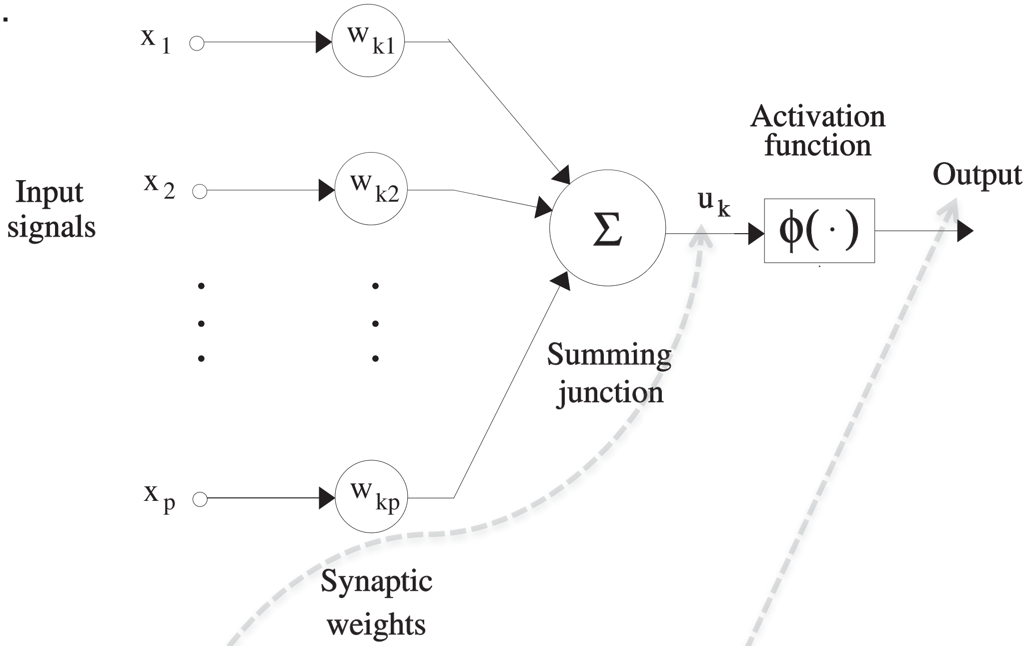
Computational Intelligence I: Neural Networks

Chapter 2: Structural Aspects

Types of neurons

⌘ Each neuron is an information processing unit fundamental to the entire operation of the ANN. The basic elements of a typical neuron are:

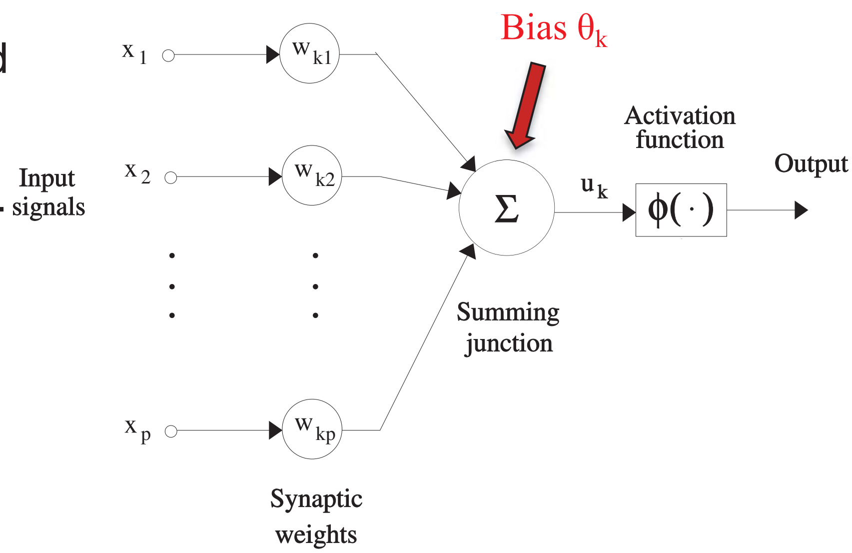
- ◎ A set of **synapses or connections**. Each of these is characterised by a weight (strength). The j^{th} synapse connected to the k^{th} neuron receives signal x_j and multiplies it by w_{kj}.
- ◎ An **adder** for summing the input signals, weighted by the respective synapses. The combined synaptic signals are denoted by u_k.
- ◎ An **activation function** ϕ, which squashes the permissible amplitude range of the output signal. The final output is denoted by y_k.



⌘ The mathematical representation of the above diagrammatic representation of a neuron is:

$$u_k = \sum_{j=1}^{p} w_{kj} x_j = \mathbf{w}_k^T \mathbf{x}, \qquad y_k = \phi(u_k)$$

⌘ The externally applied bias θ_k decreases or increases the internal input to the activation.

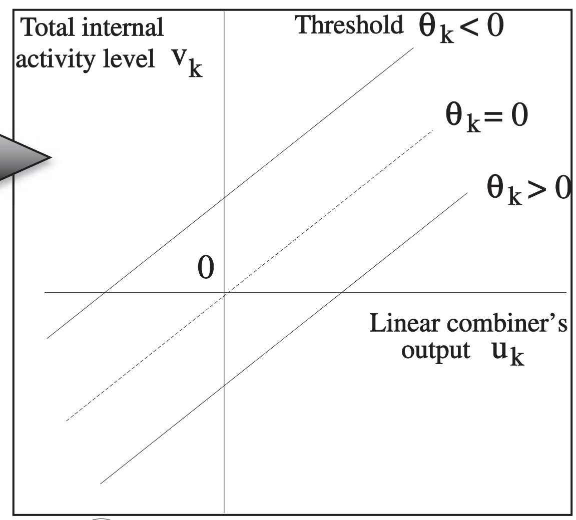


⌘ Thus, the output becomes:

$$\boxed{y_k = \phi\left(\mathbf{w}_k^T \mathbf{x} - \theta_k\right)}$$

where, $v_k=u_k-\theta_k$

⌘ Overall, the bias applies an affine transformation to the output u_k and modifies the **induced local field** or activation potential v_k.



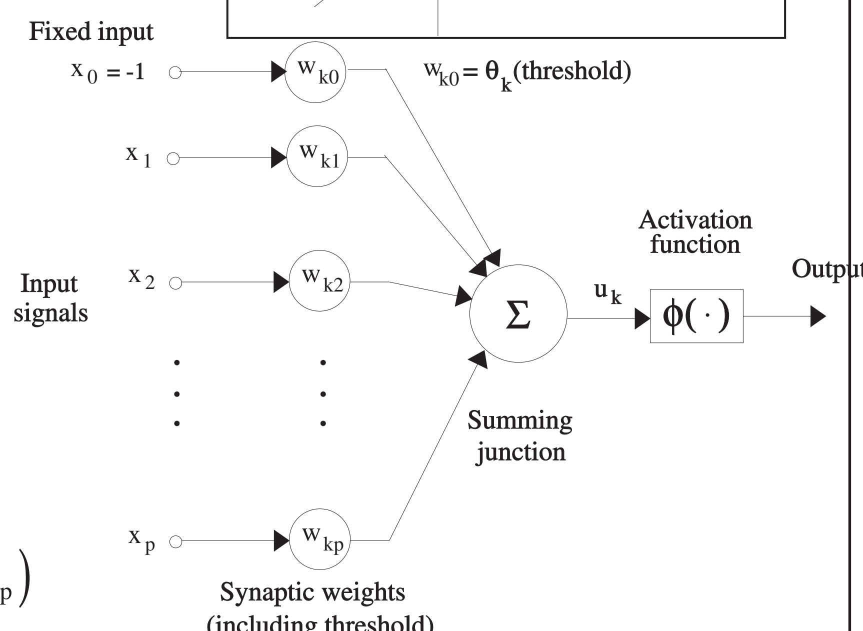
⌘ For conciseness, the bias can be incorporated as a weight $w_{k0}=\theta_k$ via a fixed input $x_0=-1$. Thus, the model becomes:

$$u_k = \sum_{j=0}^{p} w_{kj} x_j$$

$$y_k = \phi(u_k)$$

where $\quad \mathbf{x} = \left(-1, x_1, \ldots, x_p\right)$

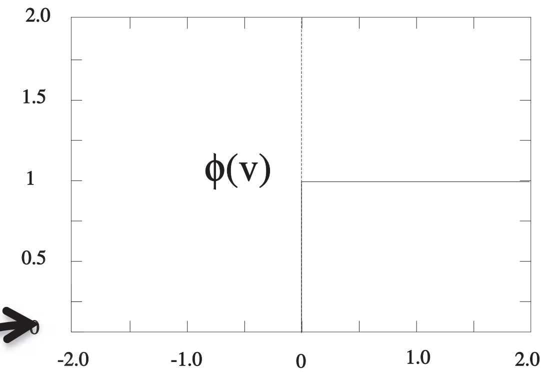
and $\quad \mathbf{w}_k = \left(\theta_k, w_{k1}, \ldots, w_{kp}\right)$



Types of activation functions

- The activation function $\phi(v)$, where v is the induced local field, defines the actual neuron output. There are various ways this can be implemented:
- **Threshold or Heaviside function.** This has a binary output:

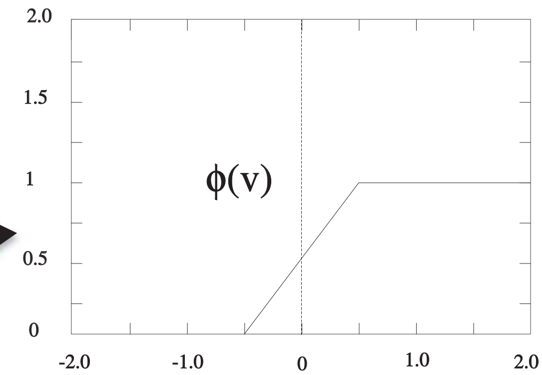
$$\phi(v) = \begin{cases} 1 & \text{if} \quad v \ge 0 \\ 0 & \text{if} \quad v < 0 \end{cases}$$



Threshold function

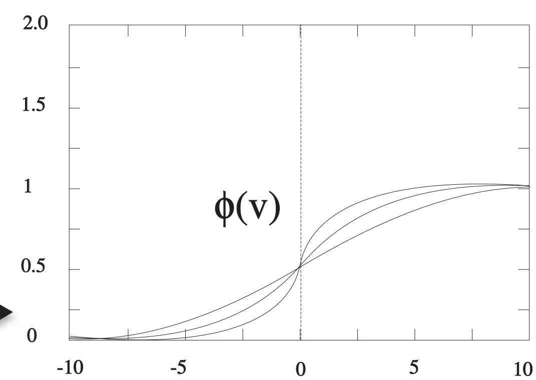
- **Piecewise linear function.** This corresponds to a nonlinear (linear truncated) amplification.

$$\phi(v) = \begin{cases} 1 & \text{if} \quad v \ge +\tfrac{1}{2} \\ v + \tfrac{1}{2} & \text{if} \quad -\tfrac{1}{2} < v < +\tfrac{1}{2} \\ 0 & \text{if} \quad v \le -\tfrac{1}{2} \end{cases}$$



Piecewise-linear function

- **Sigmoid function.** This produces a differentiable S-shaped nonlinear amplification of the field. Possible expressions are shown below. α is the slope of the Logistic sigmoid (when α➔∞ it becomes a Heaviside function). This function is continuous and ranges within (0,1).



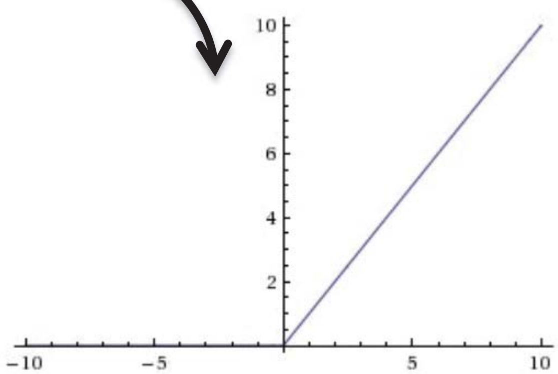
Sigmoid function

$$\phi(v) = \frac{1}{1+\exp(-\alpha \cdot v)} \quad \text{or} \quad \phi(v) = \tanh(v) = \frac{\exp(2v)-1}{\exp(2v)+1} \in (-1,+1)$$

⌘ **Rectified linear unit (RELU).** Most common choice in state-of-the-art neural networks & deep learning:

$$\phi(\mathrm{v}) = \max(\mathrm{v}, 0) = \begin{cases} \mathrm{v} & \mathrm{v} \geq 0 \\ 0 & \mathrm{v} < 0 \end{cases}$$

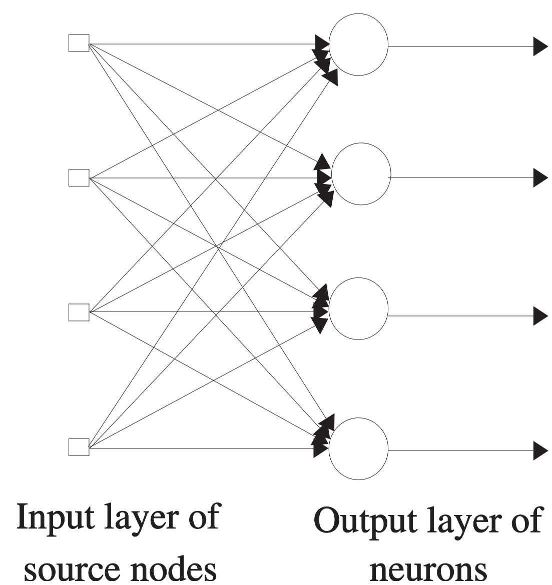
- Many more have been used!



Network architectures

⌘ There is a very strong coupling between NN **topologies** and their **learning algorithms**. Learning is the result of complex mathematical procedures, which require different specialised types of network node arrangements and inter-connections. Various topology types exist:

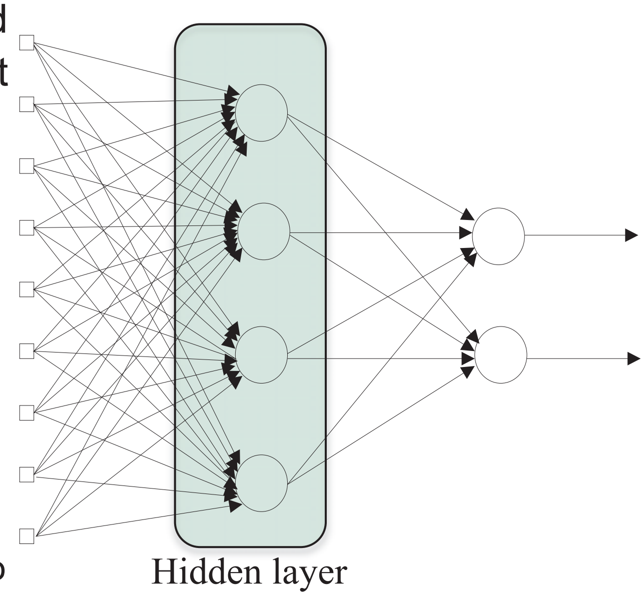
⌘ **Single-layer feedforward nets:** One input layer of source (input/non computing) nodes that project directly onto the output computation nodes. No cycles are allowed.



⌘ **Multi-layer feedforward nets:** As before, but with additional hidden layers, which they add more power and are capable to perceive, global, higher-order statistics and complex information in the input data.

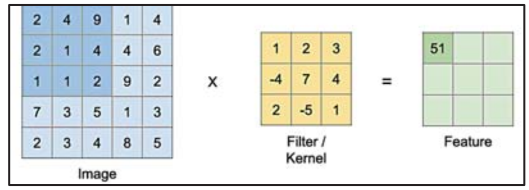
◎Typically the neurons of each layer have as input the output signals from the preceding layer(s).

◎Networks are typically fully connected in a layer-by-layer fashion (each node is connected to all nodes in the immediately previous layer only), but remote connections are also supported.

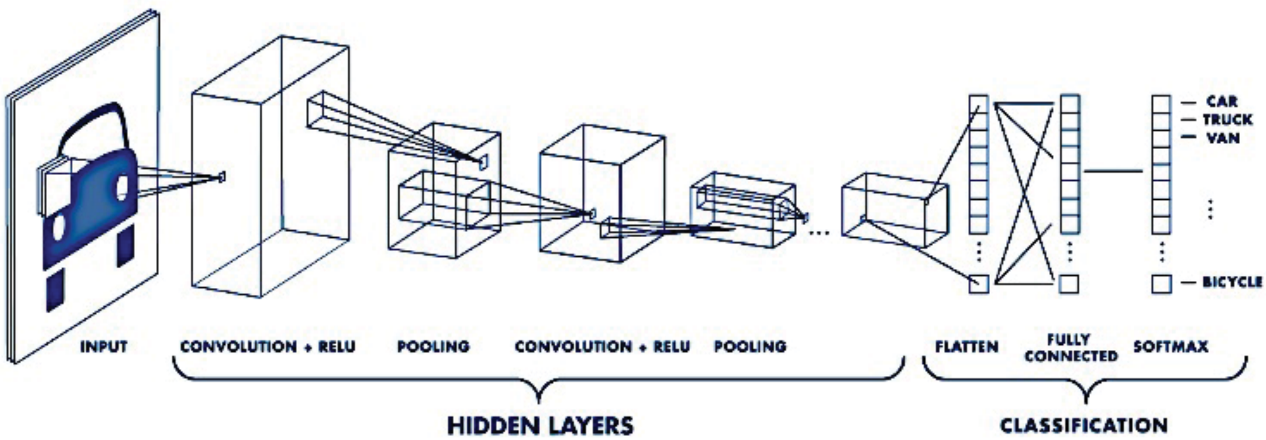


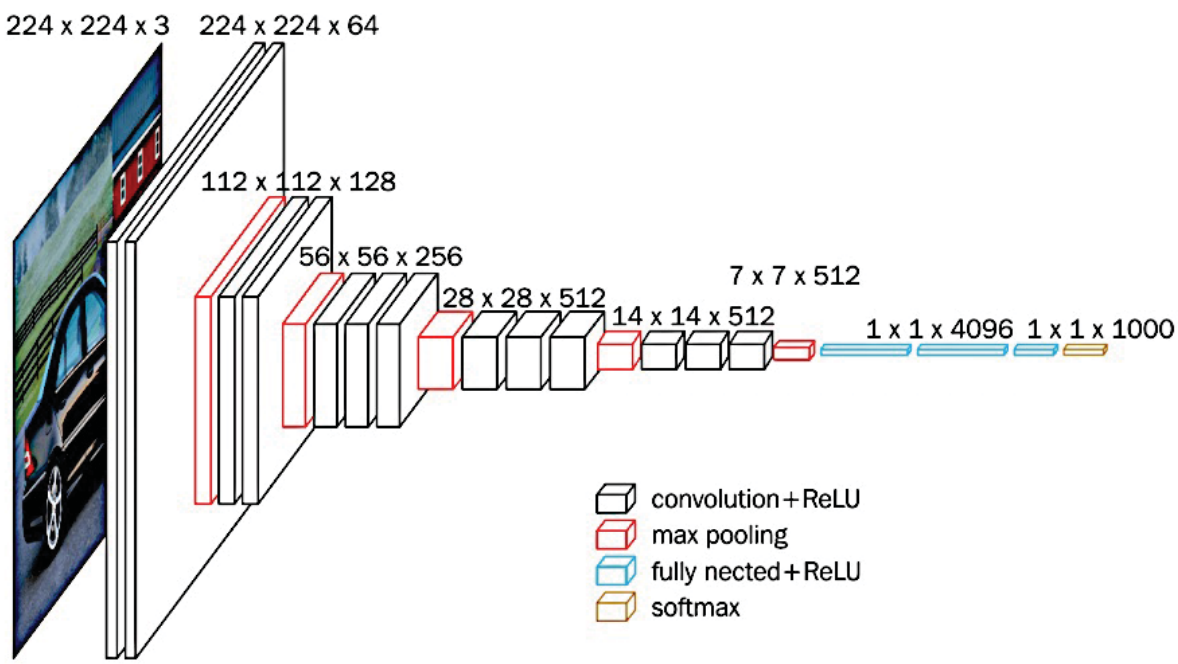
⌘ **Convolutional neural networks**: A class of multi-layer feedforward net with a special type of local connections.

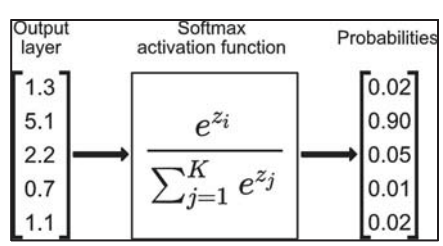
- ◎ Rely on convolutions for feature extraction.



- ◎ Widely-used for image classification

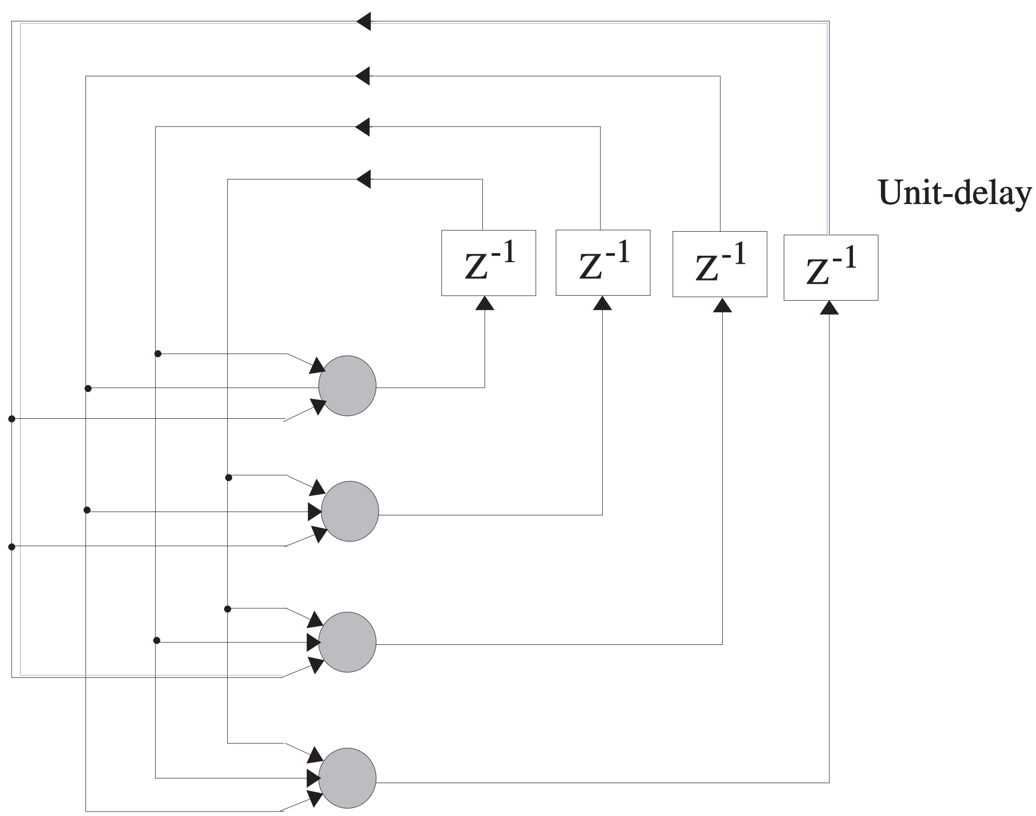






⌘ **Recurrent nets:** These are different from the aforementioned types, in that they have feedback loops, which form cycles in the graph representation.

- ◎ The simple example below has no hidden layers, but a single layer of neurons, with the output of each neuron fed as input to all other neurons (without any *self-feedback loops)*.
- ◎ Feedback loops involve connections with unit-delay (z^{-1}) elements, which together with nonlinear activations enable the nonlinear dynamic behaviour of the network.
- ◎ The existence of feedback loops have profound impact on the learning capability of the net.



Mathematical expressions

⌘ Possible examples of mathematical expressions of these topology types are:

◎ Single-layer feedforward nets:

$$y_k(\mathbf{x}) = \phi\left(\sum_{j=0}^{p} w_{kj} x_j\right)$$

◎ Multi-layer feedforward nets:

$$y_k(\mathbf{x}) = \phi\left(\sum_{j=0}^{p_{hidden}} w_{kj} \underbrace{\phi\left(\sum_{i=0}^{p_{input}} w_{ji} \overbrace{x_i}^{\text{input layer}}\right)}_{\text{hidden layer output}}\right)$$

◎ Recurrent (no hidden, delay one, self-feedback):

$$y_k(n) \equiv y_k(\mathbf{x}(n), n) = \phi\left(\underbrace{\sum_{i=0}^{p_{input}} w_{ki} x_i(n)}_{\text{current input}} + \underbrace{\sum_{j=0}^{p_{output}} w_{kj} y_j(n-1)}_{\text{past output}}\right)$$

Knowledge representation

⌘ Knowledge representation is naturally goal-directed.

⌘ "Intelligent machines" find "*good*" solutions when they represent knowledge "*well*". Exactly the same holds for ANNs, which are a type of A.I. machines.

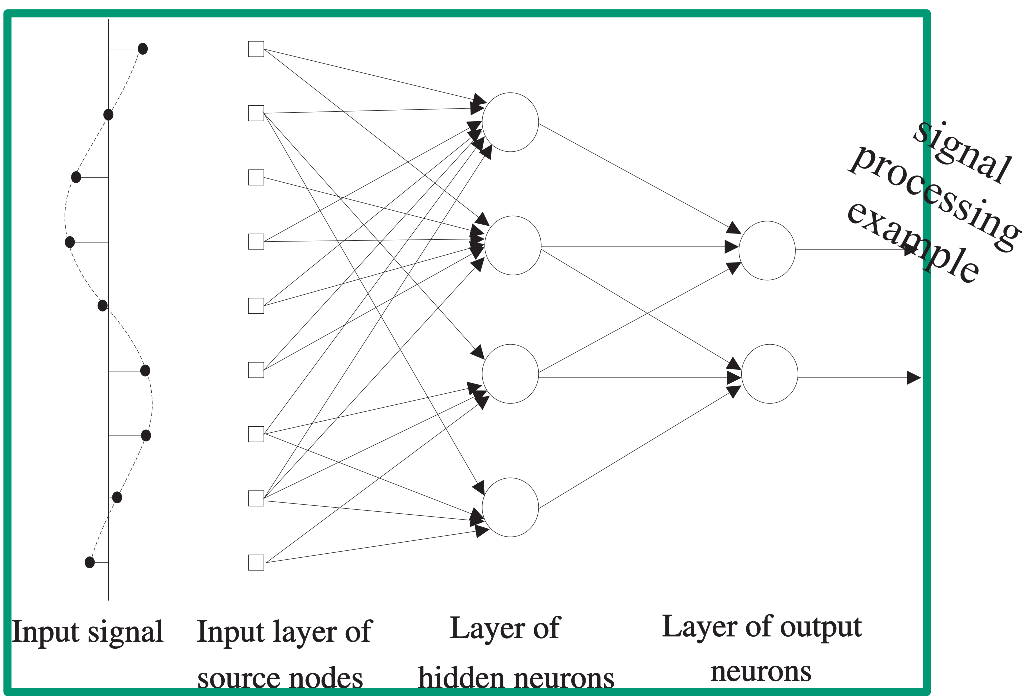
⌘ The primary task of a NN is to learn its environment. To accomplish this, we need to feed it with environment knowledge, which is based on:

- ⊚ **Observations (or measurements or samples)**. Typically, such observations are noisy, contain errors and redundant or missing information, and cannot always sample the environment adequately.
- ⊚ They can be **labelled** or **unlabelled**. Each labelled input sample **x** is paired with an **actual or target output** response **y** (measured ground truth). Unlabelled samples just represent environmental distributions and properties.

⌘ Continued…

- ◎ A set { ($\mathbf{x}_i$,y_i) }, i=1,2,… of experimentally acquired samples is referred as a **training dataset**.

- ◎ Examples:
  - Medical diagnosis: **x**=patient data, y=positive/negative of some pathology
  - OCR: **x**=pixel values and writing curves, y='A', 'B', 'C', …
  - Maths, regression (that is recover unknown function f): **x**=**x**, y=f(**x**)
  - Defence: **x**=sensor inputs, **y**=target, position, hostile, action
  - Image analysis: **x**=image pixel features, **y**=scene/objects contained in image
  - Weather: **x**=current & previous conditions per location, y=tomorrow's weather
  - Finance: **x**=today's prices or stock market, **y**=tomorrow's prices or values, etc.
  - Vehicle guidance: **x**=vehicle sensors and current status, **y**=direction & speed
  - Electronics: **x**=board specs and components, **y**=optimal circuit routing
  - Games/Entertainment: **x**=current character position, **y**=enemy attack pattern
  - **… … … this list can never end, since applications of NNs are almost all the applications of A.I., which is vast and currently extremely active!!!**



⌘ So how do the above help us to solve a real-world problem with a NN?

- ◎ Firstly, choose an **appropriate architecture**, with as many input nodes as the length of the input feature vector **x**, and as many output neurons needed for the output **y**.

- ◎ Then, choose a subset of the available observations as a **training dataset** to **train (or teach)** the NN, so that it learns the problem at hand.

- ◎ Lastly, you use the remaining observations or new observations never seen before, as a **testing dataset** in order to test the recognition performance (i.e., <u>accuracy & generalisation</u>) of your NN model (**model assessment**).

- ◎ The final NN model not only constitutes an implicit model for the environment (problem), but **also** performs the information processing function of interest!

<END of Chapter 2>