DAT200 – Applied Machine Learning I

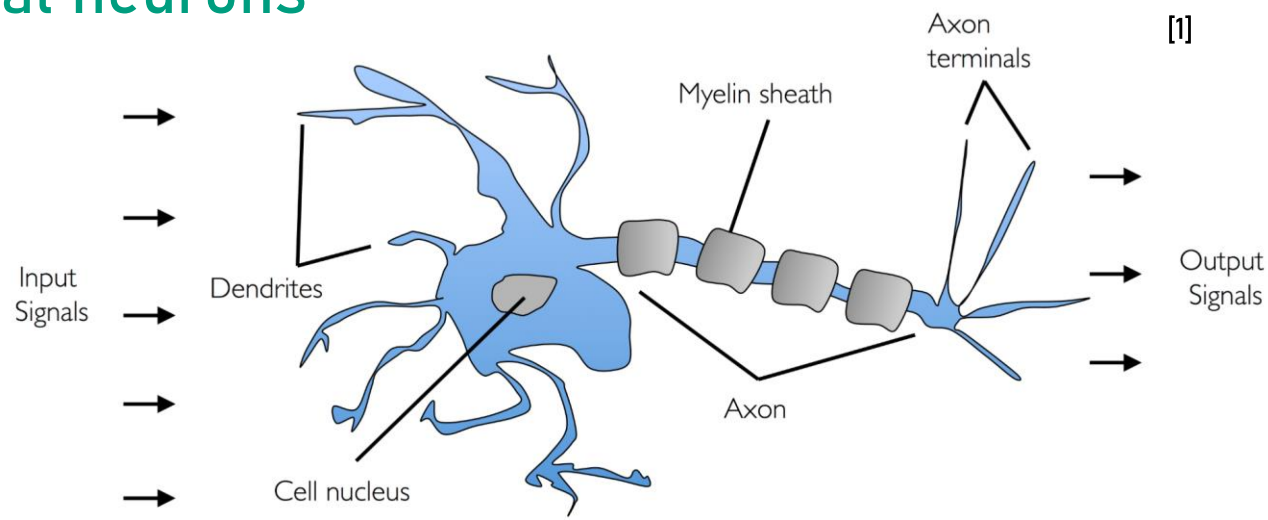
Chapter 2 in "Python Machine Learning" book

Training Simple Machine Learning Algorithms for Classification

Topics of Ch. 02 – Training Simple Machine Learning Algorithms for Classification

- Building intuition for ML algorithms
- Get to know two simple ML algorithms for classification
  - Perceptron
  - Adaline
- Basics of optimisation using adaptive linear neurons
- Read, process and visualise data with pandas, Numpy and Matplotlib
- Implement linear classification algorithms in Python

Artificial neurons



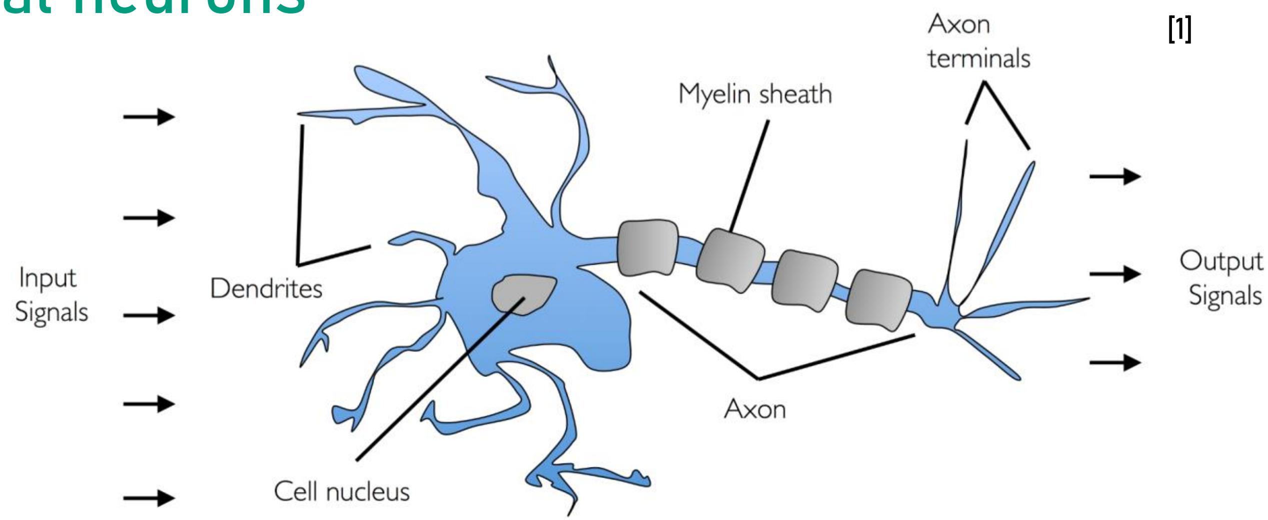
[1]

- Neurons are interconnected nerve cells in the brain
- Neurons are involved in processing and transmitting
  - Chemical signals
  - Electrical signals
- Neuron acts as a simple logic gate with binary output
- Multiple signals arrive at dendrites
- Signals are integrated into cell body
- If accumulated signal exceeds a specific threshold → output signal is generated and passed on to axon
- McCullock-Pitts (MCP) neuron: first simplified concept of brain cell (1943)

[1] Figure from S. Raschka, V. Mirjalili. *Python Machine learning – 3rd Ed.* (2019), page 20

Artificial neurons

[1]

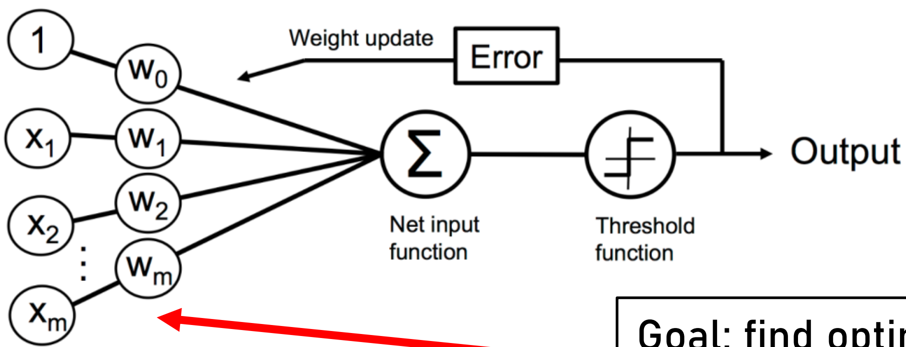


- Perceptron learning rule based on MCP neuron model published in 1957
- Algorithm learns automatically optimal weight coefficients for input features
- Product of optimal weight and input feature decides whether neuron fires or not
- Can be used to predict whether an instance belongs to one class or another

[1] Figure from S. Raschka, V. Mirjalili. *Python Machine learning – 3rd Ed.* (2019), page 20

General perceptron concept

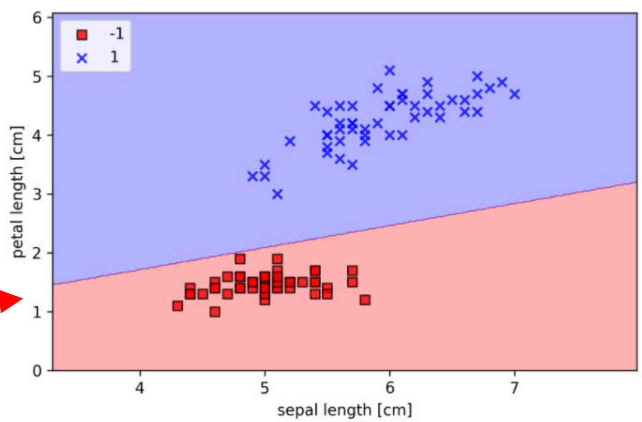
[1]



Goal: find optimal values for weights for best possible classification performance

training

Algorithm computes decision function based on weights and features



Note:
This plot is based on only two features in X

[1] Figure from S. Raschka, V. Mirjalili. *Python Machine learning – 3rd Ed.* (2019), page 26

Formal definition of an artificial neuron

- Binary classification task (two class problem)
  - First class coded as 1 (positive class)
  - Other class: -1 (negative class)

- Decision function ($\phi(z)$):
  - takes linear combinations of
    - values in vector $\boldsymbol{x}$
    - corresponding weights in weight vector $\boldsymbol{w}$
  - z: net input

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

m x 1 *m x 1*

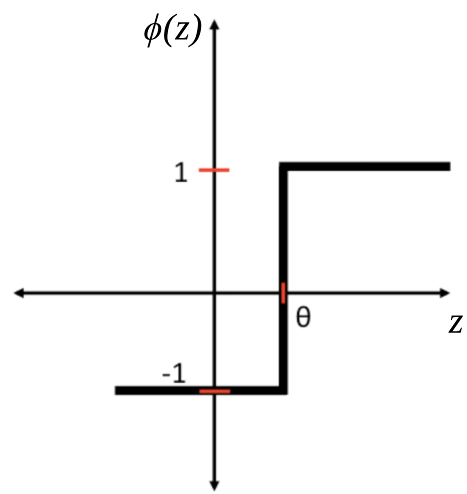
$$\boxed{z = w_1 x_1 + \ldots + w_m x_m}$$

Formal definition of an artificial neuron

- Given a specific sample $\boldsymbol{x}^{(i)}$
  - If net input $z \geq \theta$ → predict class 1 for $\boldsymbol{x}^{(i)}$
  - If net input $z < \theta$ → predict class -1 for $\boldsymbol{x}^{(i)}$
  - θ represents threshold: at which level/value should sum of all weighted input signals be to predict class 1?

- For perceptron algorithm $\phi(\cdot)$ is a variant of unit step function

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



Formal definition of an artificial neuron

- For simplicity bring θ to the left side of the equation

$$z \geq \theta$$

$$w_1 x_1 + w_2 x_2 + \cdots + w_m x_m \geq \theta$$

$$-\theta + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m \geq 0$$

$$-\theta \cdot 1 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m \geq 0$$

$$w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m \geq 0$$

where $w_0 = -\theta$

$x_0 = 1$

Formal definition of an artificial neuron

- z now can be written in a more compact form

$$\boxed{z = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \sum_{j=0}^{m} \boldsymbol{x}_j \boldsymbol{w}_j = \boldsymbol{w}^T \boldsymbol{x}}$$

1 x (m+1) *(m +1) x 1*

- and

$$\phi(z) = \begin{cases} 1 & if\ z \geq 0 \\ -1 & otherwise \end{cases}$$

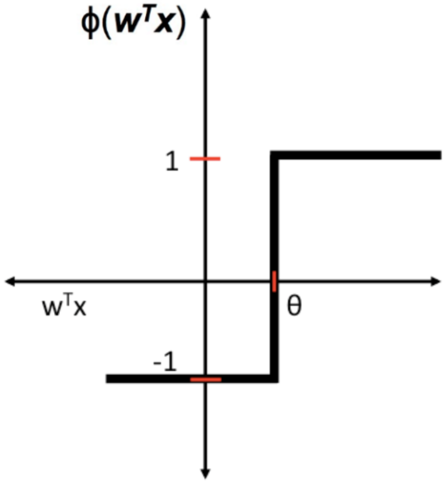
- In ML literature $w_0 = -\theta$ is often called bias unit
- Summarised:
  - from input values $\boldsymbol{x}$ and weights $\boldsymbol{w}$ → comupte net input z
  - from net input z and decision function $\phi(z)$ → to classification outputs -1 and 1

Formal definition of an artificial neuron

- The value of net input $z = \boldsymbol{w}^T\boldsymbol{x}$ decides whether decision function of the perceptron produces output 1 og -1.

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

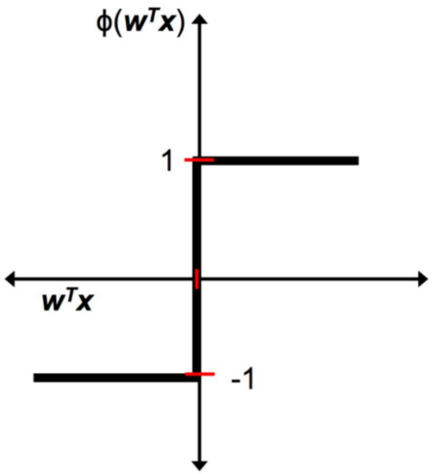
$$\phi(z) = \begin{cases} 1 & if\ z \geq 0 \\ -1 & otherwise \end{cases}$$



$$\boxed{z = w_1x_1 + \ldots + w_mx_m}$$

Include w_0x_0 in computation of z

[1]



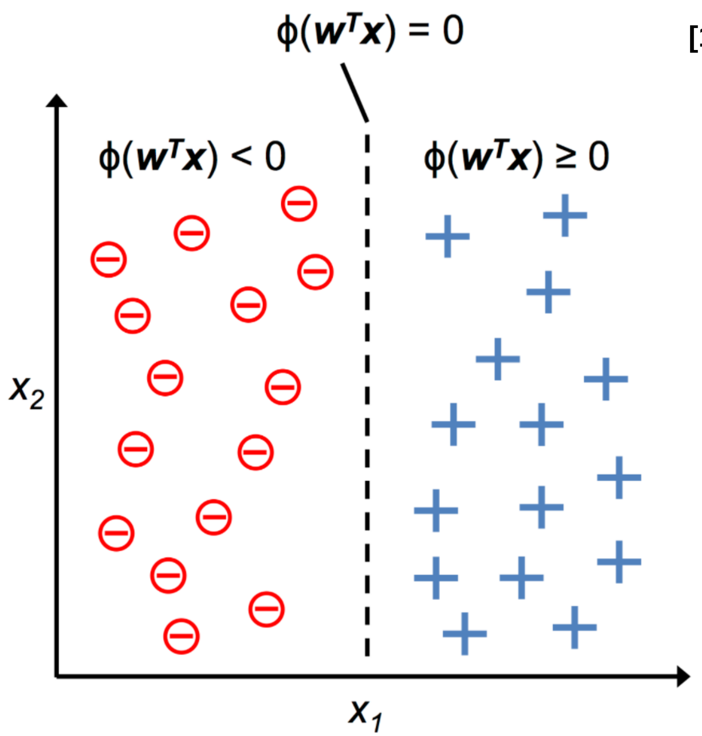
$$\boxed{z = w_0x_0 + w_1x_1 + w_2x_2 + \cdots + w_mx_m}$$

[1] Figure from S. Raschka, V. Mirjalili. *Python Machine learning – 3rd Ed.* (2019), page 23

Formal definition of an artificial neuron

- Example: the decision function discriminates between two linearly separable classes

[1]



[1] Figure from S. Raschka, V. Mirjalili. *Python Machine learning – 3rd Ed.* (2019), page 23

The perceptron learning rule

- Idea behind MCP neuron and Rosenblatt's thresholded perceptron model
  - → reductionist approach to mimic how a single neuron in the brain works
  - → the neuron either fires or not

- Initial perceptron rule
  1. Initialise the weights to 0 or small random numbers
  2. For each training sample $\boldsymbol{x}^{(i)}$
     a. Compute output value $\hat{y}$ (prediction of true class lable y)
     b. Compare true class lable y and predicted output $\hat{y}$
     c. If different from one another, update weights

How to update the weights?

The perceptron learning rule

- Weights w_j in weight vector $\boldsymbol{w}$ are updated simultaneously
- Update of each weight w_j can be more formally written as

$$\boxed{w_j := w_j + \Delta w_j}$$

- Value of Δw_j, which is used to update the weight w_j is calculated by the perceptron learning rule

$$\boxed{\Delta w_j = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)x_j^{(i)}}$$

η : Learning rate (typically contant between 0.0 and 1.0)

$y^{(i)}$: true class lable

$\hat{y}^{(i)}$: predicted class lable

$x_j^{(i)}$: value of feature j in sample vector i

The perceptron learning rule

- For a two-dimensional (two variables: x_1 and x_2) data set

$$\Delta w_0 = \eta \left(y^{(i)} - output^{(i)} \right)$$

$$\Delta w_1 = \eta \left(y^{(i)} - output^{(i)} \right) x_1^{(i)}$$

$$\Delta w_2 = \eta \left(y^{(i)} - output^{(i)} \right) x_2^{(i)}$$

The perceptron learning rule: example computations

- Correct class lable predictions
  - Negative class predicted correctly $\Delta w_j = \eta\left(-1-\left(-1\right)\right)x_j^{(i)} = 0$
  - Positive class predicted correctly $\Delta w_j = \eta\left(1-1\right)x_j^{(i)} = 0$
- Incorrect class lable predictions
  - Positivie class predicted incorrectly $\Delta w_j = \eta\left(1--1\right)x_j^{(i)} = \eta\left(2\right)x_j^{(i)}$
  - Negative class predicted incorrectly $\Delta w_j = \eta\left(-1-1\right)x_j^{(i)} = \eta\left(-2\right)x_j^{(i)}$

The perceptron learning rule: example computations

- Intuition for multiplicative factor $x_j^{(i)}$ in $\boxed{\Delta w_j = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)x_j^{(i)}}$

- Example $\hat{y}^{(i)} = -1, \quad y^{(i)} = +1, \quad \eta = 1$

  - assume: $x_j^{(i)} = 0.5$ and sample j has been misclassified as -1
  - Consequently, weight update will be $\Delta w_j = (1 - -1)0.5 = (2)0.5 = 1$
  - We see that weight update Δw_j is proportinal to value of $x_j^{(i)}$

  - Now assume: $x_j^{(i)} = 2$ and sample j has been misclassified as -1
  - Consequently, weight update will be $\Delta w_j = (1 - -1)2 = (2)2 = 4$

Important notes on perceptron algorithm

- Convergence of perceptron algorithm guaranteed only if
  - Two classes are linearly separable
  - Learning rate is sufficiently small

- If classes cannot be separated by a linear decision boundary take at least one of the following two measures
  - Set maximum number of iterations (epochs) over training samples
  - Set threshold for maximum number of tolerated misclassifications

- If classes not linearly separable AND none of the two measures above were taken → perceptron never stops updating weights