Chapter 10

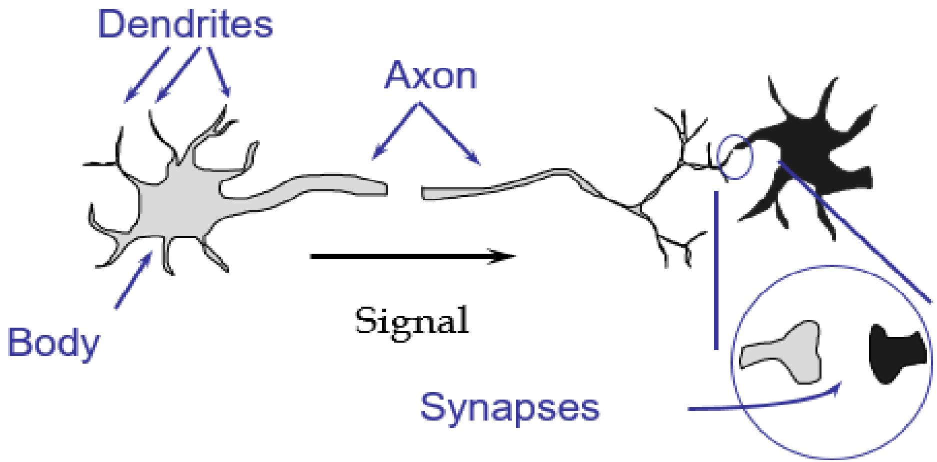
Artificial Neural Networks

Artificial neural networks

- Distributed systems inspired by the nervous system, in particular, the human brain
  - Composed of multiple processing units ("neurons")
  - Connected by a large number of connections ("synapses")
- Use learning algorithms based on how animals learn
  - Learn by adjusting values associated with the network connections

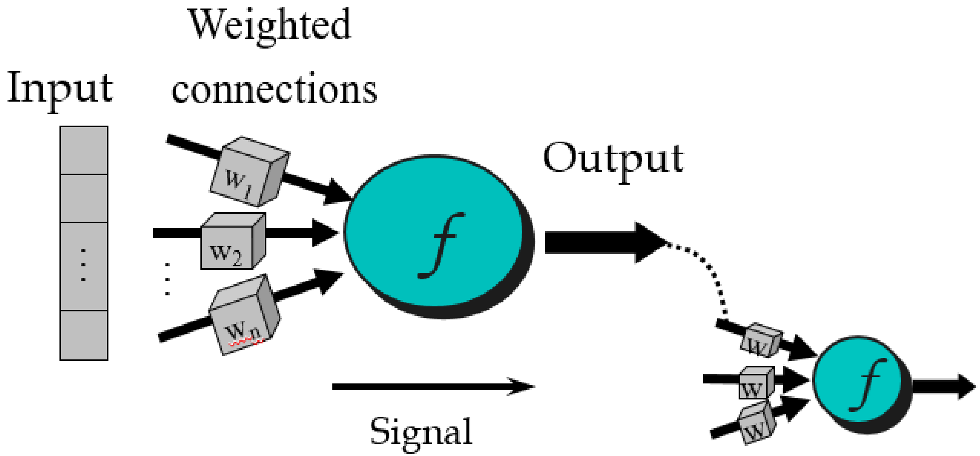
Artificial neural networks

- **Biological neurons**



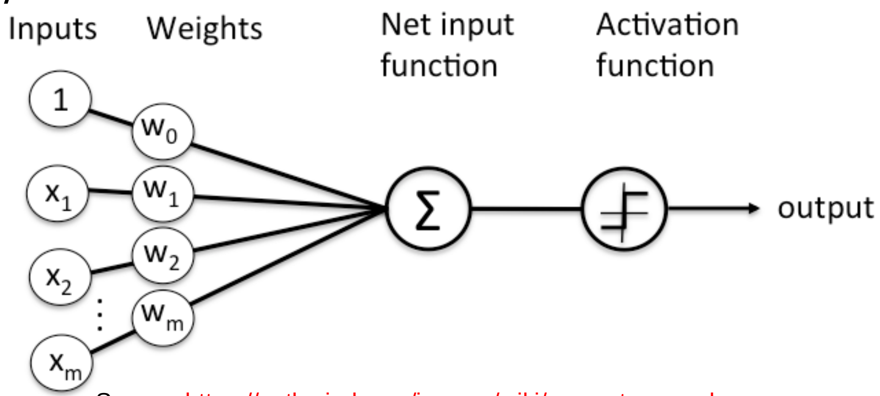
Artificial neural networks

- **Artificial neurons**



Artificial neural networks

- **A single Neuron**
  - Takes m inputs. For each input it has a multiplier (*weight*) associated
  - Has a input *bias* also
  - Calculates the sum of the product of inputs and associated weights
  - Output is calculated by feeding the weighted sum of inputs to the *activation function*



Source: https://pathmind.com/images/wiki/perceptron_node.png

Artificial neural networks

- **Activation functions**

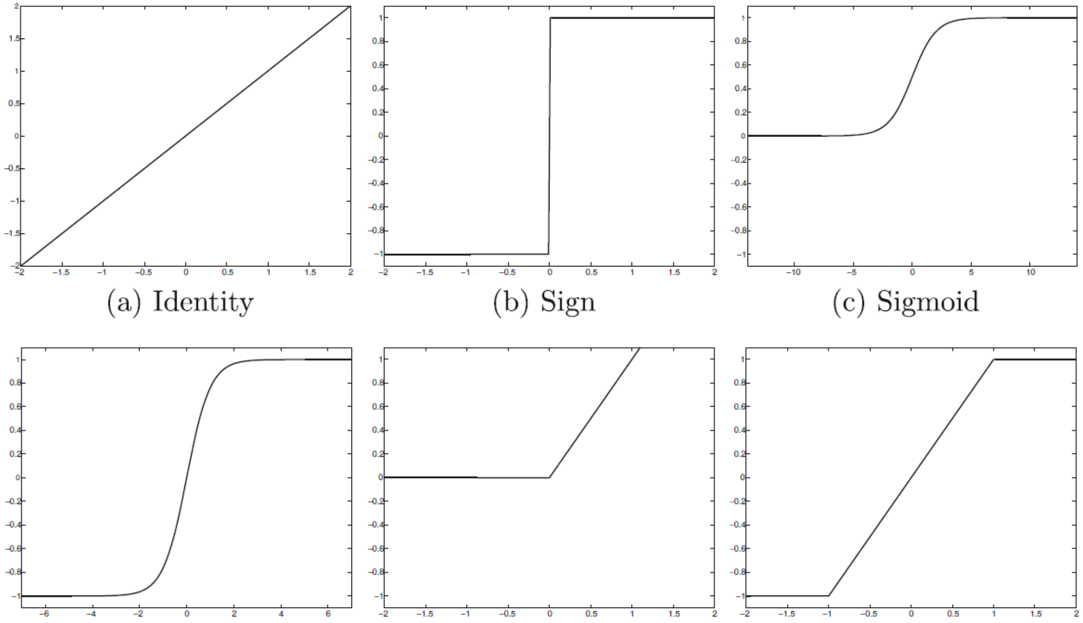
$$\Phi(v) = \text{sign}(v) \text{ (sign function)}$$

$$\Phi(v) = \frac{1}{1+e^{-v}} \text{ (sigmoid function)}$$

$$\Phi(v) = \frac{e^{2v}-1}{e^{2v}+1} \text{ (tanh function)}$$

$$\Phi(v) = \max\{v, 0\} \text{ (Rectified Linear Unit [ReLU])}$$

$$\Phi(v) = \max\left\{\min\left[v, 1\right], -1\right\} \text{ (hard tanh)}$$



(a) Identity (b) Sign (c) Sigmoid

(d) Tanh (e) ReLU (f) Hard Tanh

Source: Neural Networks and Deep Learning by Charu C. Aggarwal

Artificial neural networks

- **Training**
  - Network weight values are defined by a learning algorithm
    - Update the weight values according to an *objective function* or *loss function* (for example MSE)
  - Optimizes, for each neuron, objective function parameters (neural weights) in order to minimize the predictive error
    - Difference between the neuron output value produced by the *activation function* and the desired output value

Artificial neural networks

- **Perceptron**
  - First implemented artificial neural network
  - Architecture: one artificial neuron with input connections connected to the input data
  - Learning: update the network weights to reduce the classification mistakes made by the network
  - Only linearly separable problems
  - Convergence theorem: if the network can learn, it will learn

Artificial neural networks

Algorithm Perceptron training.

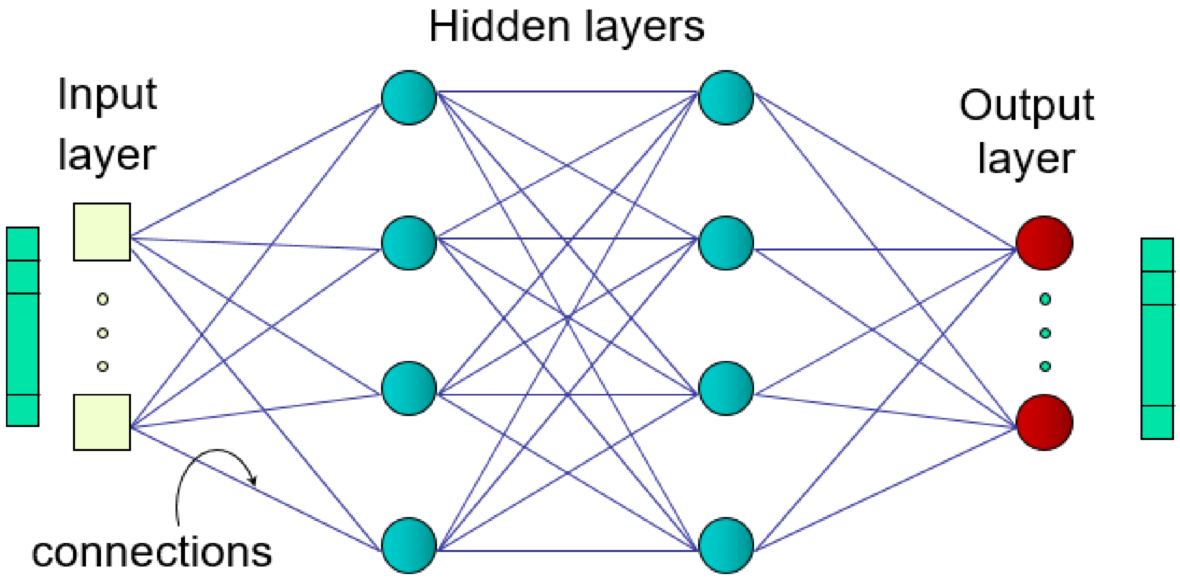
INPUT D_{train} training set
INPUT x_i object in the training set
INPUT y_i neuron predicted output value for the object x_i
INPUT d_i neuron desired output value for the object x_i
INPUT n the number of objects in the training set
INPUT m the number of predictive attributes of the objects
Define the initial weights with the value 0
repeat
    Initialize Errors with the value 0
    for all objects x_i in D_{train} **do**
        Calculate the neuron output y_i when the neuron receives as input x_i
        if $y_i \neq d_i$ **then**
            Update the m weights of x_i
until There are no differences (prediction errors)

Artificial neural networks

- **Multi-layer perceptron (MLP)**
  - Architecture: more than one layer of neurons, each layer can have more than one neuron
  - Each node from a layer feeds its output to all neurons in the next layer (Feed Forward Network)
  - Learning: update the network weights to reduce the classification mistakes made by the network
  - Originally trained by the backpropagation algorithm
  - Can solve any classification or regression problem
  - No convergence theorem

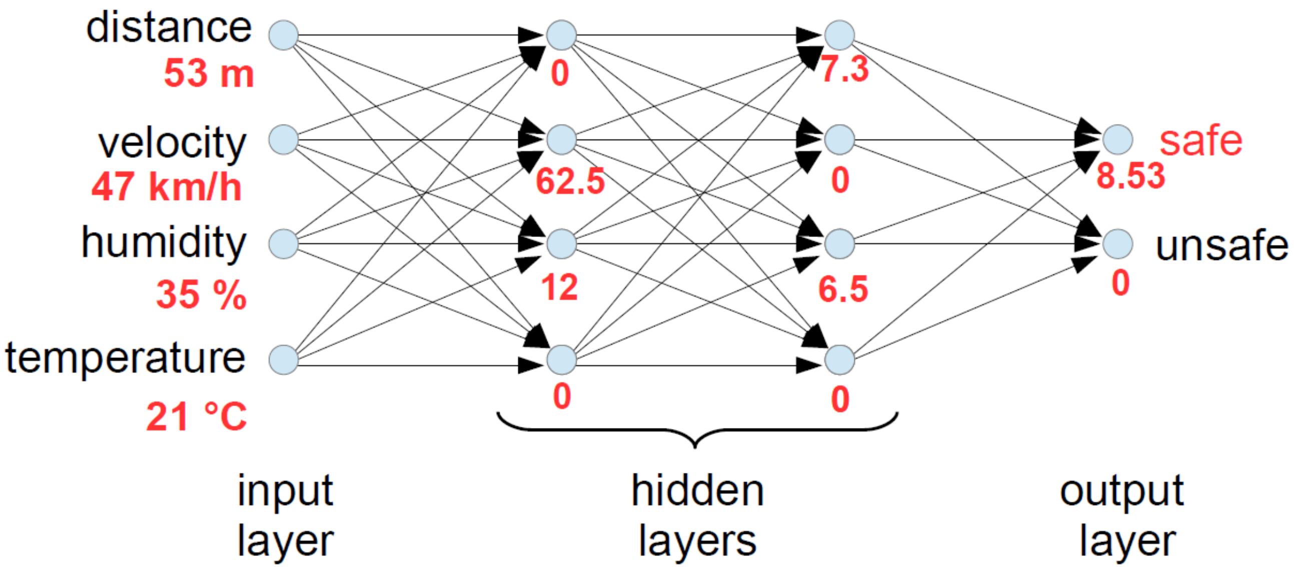
Artificial neural networks

- **Multi-layer perceptron (MLP)**



Artificial neural networks

- **Example**
  - Decide if current state of self driving car is safe or not



Artificial neural networks

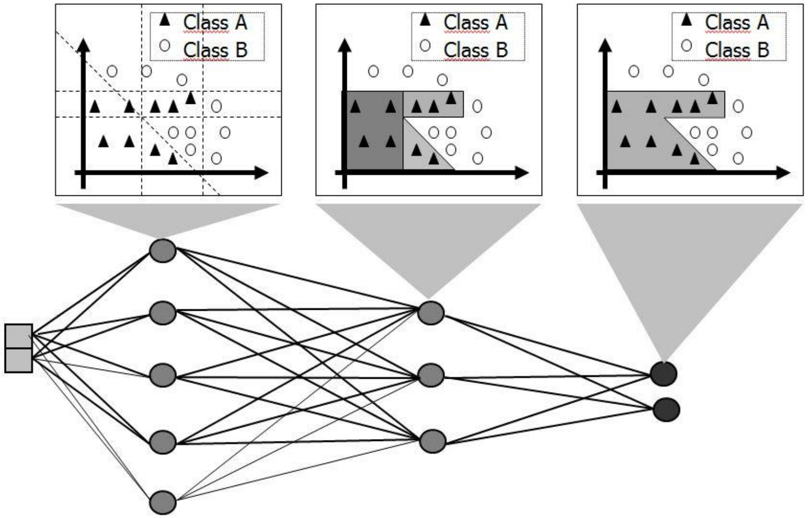
- **Backpropagation**

Algorithm Backpropagation training for MLP networks.

INPUT D_{train} training set
INPUT Y_{true} true output vector
INPUT Y_{pred} predicted output vector
INPUT x_i object in the training set
INPUT y_i neuron predicted output value for the object x_i
INPUT d_i neuron desired output value for the object x_i
INPUT n the number of objects in the training set
Initialize Errors with random values in the range $[-0.5, +0.5]$
repeat
    Initialize $\text{Error}_{Total} = 0$
    for all objects x_i in D_{train} **do**
        for all network layer, starting in the first hidden layer, forward **do**
            for all neuron in the current layer **do**
                Calculate the neuron output y_i when the neuron receives as input x_i or the output from the neurons in the previous layer
        $\text{Error}_{Partial} = Y_{true} - Y_{pred}$
        $\text{Error}_{Total} = \text{Error}_{Total} + \text{Error}_{Partial}$
        for all network layer, starting in the output layer, backward **do**
            for all neuron in the current layer **do**
                if error > 0 **then**
                    Update weight values
until Predictive performance is acceptable

Artificial neural networks

- **Layers**
  - As we move deeper in the network layers the more complex structures each neuron can recognize
  - This allows us to teach networks to recognize abstract conceptual patterns like cats, dogs, stock price movements, text semantics, etc.



Artificial neural networks

- Most classification tasks solved by MLP networks usually have 1 or 2 hidden layers
- A network with 1 hidden layer, with enough neurons, can approximate any multivariate continuous function
  - Shallow network
- A network with 2 hidden layers, under certain conditions, can learn any function
- A network with more than 1 hidden layer is a deep network

Artificial neural networks

- **Pros**
  - Good predictive performance in many real problems
  - (Near?) Human level problem solving in pattern recognition problems
  - Can be easily applied to multiclass and multilabel classification tasks
  - Similar to the structure and functioning of the nervous system
  - Very robust in the presence of noise, outliers
  - Networks will converge even without normalizing the data (however convergence will be slower)
- **Cons**
  - Models are of difficult interpretation
  - Training usually has a high computational cost
  - Lack of strong mathematical foundation

References, Literature, further reading

- Relevant sections from **A General Introduction to Data Analytics** by João Mendes Moreira, André C. P. L. F. de Carvalho and Tomáš Horváth
- **Neural Networks and Deep Learning** by Charu C. Aggarwal
- https://www.deeplearningbook.org/
- http://biointelligence.hu/pdf/02-from-linear-regression-to-deep-learning.pdf

Questions?