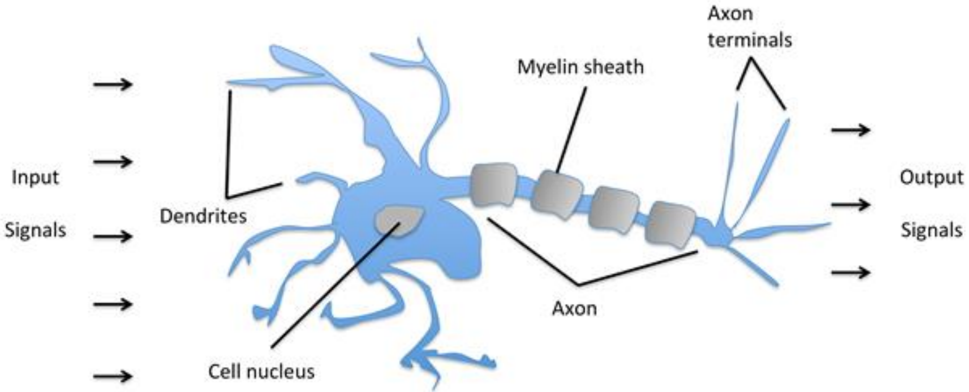
[Action potential]



Perceptron (Adaline) - Intro

- Multiple signals arrive at the dendrites and are integrated into the cell body.
- If the accumulated signal is >= a threshold θ, an output signal is generated.



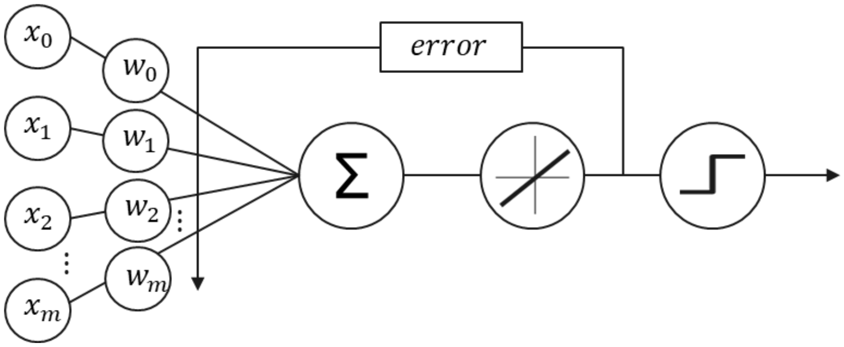


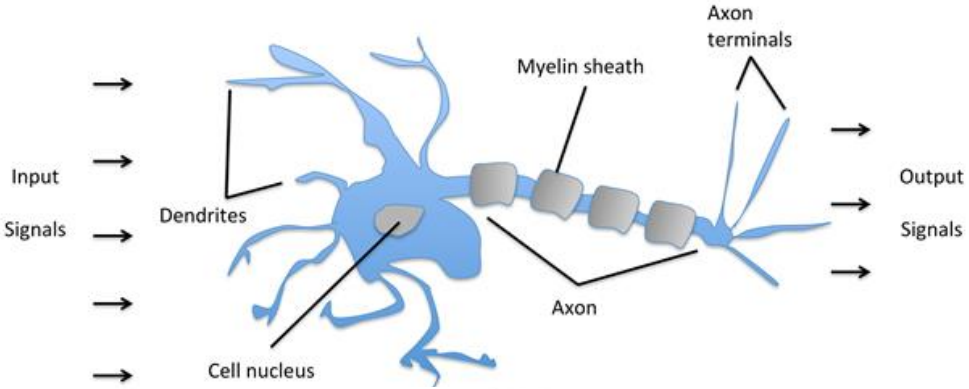
$with\ w_0 = -\theta$ and $x_0 = 1$

$$q(z) = \begin{cases} 1\ if\ z \geq \theta \\ -1\ otherwise \end{cases} \Rightarrow q(z) = \begin{cases} 1\ if\ z \geq 0 \\ -1\ otherwise \end{cases}$$

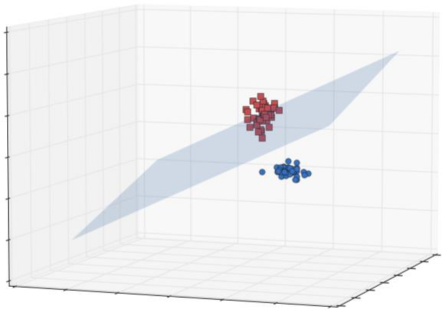
Perceptron (Adaline) - Intro

- Multiple signals arrive at the dendrites and are integrated into the cell body.
- If the accumulated signal is >= a threshold θ, an output signal is generated.





- Such disequation defines a **separation hyperplane**

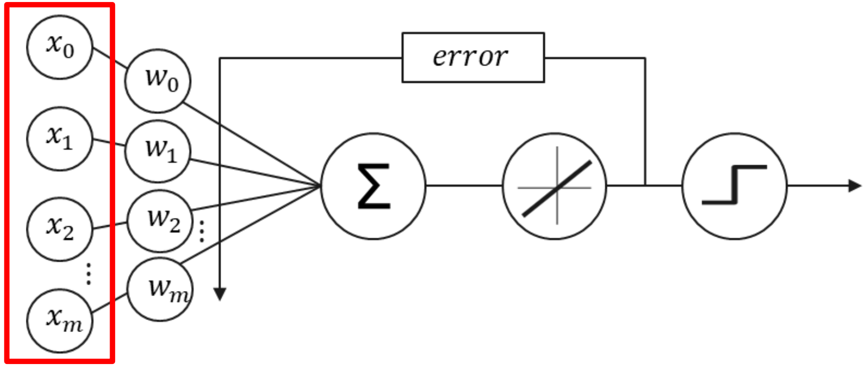


$$y(z) = \begin{cases} 1 \; if \; z \geq 0 \; \rightarrow \quad z \geq 0 \; \rightarrow w_1 x_1 + \ldots + w_m x_m - \theta \geq 0 \\ -1 \; otherwise \end{cases}$$

Perceptron (Adaline) - Notation

- Superscript i refers to the i-th training sample
- Subscript j refers to the j-th dimension (i.e., feature) of the given sample

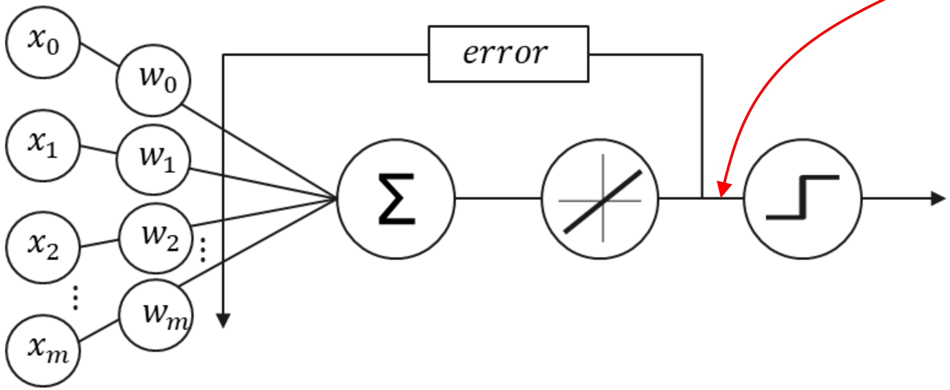
$$\begin{bmatrix} 1 & x_1^1 & x_2^1 & \dots & x_m^1 \\ 1 & x_1^2 & x_2^2 & \dots & x_m^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^n & x_2^n & \dots & x_m^n \end{bmatrix}$$



Single sample, m features

Perceptron (Adaline) - Learning

- Adaline uses **continuous predicted values** to **learn the model coefficients**.
- Given a training sample x^i, the corresponding target value t^i and output o^i we can define the **Sum of Squared Errors (SSE)** cost function $J(\boldsymbol{w})$



$$J(\boldsymbol{w}) = \frac{1}{2}\sum_i (t^i - o^i)^2 = \frac{1}{2}\sum_i (t^i - \varphi(z^i))^2 = \frac{1}{2}\sum_i (t^i - \boldsymbol{w}^T\boldsymbol{x}^i)^2$$

Perceptron (Adaline) - Learning

- We want to minimize $J(\boldsymbol{w}) = \frac{1}{2}\sum_i (t^i - o^i))^2 = \frac{1}{2}\sum_i (t^i - \boldsymbol{w}^T \boldsymbol{x}^i)^2$

- Using **gradient descent**, the idea is to update the weights by taking repeated steps in **the opposite direction of the gradient** of the cost function $J(\boldsymbol{w})$

$$\boldsymbol{w} \coloneqq \boldsymbol{w} + \Delta\boldsymbol{w} = \boldsymbol{w} - \eta \nabla J(\boldsymbol{w})$$

- Such step is multiplied by the learning rate

- The gradient can be approximated by computing the partial derivative of $J(\boldsymbol{w})$ with respect to the weights on each iteration:

$$\Delta w_j = \frac{\partial J}{\partial w_j} = \eta \sum_i (t^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)})(-x_j^{(i)}) = -\eta \sum_i (\boldsymbol{w}^T \boldsymbol{x}^{(i)} - t^{(i)}) x_j^{(i)}$$

Perceptron (Adaline) - Implementation

$$\Delta w_j = \frac{\partial J}{\partial w_j} = -\eta \underbrace{\sum_i}_{\text{sum\_i}} \underbrace{(\overbrace{\boldsymbol{w}^T\boldsymbol{x}^i}^{\text{input\_w}} - t^i)}_{\text{input\_e}} x_j^i$$

Note that:

1. Δw_j is weighted by the feature *j*
2. Δw_j depends on all the training samples (sum over *i*)
3. Tailoring η is crucial for an appropriate training