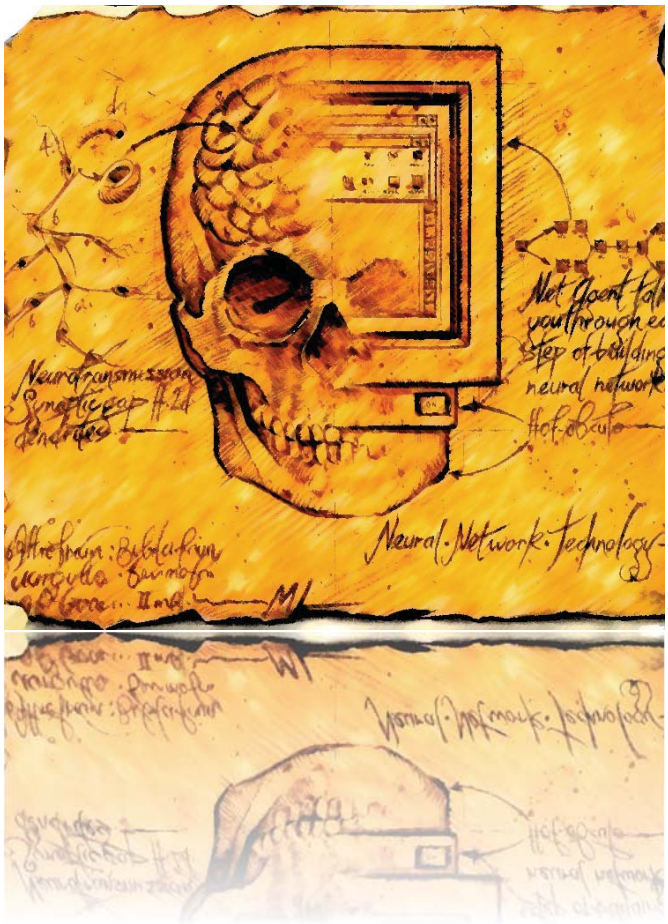
Computational Intelligence I: Neural Networks

Chapter 3: Learning Processes



Learning… in general

⌘ The primary characteristics of a NN is that it can learn from its environment, via adapting its structure in order to improve its performance.

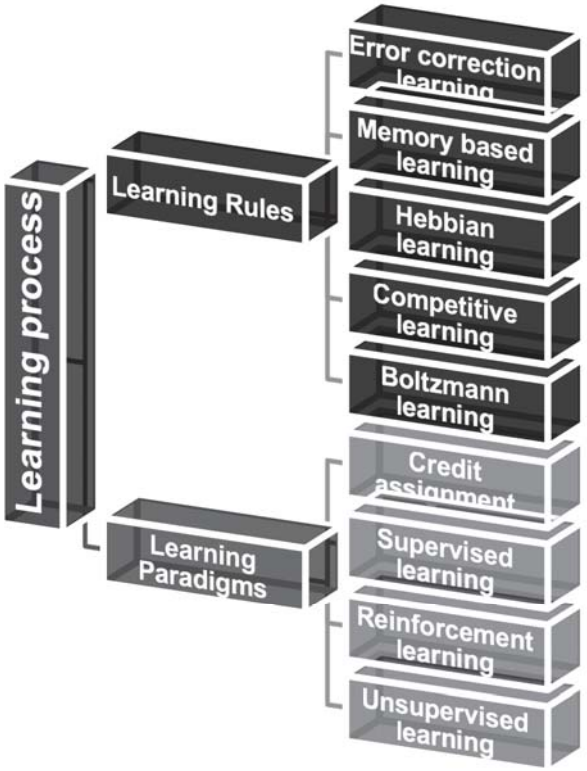
◎ *A possible definition:* ***Learning*** *is a process by which the free* ***parameters*** *of a NN are adapted through a process of stimulation by the* ***environment*** *in which the network is embedded. The type of learning is determined by the manner in which the parameter adaptations take place.*

⌘ This implies the following sequence of actions:

◎ The network is stimulated by the environment.

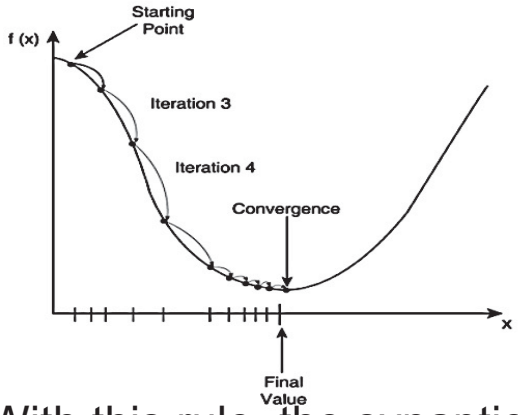
◎ Its free parameters undergo changes to reflect this stimulation (**training mode**).

◎ The network responds in a new way to the environment, as a result of the changes in its internal structure (**online mode**).



Error-correction Learning

- Assume that the k^{th} neuron receives a signal vector **x**(n), where n is the time variable of a discrete synaptic adaptation procedure. This signal can be the result of other neurons in previous layers.
- The neuron **output** signal is denoted by $y_k(n)$.
- The **desired** output signal is denoted by $d_k(n)$.
- Therefore, we can define the **error** signal: $e_k(n) = d_k(n) - y_k(n)$
- This constitutes a control mechanism by which a sequence of corrective step-by-step adjustments are driven to bring the output y_k closer to the desired d_k.
- This is achieved by minimising a cost function E(n) until the system reaches a steady-state (synaptic weights are stabilised): $E(n) = \frac{1}{2} e_k^2(n)$
- Employing a positive learning parameter η, this corresponds to the **Delta or Widrow-Hoff rule**:



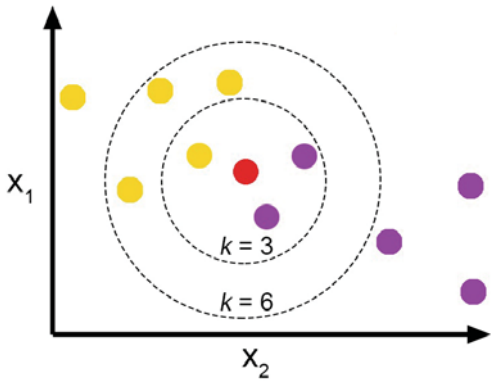
$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

$$\underbrace{w_{kj}(n+1)}_{\text{new weight}} = \underbrace{w_{kj}(n)}_{\text{old weight}} + \Delta w_{kj}(n)$$

- With this rule, the synaptic weight adjustment is **proportional** to the error signal and the input signal of the synapse under adjustment.

Memory-based Learning

- Under this model all past experiences are explicitly stored in **memory** as a set D_{train} of N pairs with samples $\mathbf{x}_i$ associated with known desired responses d_i: $D_{train} = \{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$
- When we need to estimate the response of an unseen vector $\mathbf{x}_{test}$ (i.e., respond to the environment), the algorithm **retrieves** and **analyses** some training data (a subset of D_{train}) with patterns "close" to $\mathbf{x}_{test}$. This requires:
  - A criterion for defining a local neighbourhood around $\mathbf{x}_{test}$.
  - The learning rule to be applied to the training examples in that local neighbourhood.
- A simple example of this rule is the **nearest neighbour rule** (1NN), where the sample $\mathbf{x}^*$ in D_{train} closest (in the Euclidean norm sense, perhaps) to $\mathbf{x}_{test}$, is calculated as:



$$\mathbf{x}^* = \underset{x \in D_{train}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{x}_{test}\|_2$$

- Then, the pattern d* (the stored response of $\mathbf{x}^*$) becomes associated with $\mathbf{x}_{test}$ (**predicted** output).
- Extensions of the above include kNN (k-nearest neighbour classifier), weighted nearest neighbour, or Parzen kernels.

Hebbian Learning

- This type of learning is based on the model of Hebbian synapse:
  - If the **neurons on either side of a synapse** are activated simultaneously, then the strength of that synapse should be selectively increased.
  - If the above activation is asynchronous, then that synapse is weakened.
- A Hebbian synapse is defined as one that uses a time-dependent, highly local and strongly interactive mechanism to to increase synaptic efficiency.
- Assume the k^{th} neuron has pre- and post-synaptic signals denoted by x_j and y_k, respectively. The general Hebbian adjustment applied to its synaptic weight w_{kj} at time n is:

$$\Delta w_{kj}(n) = F\left[y_k(n), x_j(n)\right]$$

  - **Hebb's hypothesis**:
  - The simplest form for $F(\cdot,\cdot)$ yields the **activity product rule** which based on a learning rate η acts with a change of:

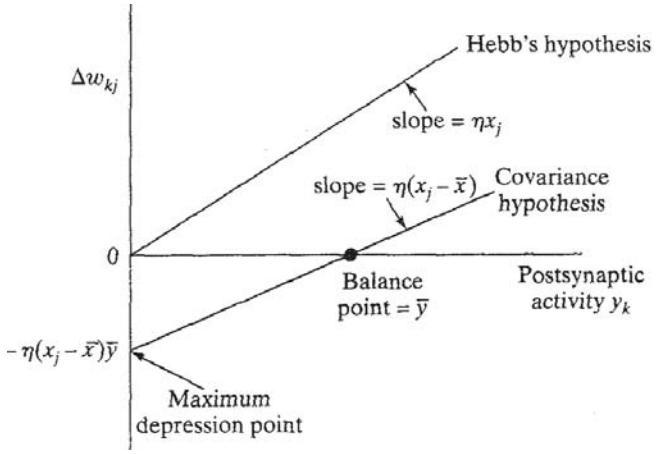
$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$$

  - However, repeated application of the input signal x_j, leads to an increase in y_k, which leads to ***synaptic saturation*** due to the exponential growth.

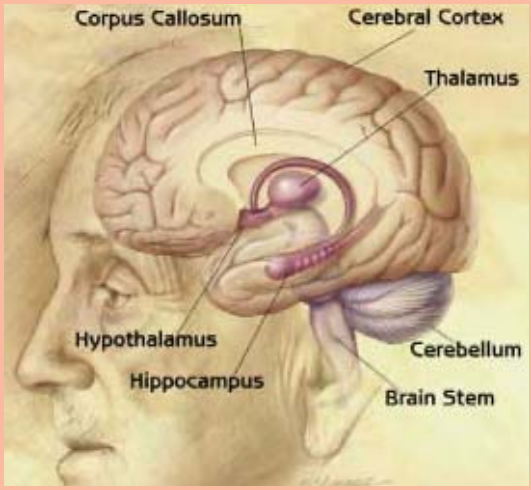
- **Covariance hypothesis**:
- To overcome the previous limitation, the pre- and post-synaptic signals are replaced by their departure from their average values over a time interval, giving the adaptation:

$$\Delta w_{kj}(n) = \eta(y_k - \bar{y})(x_j - \bar{x})$$

- The average values act like pre-synaptic and post-synaptic thresholds.
- They achieve both convergence and synaptic strengthening or weakening.



- There is strong physiological evidence for Hebbian learning in the area of the human brain called hippocampus, which plays an important role in learning, long-term memory and navigation.
  - Neurons that fire together wire together!



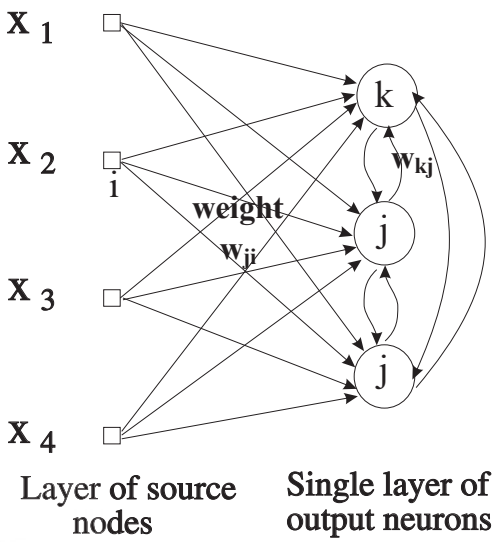
Competitive Learning

- In this model the output neurons **compete** amongst themselves to become active (to fire), as opposed to other learning rules where more than one neurons can be active.
- Since **only one output neuron** can be active, it is more suitable for classification problem based on salient features.
  - For the k^{th} neuron to be the winning one (y_k=1), its induced local field v_k (i.e., the combined signals of all feedforward (excitatory) + lateral (inhibitory) inputs to the neuron) should be the largest compared to that of all other neurons (y_j<1):

$$y_k = \begin{cases} 1 & \overbrace{\text{if } v_k > v_j \ \forall j \neq k}^{\text{activation } \phi} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{where } v_k = \sum_{i=1}^{p_{in}} w_{ki} x_i + \sum_{\substack{j=1 \\ j \neq k}}^{p_{out}} w_{kj} y_j$$

$$\sum_{i=1}^{p_{in}} w_{ki} = 1, \forall k$$



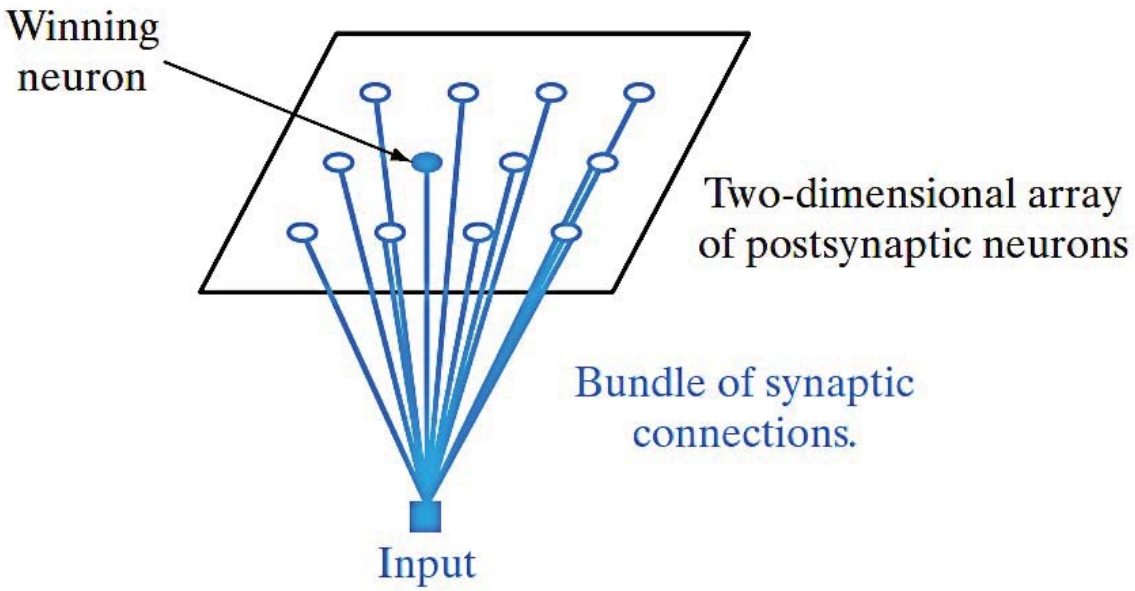
- Each k^{th} node is allotted a fixed 'amount' of synaptic weight as, so that learning is facilitated by shifting weights from its inactive to its active nodes.

⌘ The standard **competitive learning rule** adapts the weight as:

$$\Delta w_{ki} = \begin{cases} \eta(x_i - w_{ki}) & \text{if neuron k wins the competition} \\ 0 & \text{otherwise} \end{cases}$$

◎ This has the effect of moving the synaptic vector $\mathbf{w}_k$ of the winning neuron k closer to the input pattern **x**.

◎ That is why such NNs apply their competitive learning characteristics to **Clustering** of input patterns (i.e., samples are split into different groups which share certain statistical and geometric characteristics).



Kohonen model

Boltzmann Learning

⌘ *Boltzmann machines* are NNs with stochastic learning algorithms inspired by **statistical mechanics**.

◎ Neurons have only binary states: on (+1) or off (-1).

◎ An energy function is defined to depend on the state x_j of each individual j^{th} neuron and synaptic weight w_{kj} (but not self-feedbacks):

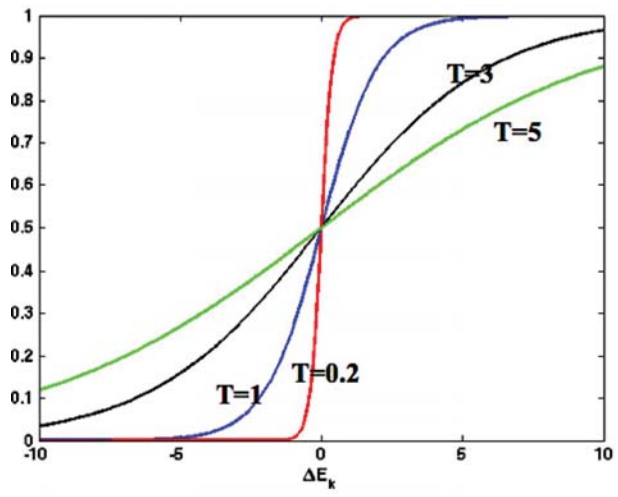
$$E = -\frac{1}{2}\sum_{j}\sum_{k\neq j} w_{kj}x_k x_j$$

⌘ The machine learns as follows:

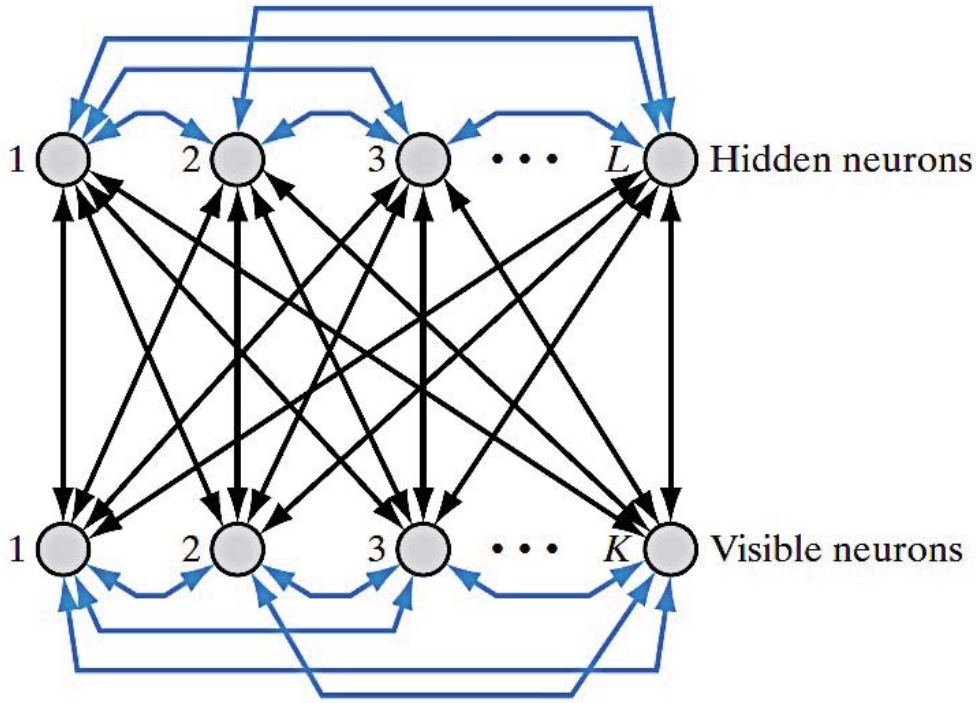
◎ An arbitrary neuron k is chosen at temperature T, and its state x_k is flipped to $-x_k$ with probability:

$$p\left(x_k \rightarrow -x_k\right) = \frac{1}{1+\exp\left(-\frac{\Delta E_k}{T}\right)}$$

where ΔE_k is the **energy change** resulting from this state flip at system temperature T (*or switch into whichever of its states makes the total energy lower*).



◎ Learning is accomplished by reaching a thermal equilibrium by repeatedly applying this rule.



- The neurons of a Boltzmann machine can be:
  - **Visible**: interface between environment and machine.
  - **Hidden**: act as freely operating ones.
- Neurons can operate in two modes:
  - **Clamped**: visible neurons are all clamped onto specific states.
  - **Free-running**: all visible and hidden neurons are operating freely.
- **Synaptic adaptation**:
  - Let $\rho_{kj}^{+/-}$ denote the correlation (within [-1,+1]) between the states of the k^{th} and j^{th} neurons with the network in its **clamped / free-running** conditions. These are averaged over all possible states of the machine in its thermal equilibrium.
  - Then, the learning is achieved via:

$$\Delta w_{kj} = \eta\left(\rho_{kj}^{+} - \rho_{kj}^{-}\right), \quad \forall j \neq k$$

Credit Assignment Problem

- The Credit Assignment refers to the concept of assigning **credit or blame** for overall outcomes to each of the internal decisions made by a learning algorithm, which contributed to those outcomes.

- Two main subproblems exist:

  - **Temporal**: assignment of credit to actions involving the **instants of time** when the actions that deserve credit were actually taken. More related to cases where many actions were taken and some result to certain outcomes.

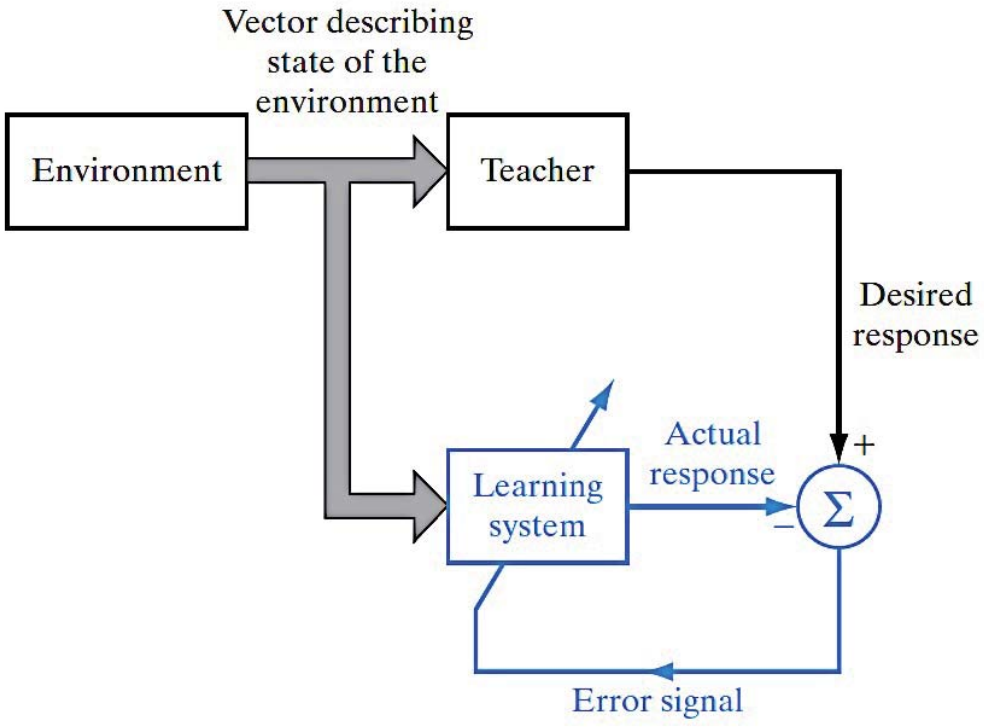
  - **Structural**: assignment of credit to **which internal structures** of actions generated by the system and by how much. More related to multi-component machine learning.

- Examples include the error-correction learning applied to multilayer feedforward NNs (see chapter 5). The overall performance in the learning task at hand depends on the operation of each hidden and output neuron.

Supervised Learning

⌘ Supervised learning (or *Learning with a Teacher*):

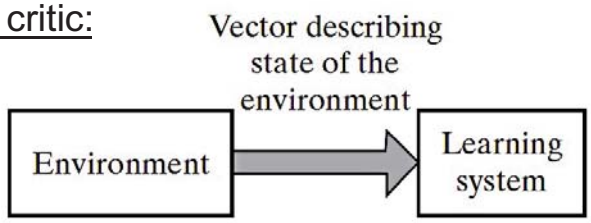
- The **teacher** has **environmental knowledge**; this is usually in the form of input-output examples: $\mathrm{D}_{\text{train}} = \left\{\left(\mathbf{x}_i, \mathrm{d}_i\right)\right\}_{i=1}^{N}$
- Despite having measurements from it, the environment is unknown.
- Through the teacher's knowledge of the ideal desired responses d_i, the network is guided towards performing the optimum action.
- Finally, the network parameters are adjusted based on the combined effect of the error signal and input vector, via an iterative procedure with the ultimate aim to emulate the teacher.
- When this is achieved accurately, the teacher is removed from the system and the NN is left to its online operational mode.



Unsupervised Learning

⌘ A form of *Learning without a Teacher*; also referred to as *Self-organised Learning*.

◎ It is **not** facilitated by a teacher or a critic:

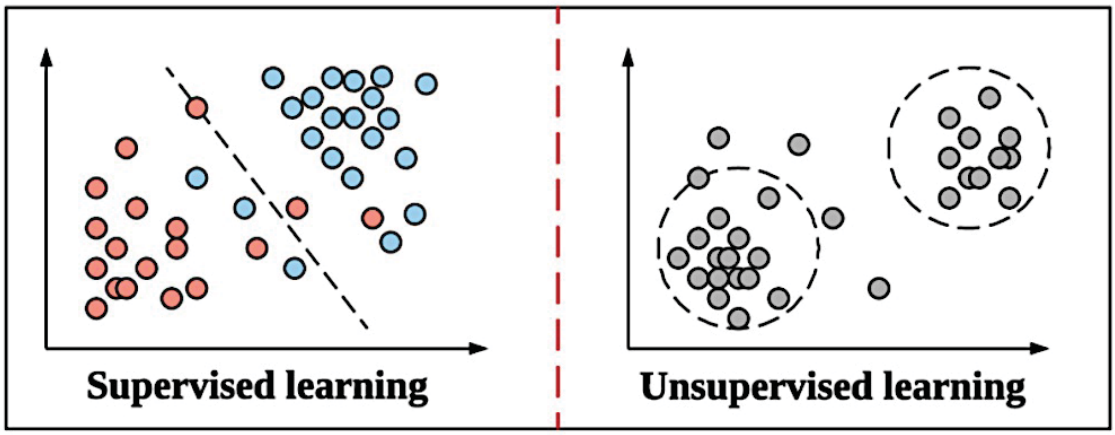


◎ A **learning task-independent** measure of quality is used to adjust the system's free parameters.

◎ The system forms **internal representations** to encode the features and the geometrical and statistical properties of the datasets.

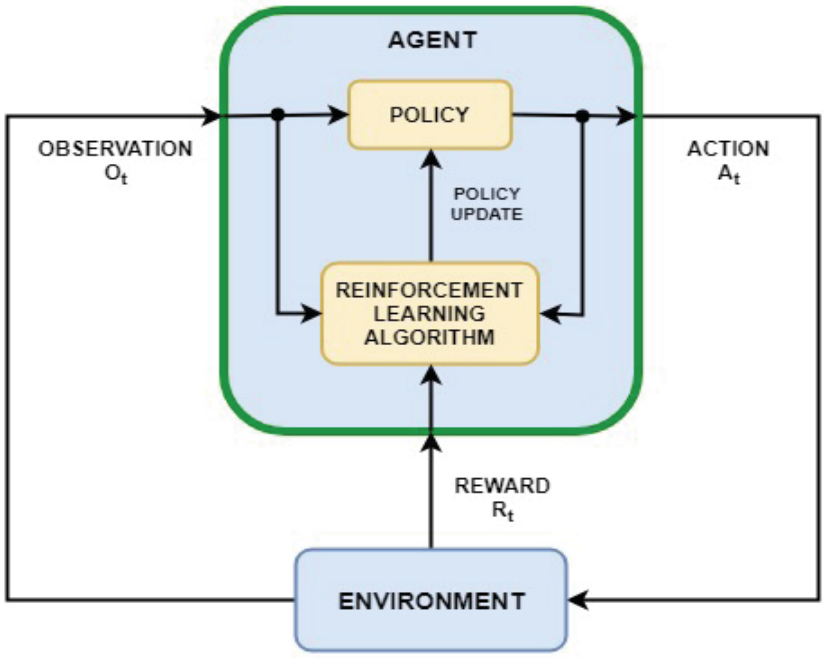
◎ An example is the use of competitive learning NNs:

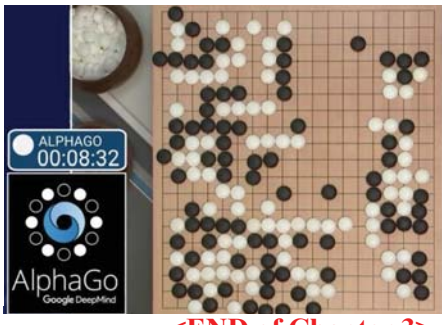
- An input layer receives environmental measurements **x**.
- A competitive layer has nodes that compete with each other for the privilege of responding to "certain features" found in the input data.
- During online mode, a new input $\mathbf{x}_{test}$ activates only specific nodes from the output layer.



Reinforcement Learning

- A type of *Learning without a Teacher*; also referred to as *Neurodynamic Programming*.
  - The goal of reinforcement learning is to train an agent to complete a task within an uncertain environment.
  - The agent receives **observations** and a **reward** from the environment and takes actions.
  - The agent contains two components: a **policy** and a **learning algorithm**.
  - The **policy** is a mapping that selects actions based on the observations from the environment. Typically, the policy has tunable parameters, e.g., a neural network.
  - The **learning algorithm** continuously updates the policy parameters based on the actions, observations, and rewards. The goal of the learning algorithm is to find an **optimal policy** that maximizes the expected cumulative long-term reward received during the task.





<END of Chapter 3>