Introduction to Neural Networks

Neural Networks

- **McCulloch & Pitts (1943) are generally recognised as the designers of the first neural network**

- **Many of their ideas still used today (e.g. many simple units combine to give increased computational power and the idea of a threshold)**

Neural Networks

- **Hebb (1949) developed the first learning rule (on the premise that if two neurons were active at the same time the strength between them should be increased)**

General functions of neural networks

Classification

separation of input data in specified classes

e.g. Handwritten character recognition

Clustering

grouping together objects that are similar to each other

e.g. Data conceptualization

e.g. Solving the TSP

Prediction

given a sequence of data, predict the upcoming ones

e.g. Stock market forecasting

Associative memory

restores deformed input samples to its original content

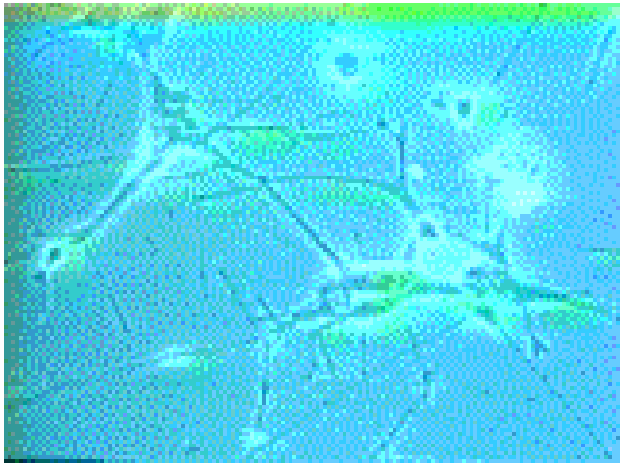
e.g. Filtering of noisy samples

e.g. Image compression

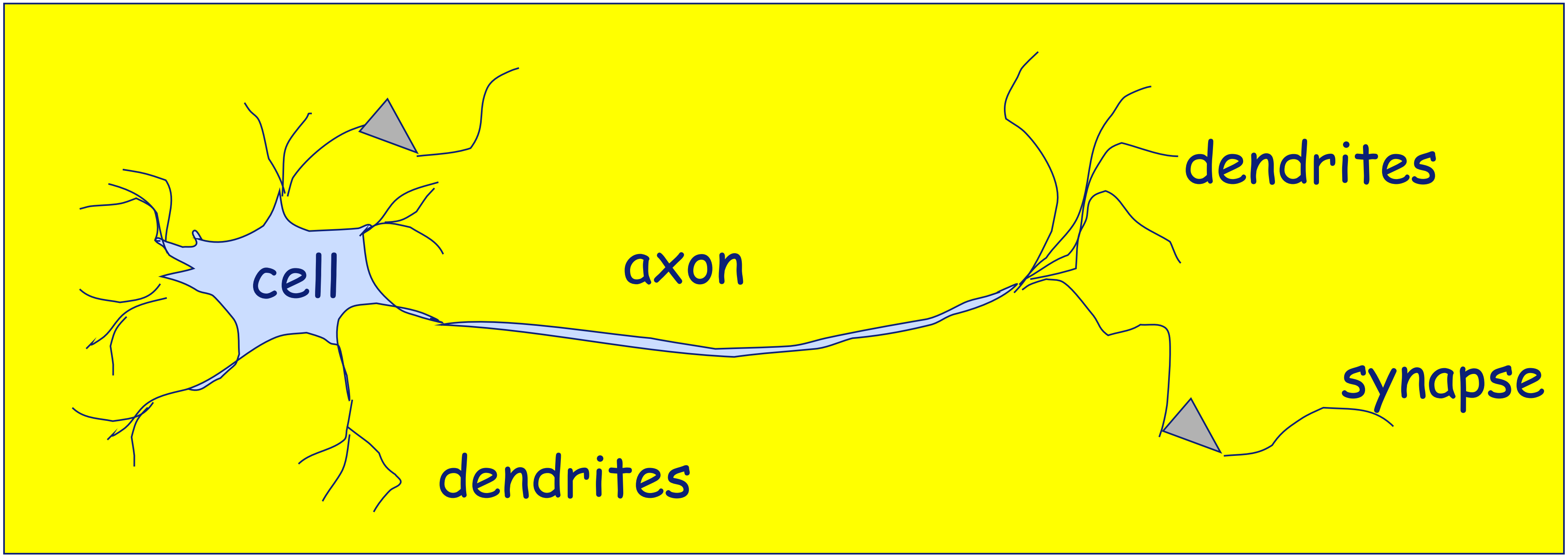
How Does the Brain Work ? (1)

NEURON

- The cell that perform information processing in the brain
- Fundamental functional unit of all nervous system tissue



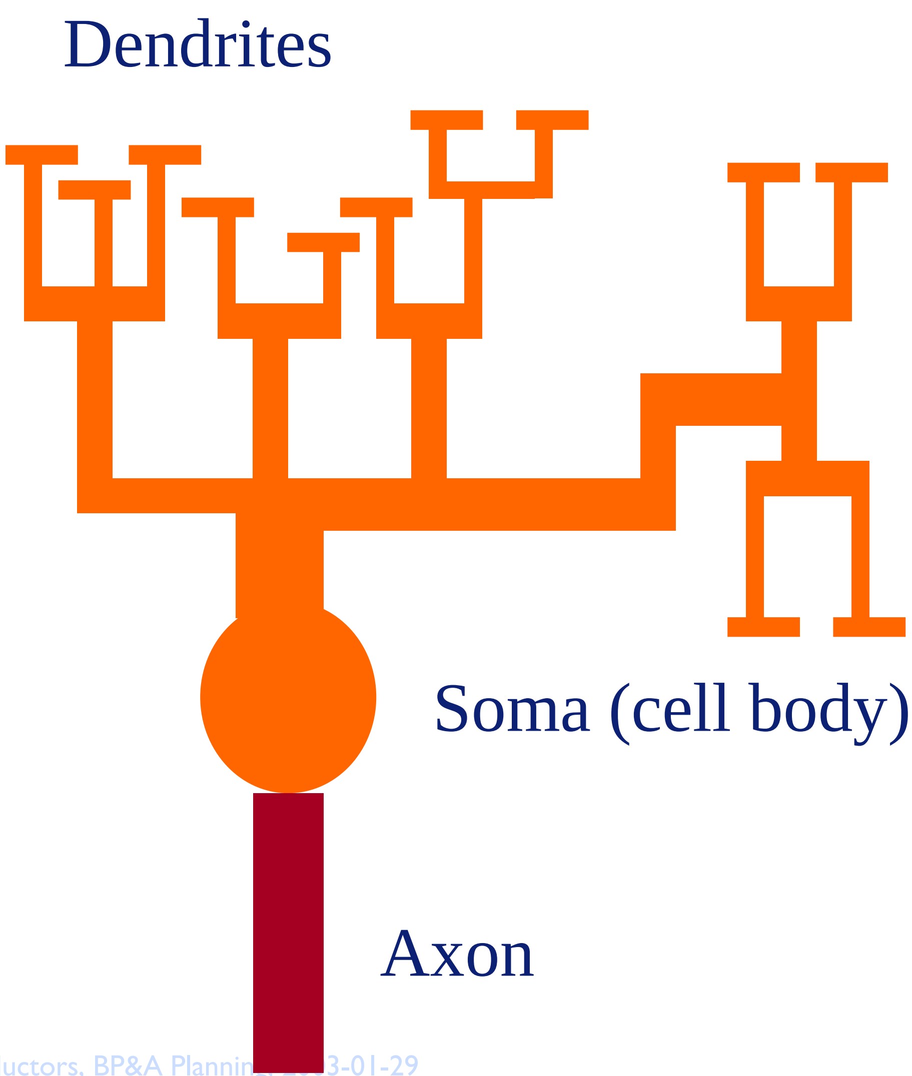
Biological neurons

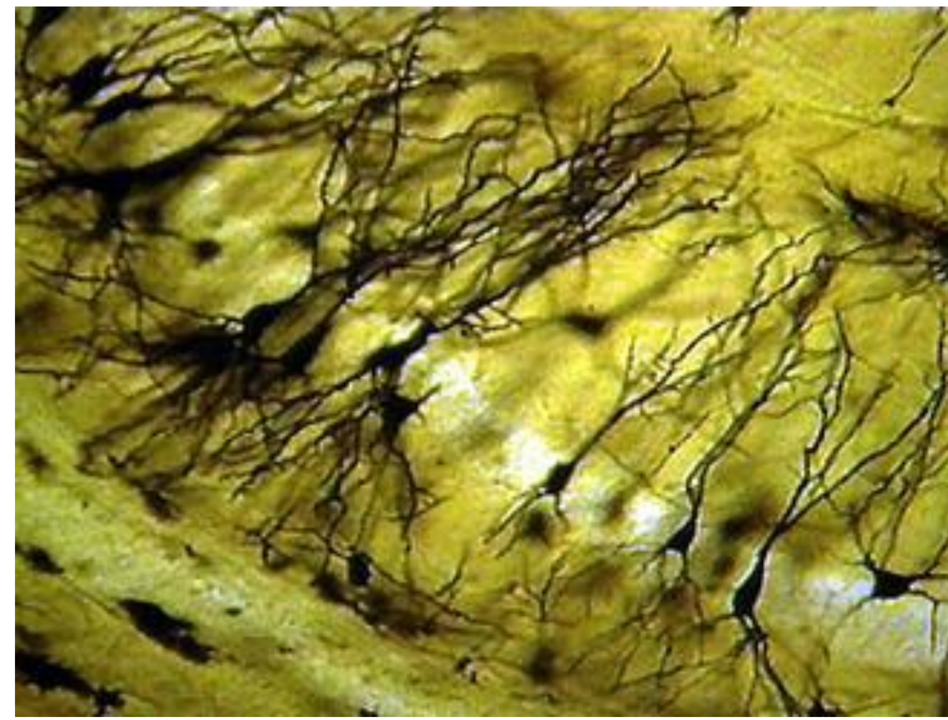


Neural Networks

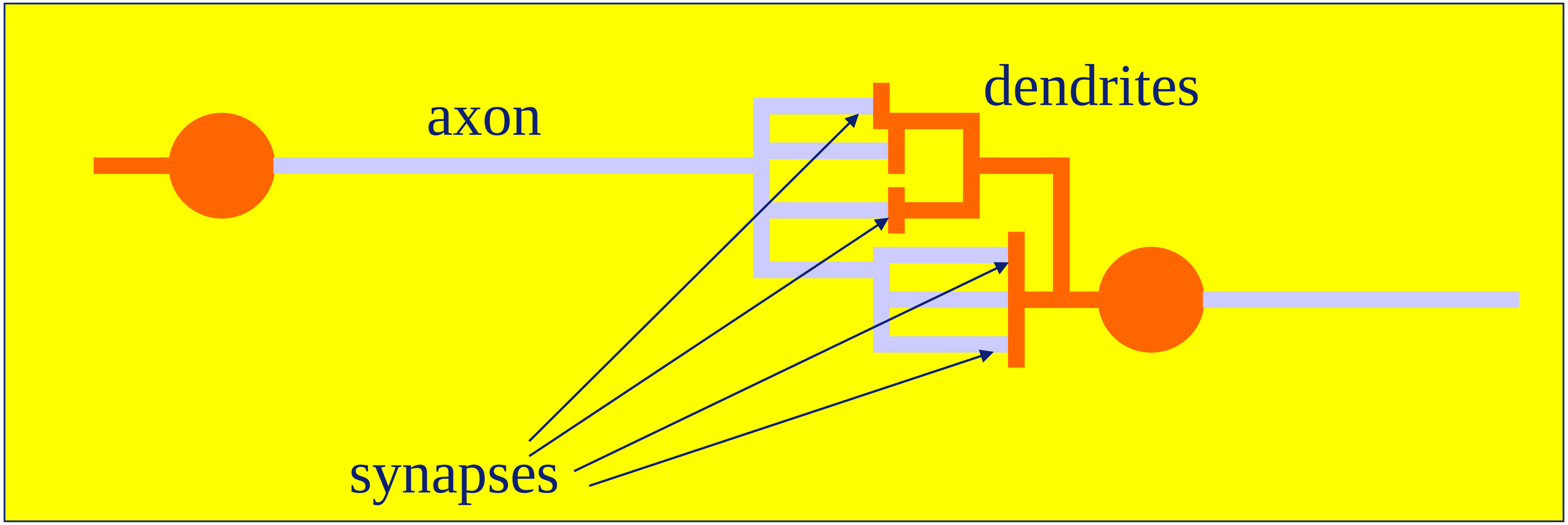
- **We are born with about 100 billion neurons**
- **A neuron may connect to as many as 100,000 other neurons**

Biological inspiration





Biological inspiration



The information transmission happens at the synapses.

Biological inspiration

The spikes travelling along the axon of the pre-synaptic neuron trigger the release of neurotransmitter substances at the synapse.

The neurotransmitters cause excitation or inhibition in the dendrite of the post-synaptic neuron.

The integration of the excitatory and inhibitory signals may produce spikes in the post-synaptic neuron.

The contribution of the signals depends on the strength of the synaptic connection.

Biological Neurons

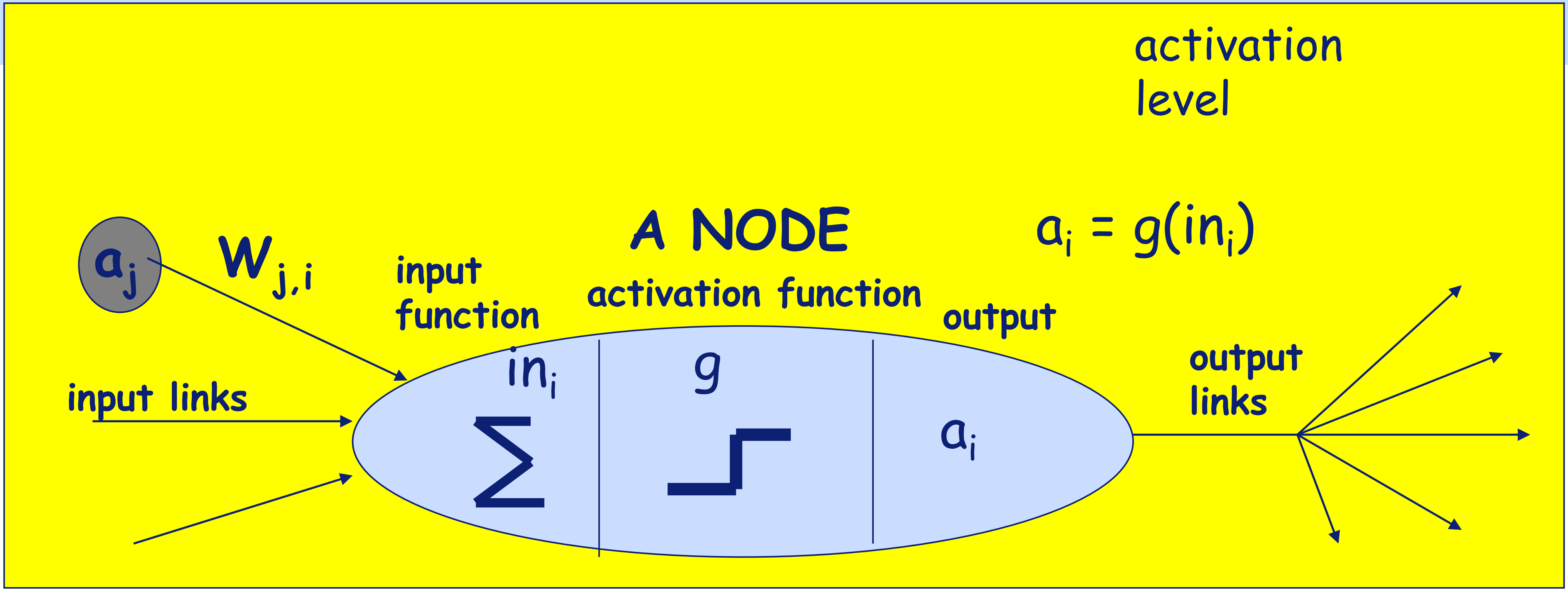
- human information processing system consists of brain neuron: basic building block
  - cell that communicates information to and from various parts of body
- Simplest model of a neuron: considered as a threshold unit –a processing element (PE)
- Collects inputs & produces output if the sum of the input exceeds an internal threshold value

Artificial Neural Nets (ANNs)

- Many neuron-like PEs units
  - Input & output units receive and broadcast signals to the environment, respectively
  - Internal units called hidden units since they are not in contact with external environment
  - units connected by weighted links (synapses)

- A parallel computation system because
  - Signals travel independently on weighted channels & units can update their state in parallel
  - However, most NNs can be simulated in serial computers

- A directed graph, with labeled edges by weights is typically used to describe the connections among units

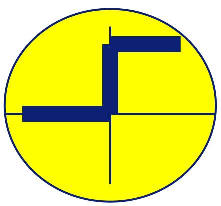


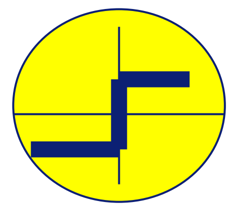
Each processing unit has a simple program that:

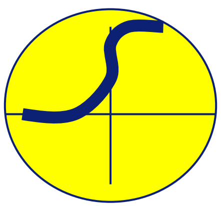
a) computes a weighted sum of the input data it receives from those units which feed into it

b) outputs of a single value, which in general is a non-linear function of the weighted sum of the its inputs ---this output then becomes an input to those units into which the original units feeds

g = Activation functions for units







Step function

(Linear Threshold Unit)

step(x) = 1, if x >= threshold
0, if x < threshold

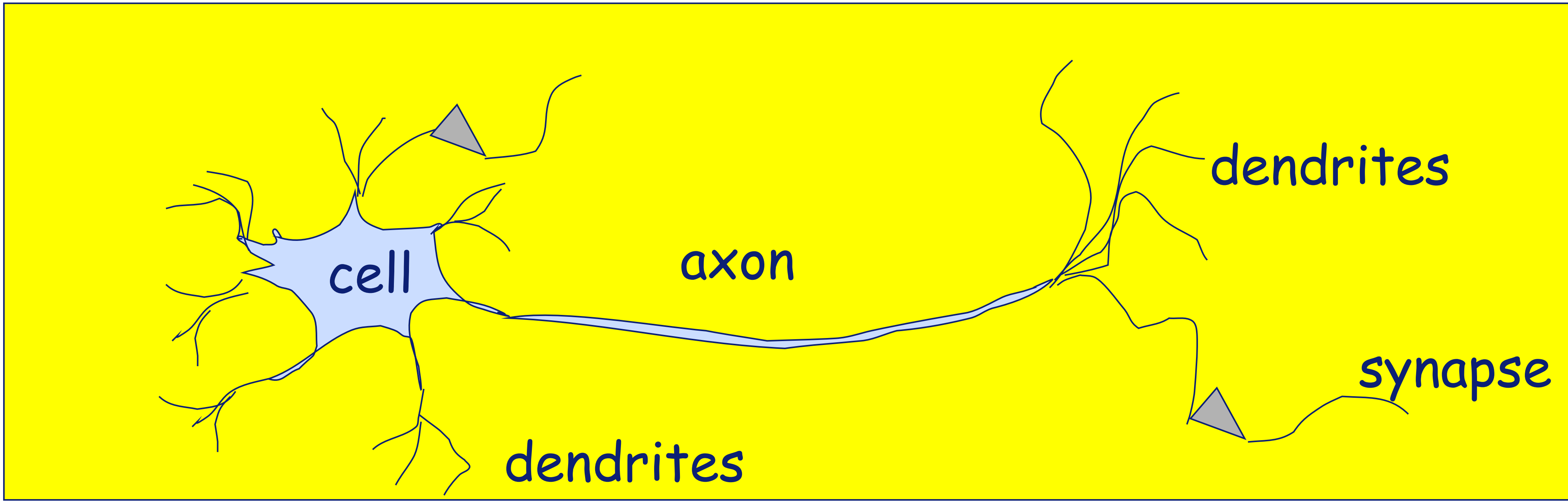
Sign function

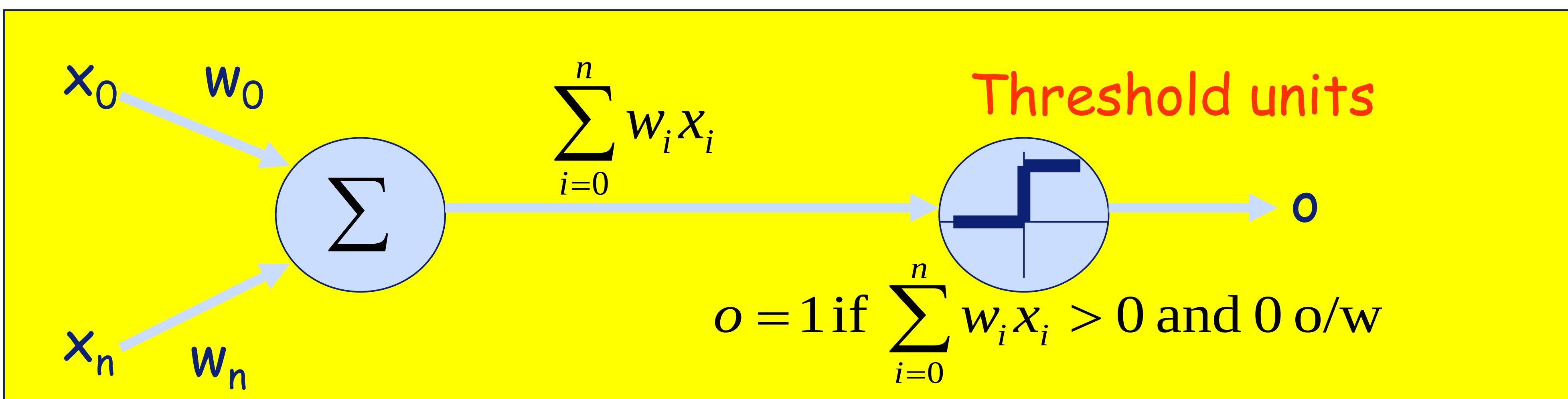
sign(x) = +1, if x >= 0
-1, if x < 0

Sigmoid function

sigmoid(x) = $1/(1+e^{-x})$

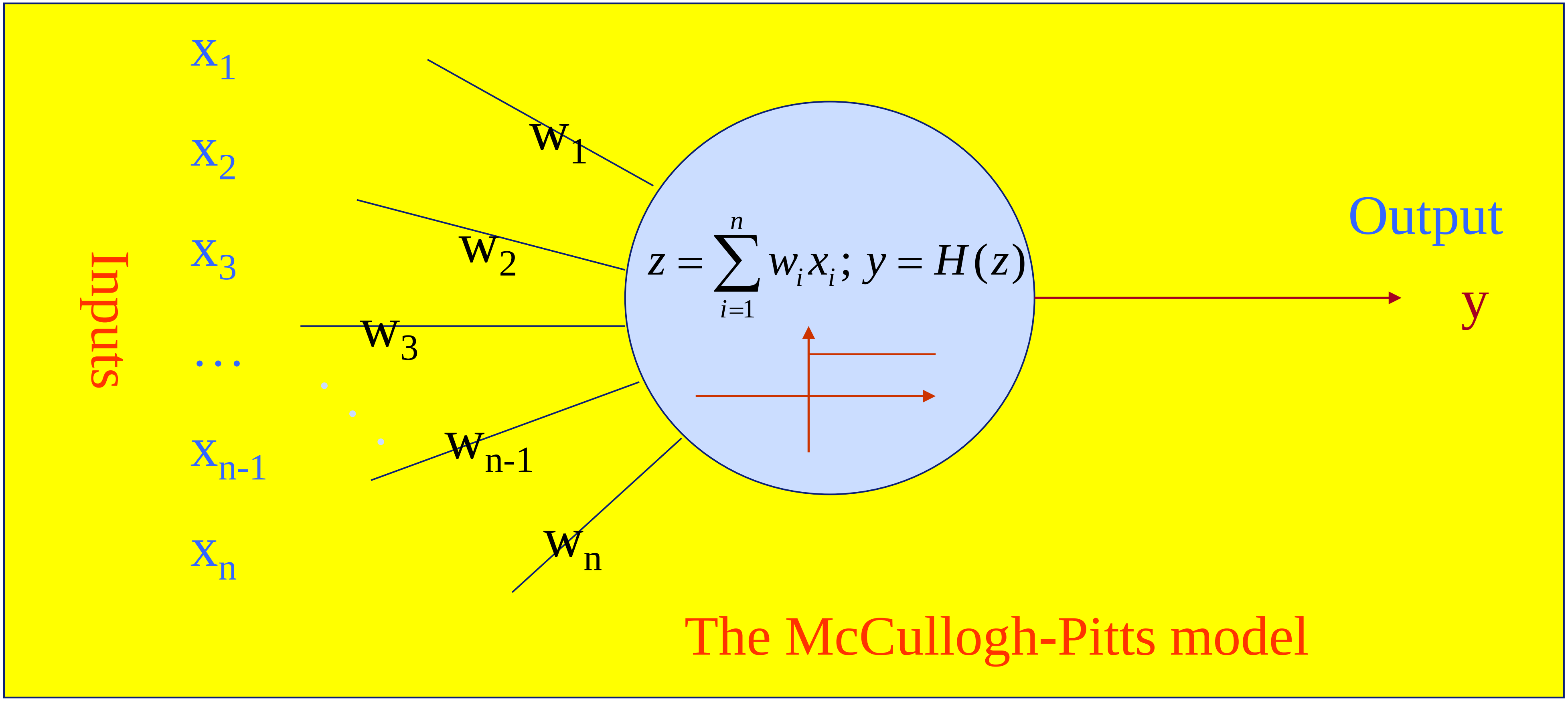
Real vs artificial neurons





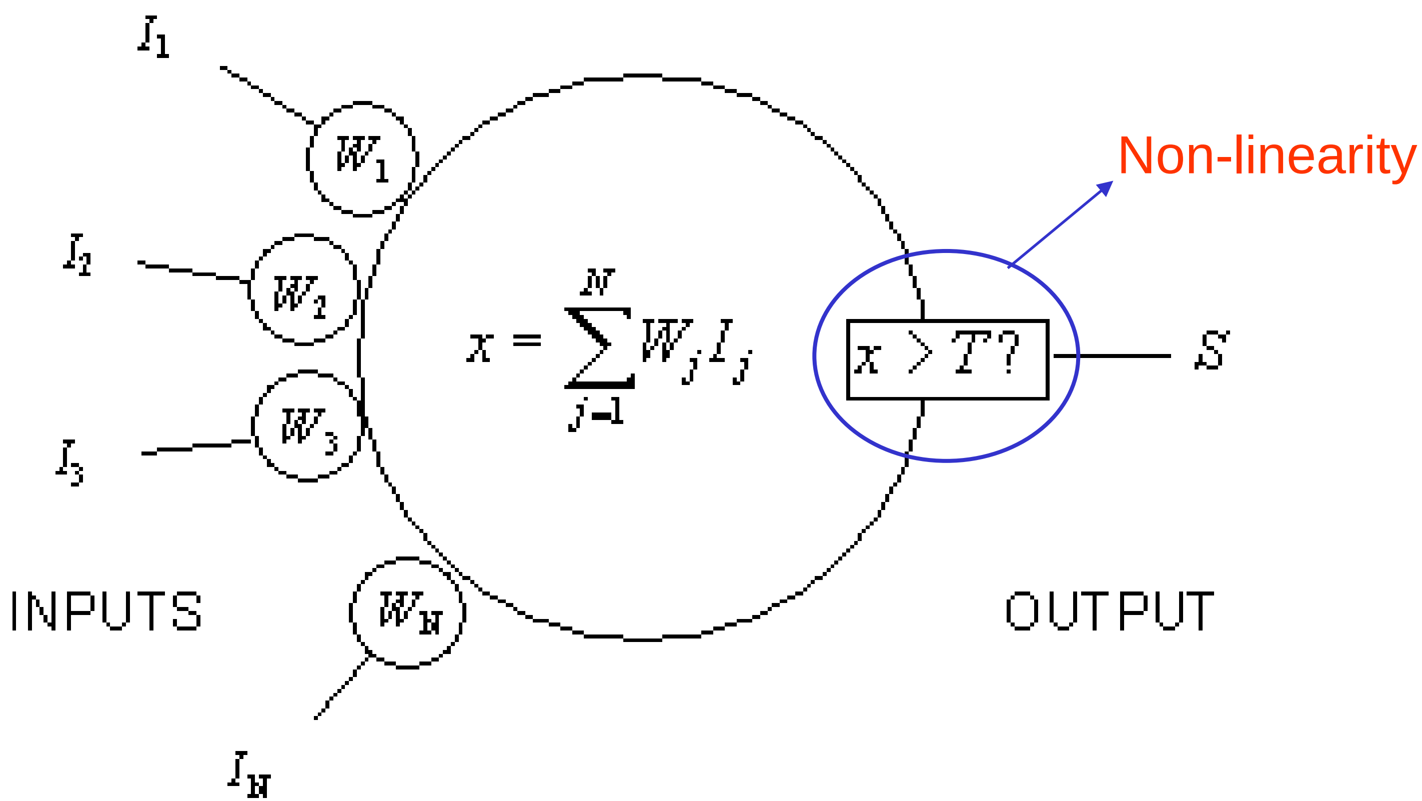
Artificial neurons

Neurons work by processing information. They receive and provide information in form of spikes.

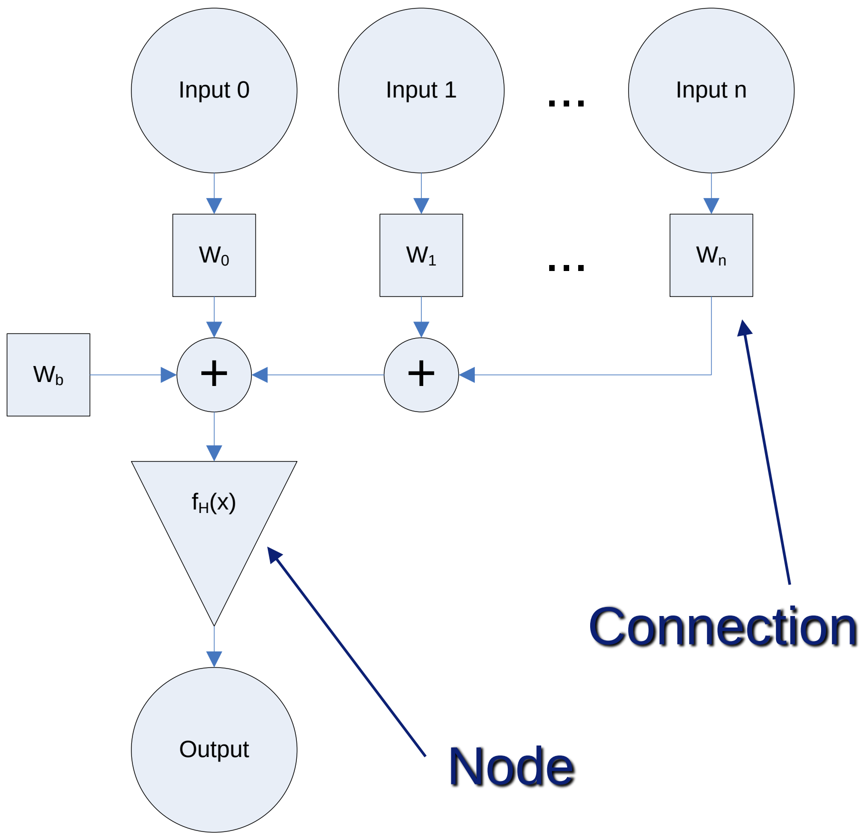


Mathematical representation

The neuron calculates a weighted sum of inputs and compares it to a threshold. If the sum is higher than the threshold, the output is set to 1, otherwise to -1.



Basic Concepts

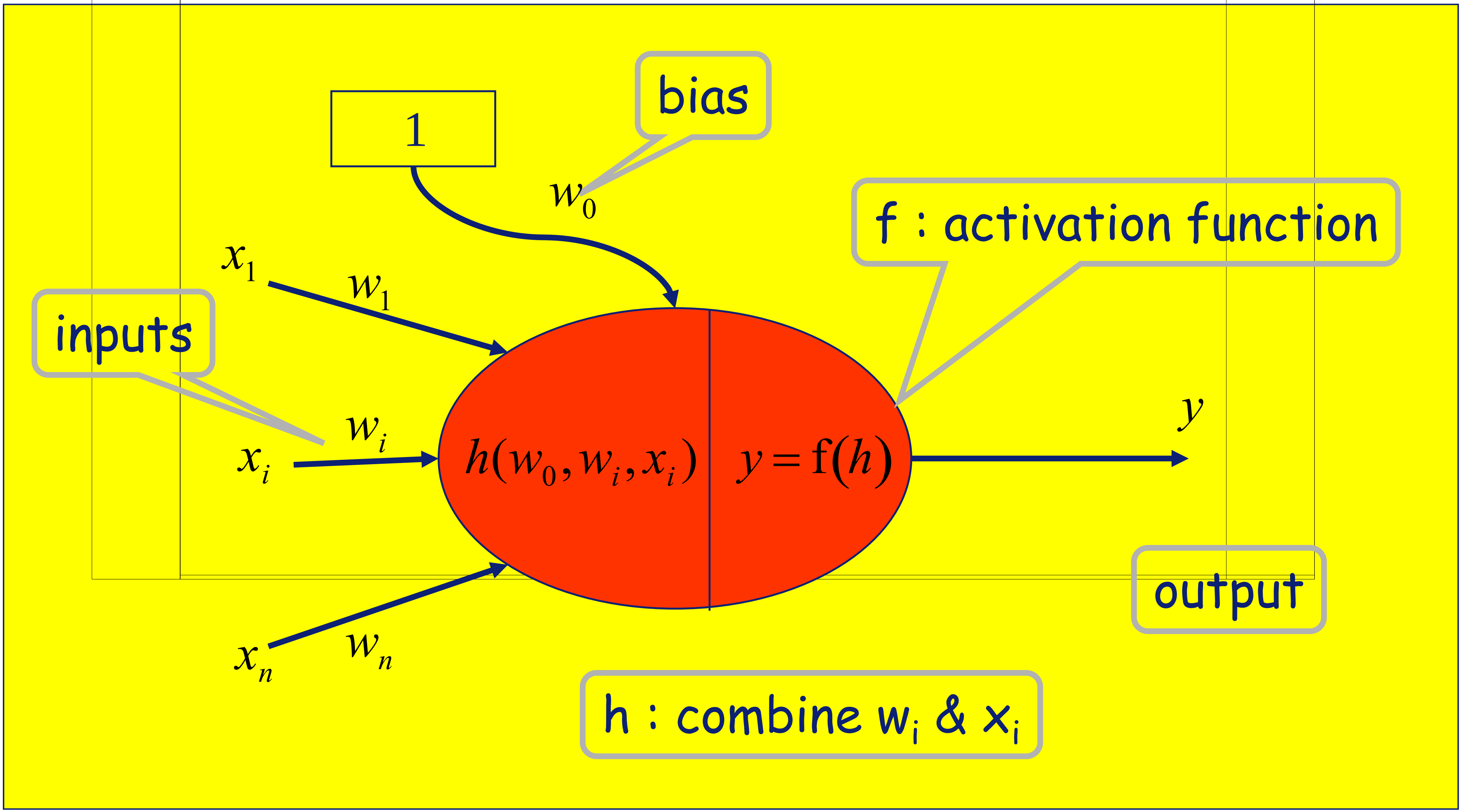


Definition of a node:

- A node is an element which performs the function

  $y = f_H(\sum(w_i x_i) + W_b)$

Anatomy of an Artificial Neuron

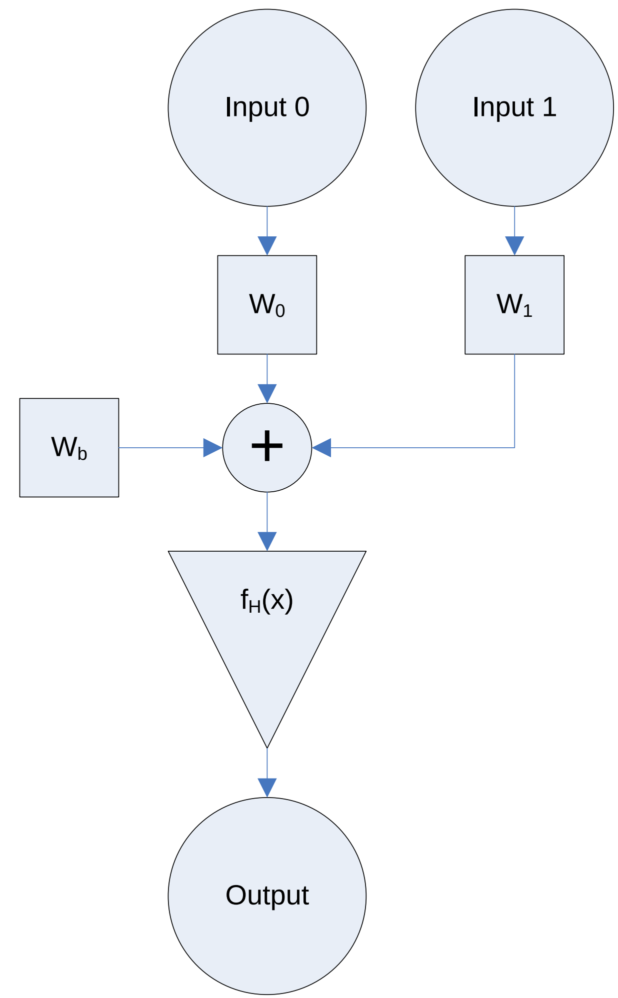


Simple Perceptron

- Binary logic application
- $f_H(x) = u(x)$ [linear threshold]
- $W_i = \text{random}(-1,1)$

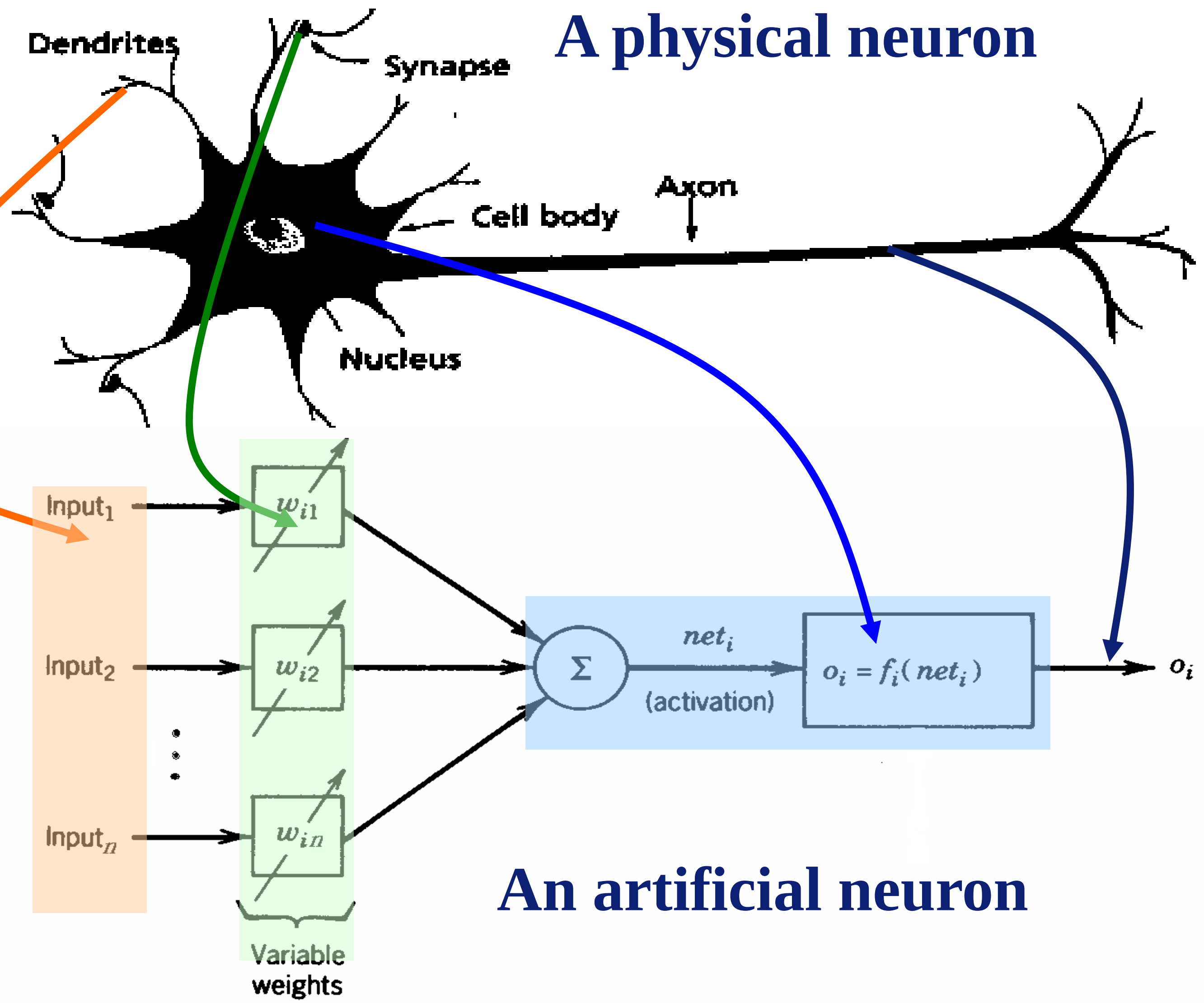
- $Y = u(W_0X_0 + W_1X_1 + W_b)$

- Now how do we train it?

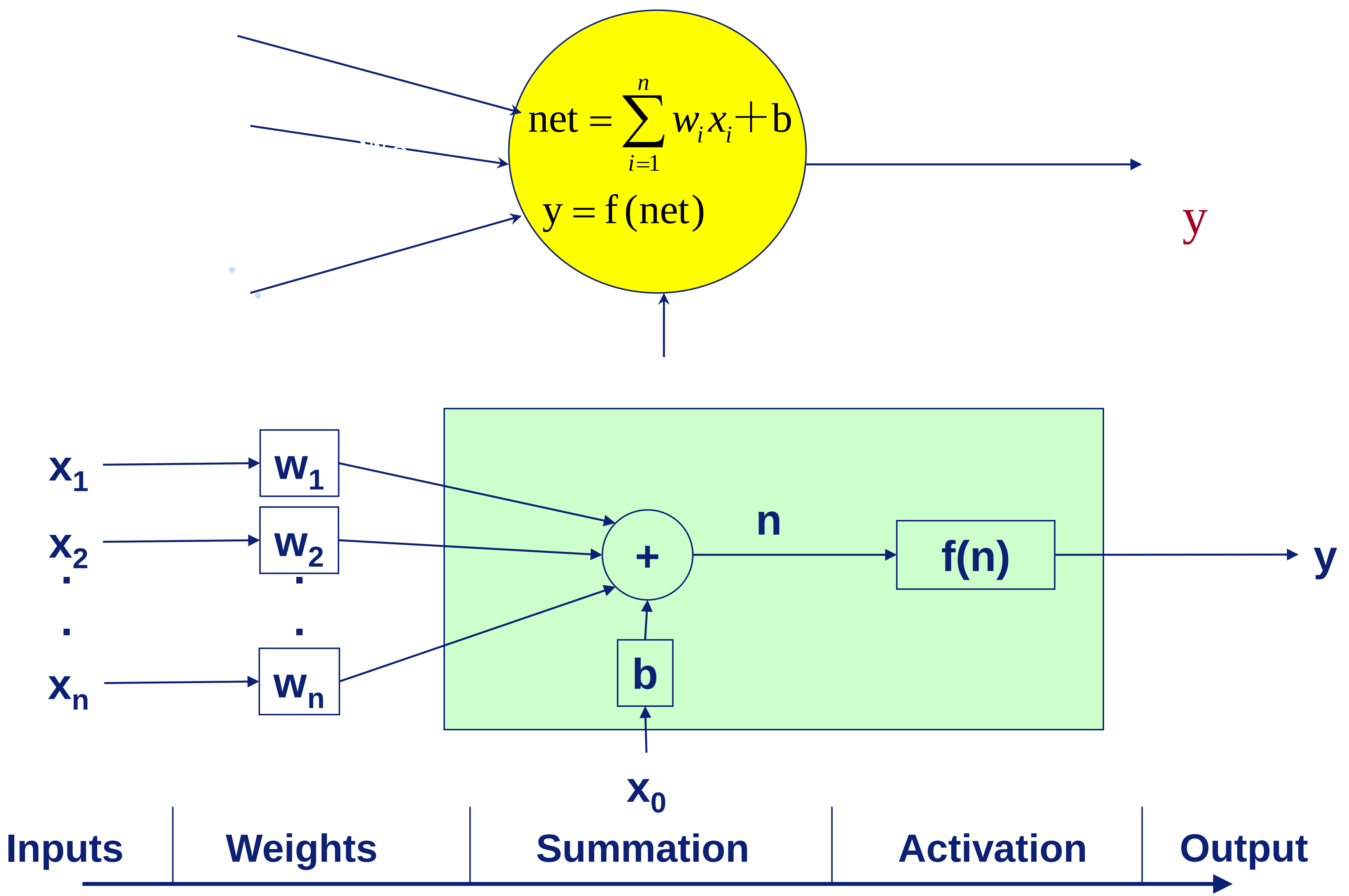


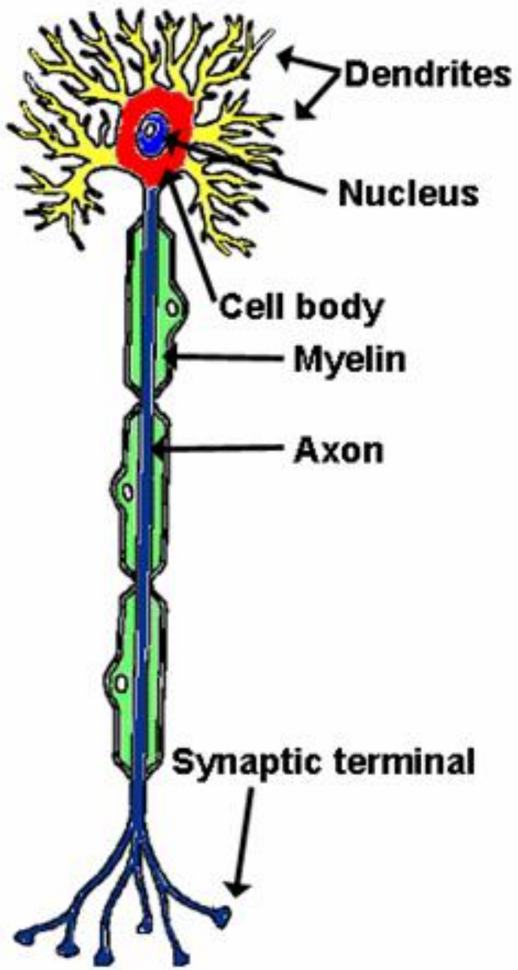
Artificial Neuron

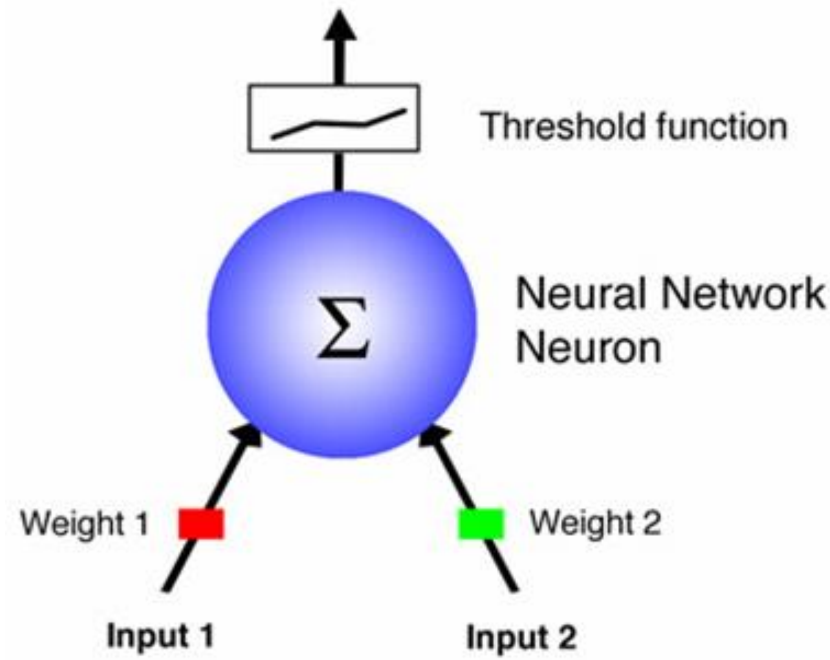
- From experience: examples / training data
- Strength of connection between the neurons is stored as a weight-value for the specific connection.
- Learning the solution to a problem = changing the connection weights



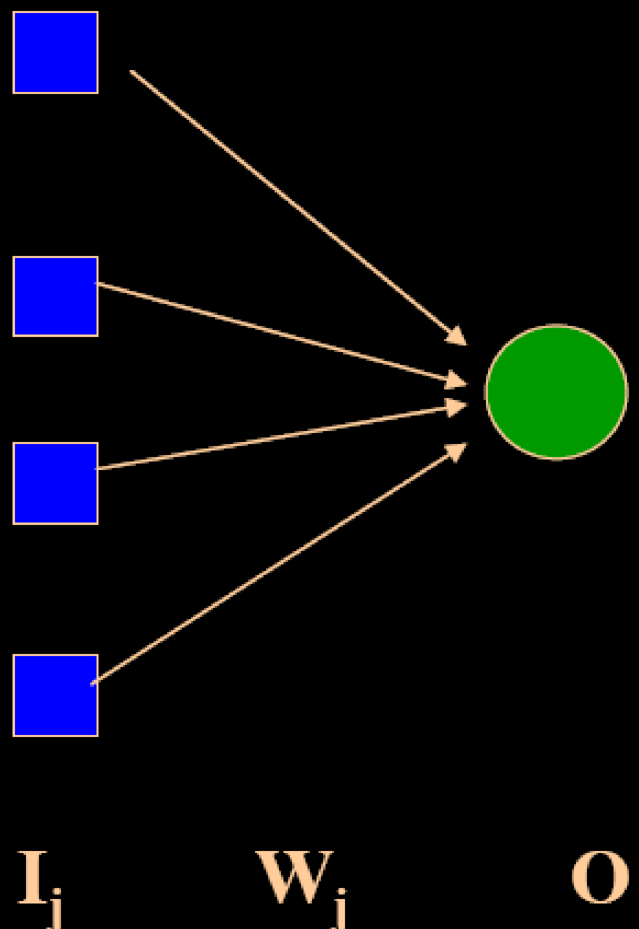
Mathematical Representation




Dendrites
Nucleus
Cell body
Myelin
Axon
Synaptic terminal


Threshold function
Σ
Neural Network
Neuron
Weight 1
Weight 2
Input 1
Input 2

Single Perceptron Learning



$$Err = T - O$$

$$Wj = Wj + \alpha * Ij * Err$$

A perceptron can learn any linearly separable function!

A simple perceptron

- It's a single-unit network
- *Change the weight by an amount proportional to the difference between the desired output and the actual output.*

$$\Delta W_i = \eta * (D-Y).I_i$$

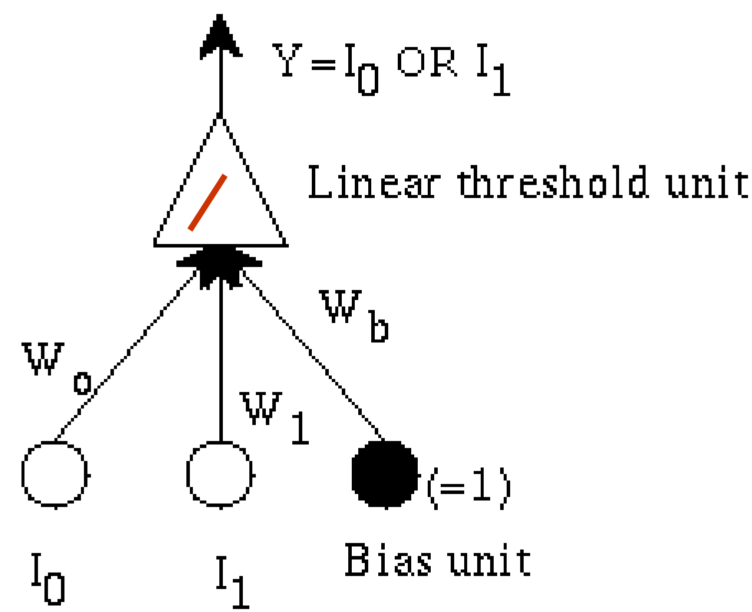
Input

Learning rate

Desired output

Actual output

Perceptron Learning Rule



Linear Neurons

•Obviously, the fact that threshold units can only output the values 0 and 1 restricts their applicability to certain problems.

•We can overcome this limitation by eliminating the threshold and simply turning f_i into the **identity function** so that we get:

$$o_i(t) = \mathrm{net}_i(t)$$

•With this kind of neuron, we can build networks with m input neurons and n output neurons that compute a function

$f: R^m \to R^n$.

Linear Neurons

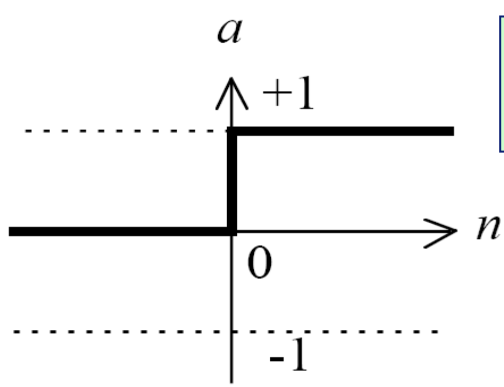
•Linear neurons are quite popular and useful for applications such as interpolation.

•However, they have a serious limitation: Each neuron computes a linear function, and therefore the overall network function $f: R^m \to R^n$ is also **linear**.

•This means that if an input vector x results in an output vector y, then for any factor ϕ the input $\phi \cdot x$ will result in the output $\phi \cdot y$.

•Obviously, many interesting functions cannot be realized by networks of linear neurons.

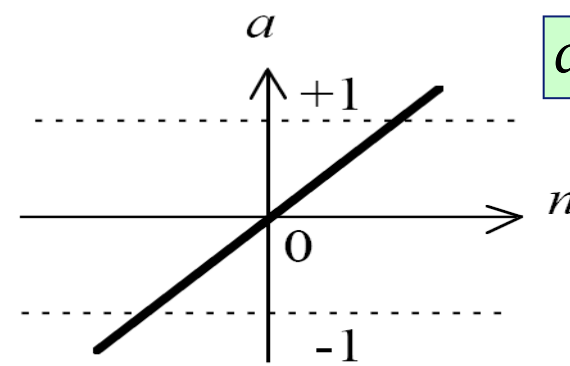
Mathematical Representation



$a = hardlim(n)$

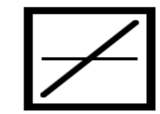
$$a = f(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

Hard-Limit Transfer Function

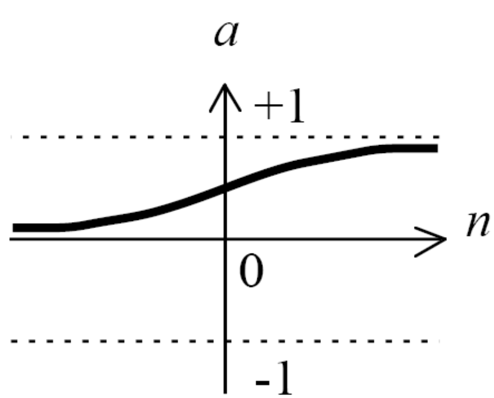


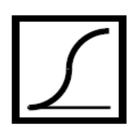
$a = purelin(n)$

$$a = f(n) = n$$



Linear Transfer Function

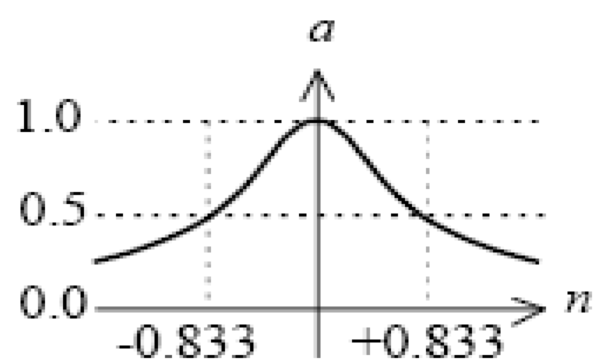


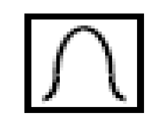


$a = logsig(n)$

$$a = f(n) = \frac{1}{1+e^{-n}}$$

Log-Sigmoid Transfer Function





$a = radbas(n)$

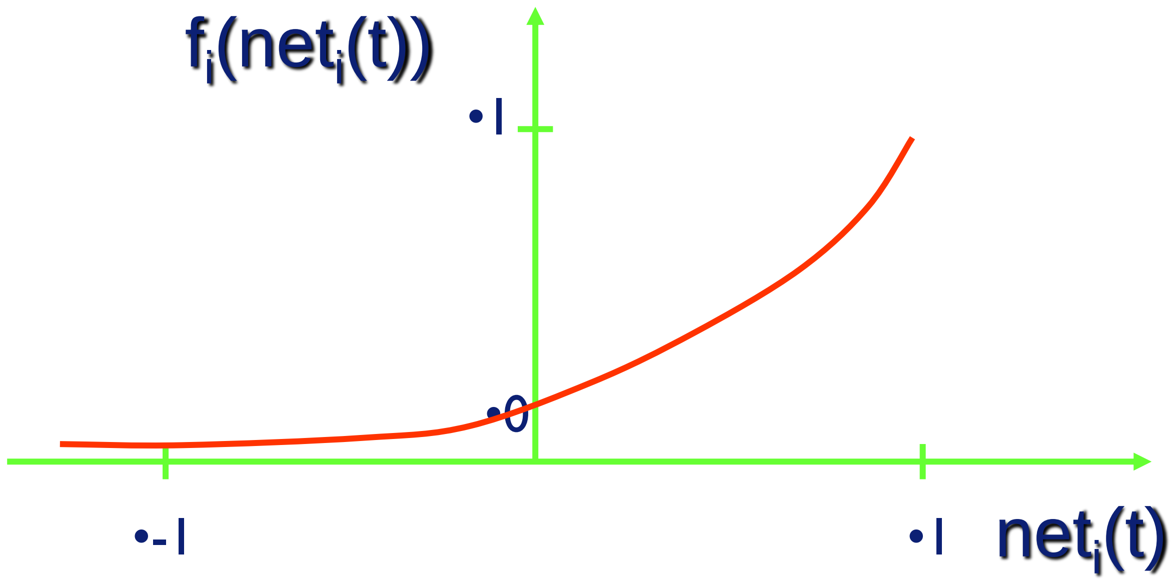
$$a = f(n) = e^{-n^2}$$

Radial Basis Function

Gaussian Neurons

- Another type of neurons overcomes this problem by using a **Gaussian** activation function:

$$f_i(\mathrm{net}_i(t)) = e^{\frac{\mathrm{net}_{i(t)-1}}{\sigma^2}}$$



Gaussian Neurons

•Gaussian neurons are able to realize **non-linear** functions.

•Therefore, networks of Gaussian units are in principle unrestricted with regard to the functions that they can realize.

•The drawback of Gaussian neurons is that we have to make sure that their net input does not exceed 1.

•This adds some difficulty to the learning in Gaussian networks.

Sigmoidal Neurons

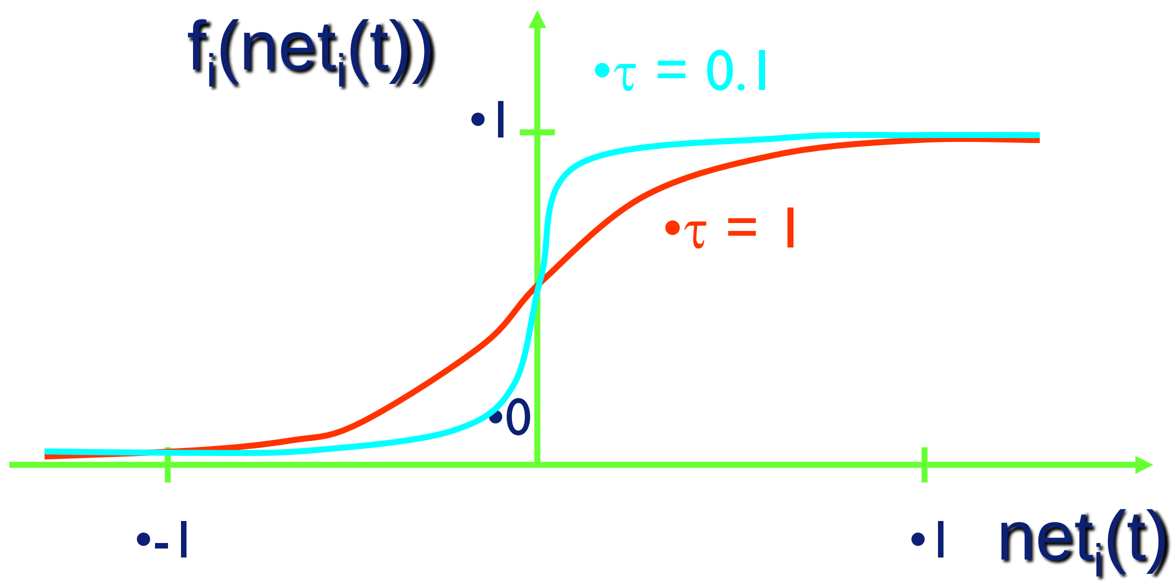
•**Sigmoidal neurons** accept any vectors of real numbers as input, and they output a real number between 0 and 1.

•Sigmoidal neurons are the most common type of artificial neuron, especially in learning networks.

•A network of sigmoidal units with m input neurons and n output neurons realizes a network function
$f: R^m \rightarrow (0,1)^n$

Sigmoidal Neurons

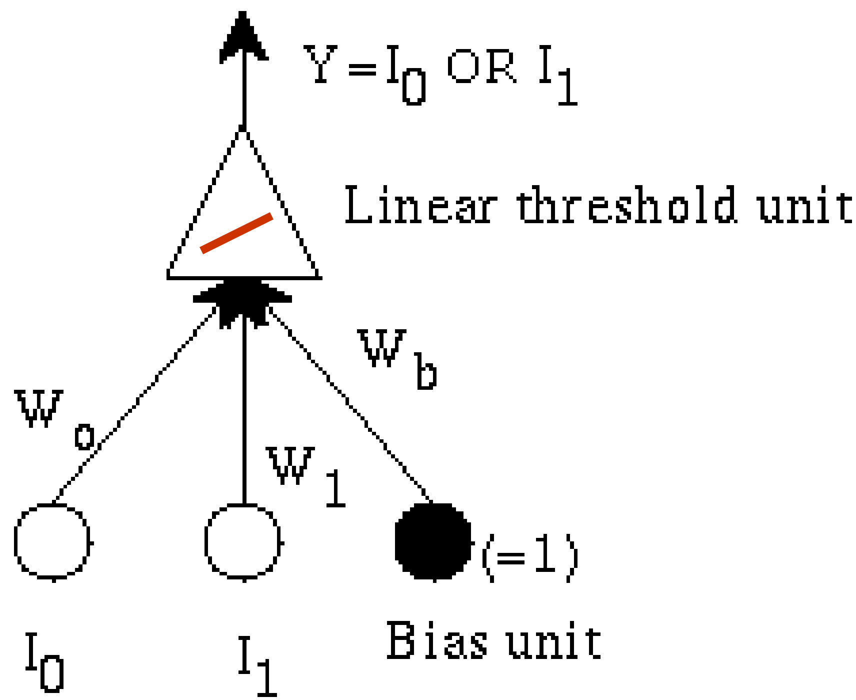
$$f_i(\mathrm{net}_i(t)) = \frac{1}{1 + e^{-(\mathrm{net}_i(t) - \theta)/\tau}}$$



•The parameter τ controls the slope of the sigmoid function, while the parameter θ controls the horizontal offset of the function in a way similar to the threshold neurons.

Example: A simple single unit adaptive network

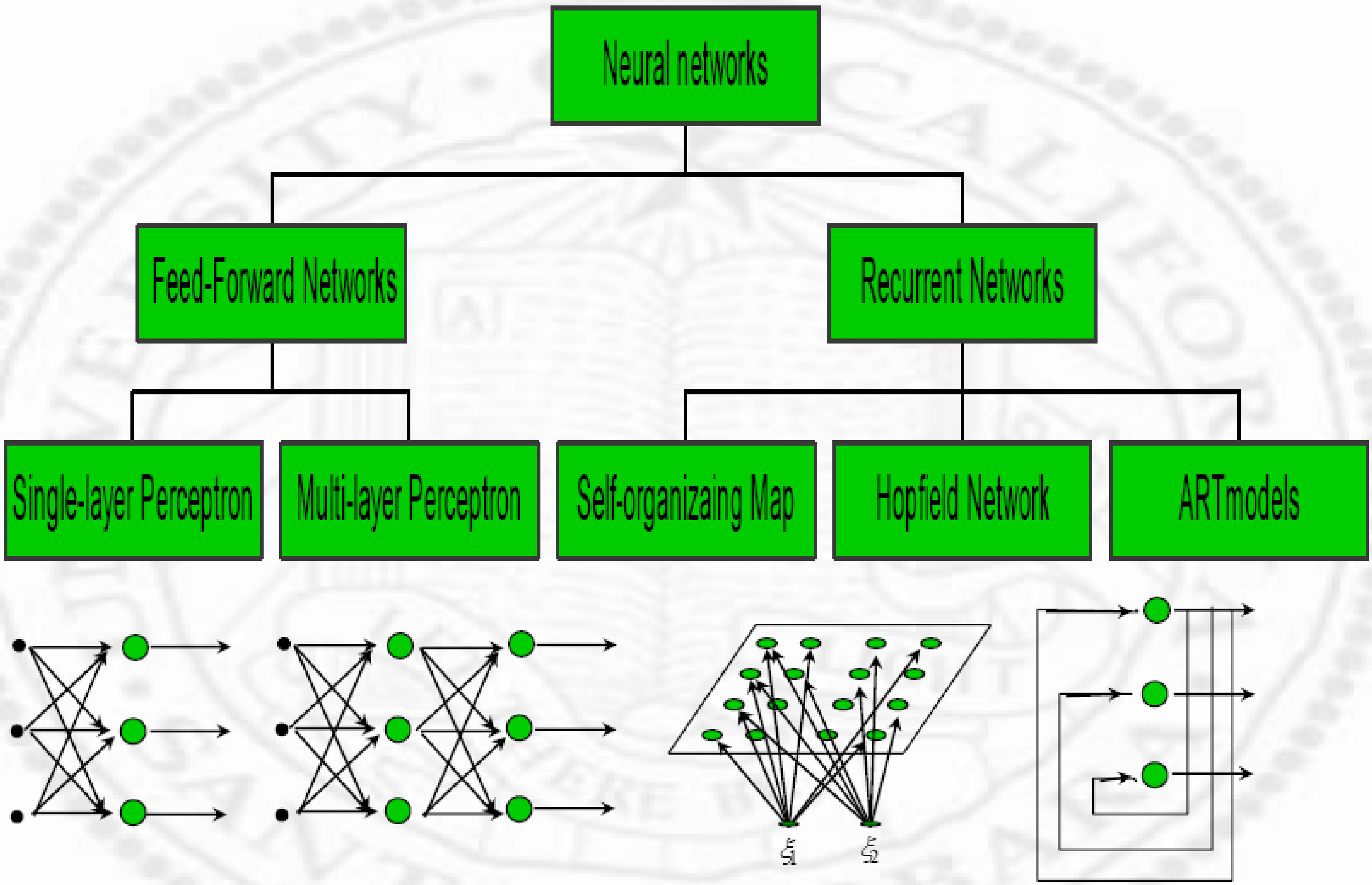
- The network has 2 inputs, and one output. All are binary. The output is
  - 1 if $W_0 I_0 + W_1 I_1 + W_b > 0$
  - 0 if $W_0 I_0 + W_1 I_1 + W_b \leq 0$
- We want it to learn simple OR: output a 1 if either I_0 or I_1 is 1.



NNs: Dimensions of a Neural Network?

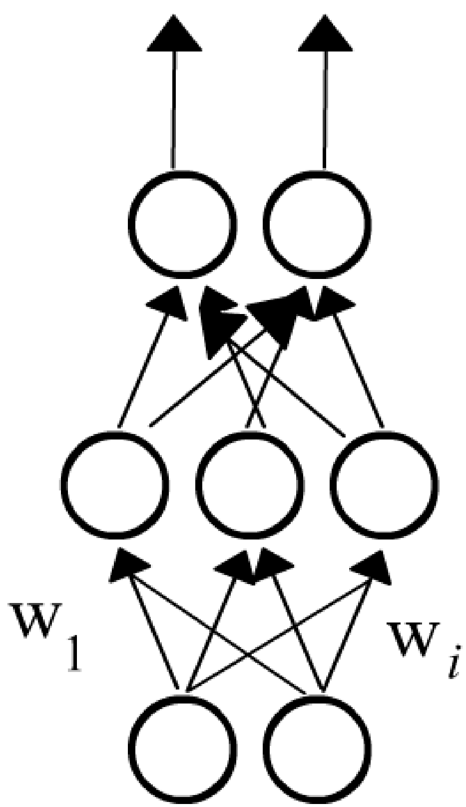
- A NN is specified by:?
  - an architecture: a set of neurons and links connecting neurons. Each link has a weight,
  - a neuron model: the information processing unit of the NN,
  - a learning algorithm: used for training the NN by modifying the weights in order to solve the particular learning task correctly on the training examples.

Computational Network Architecture



Adaptive resonance theory-Copy back

Supervised learning in feedforward networks--Backpropagation



In supervised learning, we need some *training data*, which is a set of input values that have some desired outputs [d].

Given a particular feedforward net, one can calculate the output o given an input i from the training data.

$$\sum (d - o)^2$$ **--Sum Square Error**

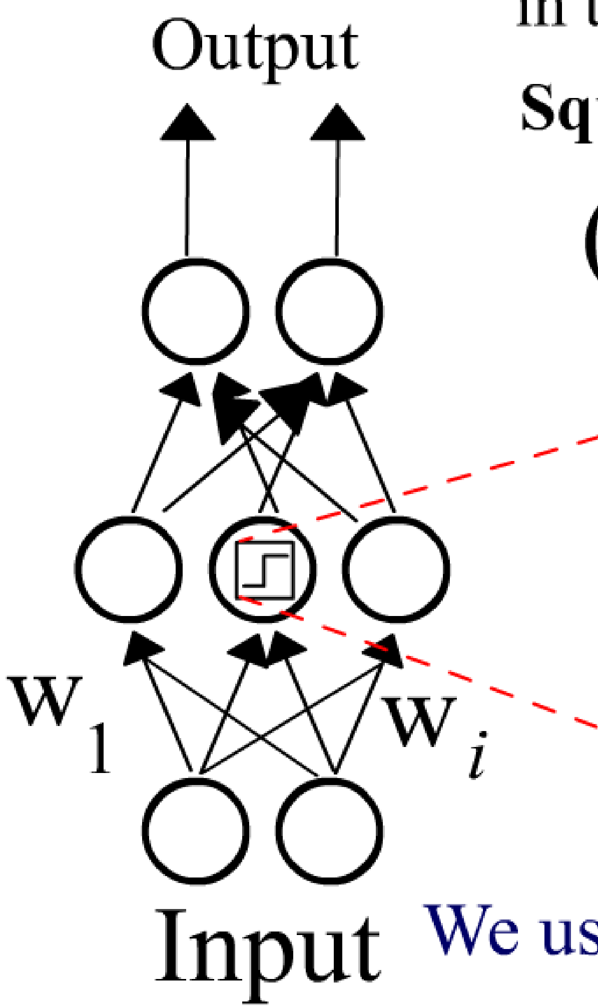
Goal: Minimize Sum Square Error

Backpropagation: A *little* math

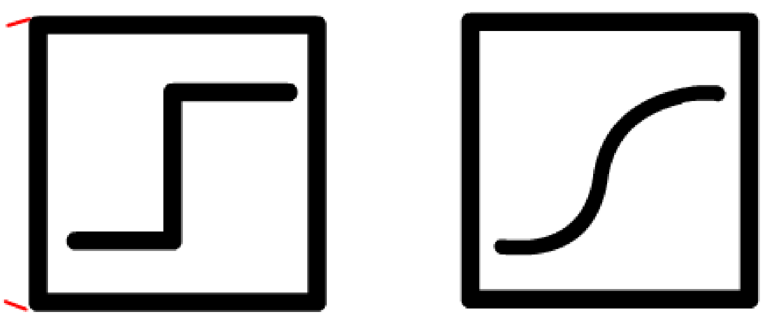
The squared error for each input can be expressed in terms of w and i

Squared Error

$$(d - o)^2 = f(w, i)$$



New version

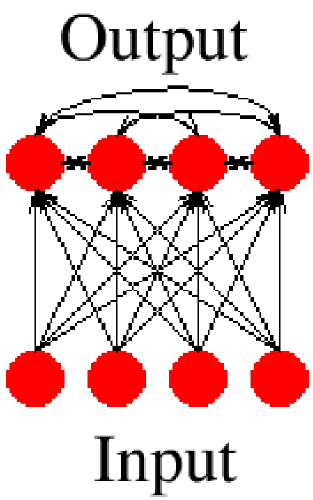


Differentiable

We use the following formula to adjust the weights

$$w_{new} = w_{old} - n\frac{\partial f}{\partial w}$$

Unsupervised learning



The goal of unsupervised learning is to discover trends in a data set without prior knowledge of the answers.

- We train the network using only inputs without knowing a desired output; hence we cannot modify the weights using backpropagation
- The network discovers special features and patterns from available data through competition
- The weight factors in the network are modified (i.e. the network learns) in such a way that data that are similar will trigger the same output

Feed-forward networks:

Advantages:

-lack of cycles
-it has no internal state other than the weights themselves.
-fixed structure and fixed activation function g:

Learning in biological systems

-At the neural level the learning happens by
-changing of the synaptic strengths,
-eliminating some synapses, and
-building new ones.

Learning as optimisation

the objective of learning in biological organisms is to optimise the amount of available resources, happiness, or in general to achieve a closer to optimal state.

Synapse concept

Hebb's Rule:

If an input of a neuron is repeatedly and persistently causing the neuron to fire, a metabolic change happens in the synapse of that particular input to reduce its resistance

Supervised Learning in ANNs

•In supervised learning, we train an ANN with a set of vector pairs, so-called **exemplars**.

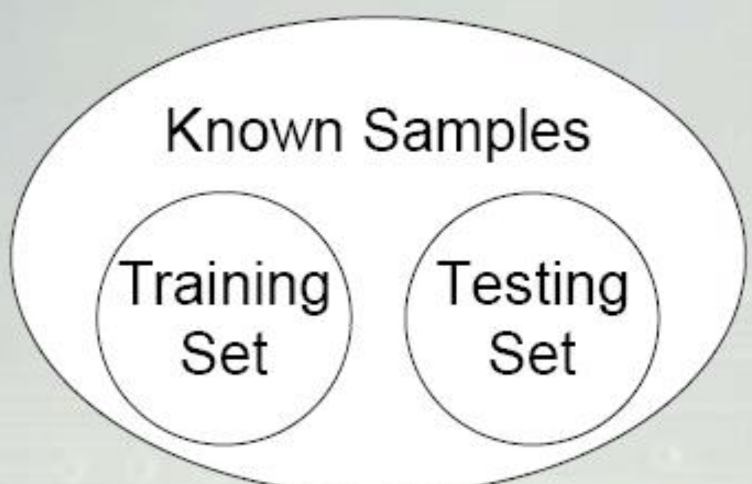
•Each pair (**x**, **y**) consists of an input vector **x** and a corresponding output vector **y**.

How is the Training Set Chosen?

- Overfitting can also occur if a “good” training set is not chosen
- What constitutes a “good” training set?
  - Samples must represent the general population
  - Samples must contain members of each class
  - Samples in each class must contain a wide range of variations or noise effect

Training and Verification

- The set of all known samples is broken into two orthogonal (independent) sets:
  - Training set
    - A group of samples used to train the neural network
  - Testing set
    - A group of samples used to test the performance of the neural network
    - Used to estimate the error rate



Verification

- Provides an unbiased test of the quality of the network
- Common error is to "test" the neural network using the same samples that were used to train the neural network
  - The network was optimized on these samples, and will obviously perform well on them
  - Doesn't give any indication as to how well the network will be able to classify inputs that weren't in the training set

Learning in Neural Nets

Supervised

Data:
Labeled examples
(input , desired output)

Tasks:
classification
pattern recognition
regression
NN models:
perceptron
adaline
feed-forward NN
radial basis function
support vector machines

Unsupervised

Data:
Unlabeled examples
(different realizations of the input)

Tasks:
clustering
content addressable memory

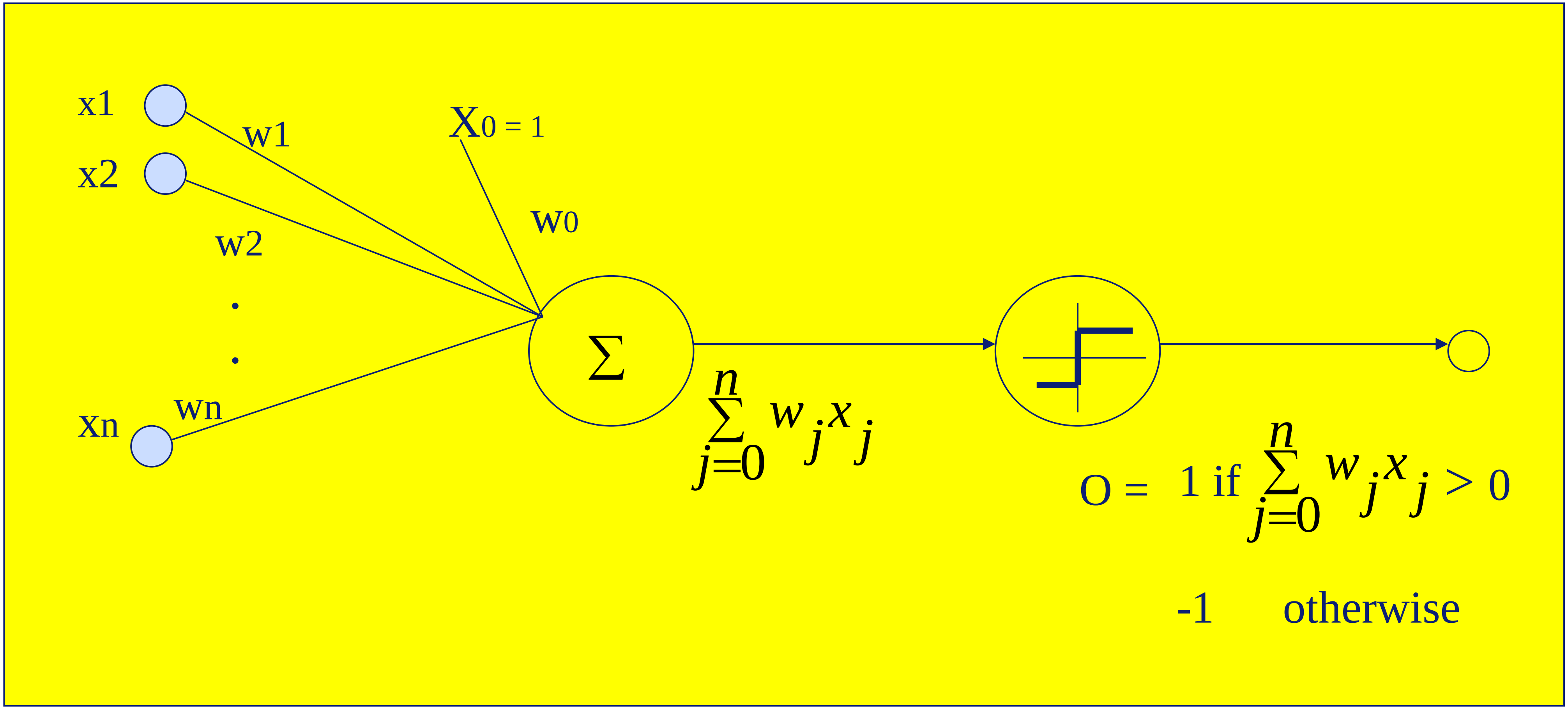
NN models:
self-organizing maps (SOM)
Hopfield networks

Learning Algorithms types

Depend on the network architecture:

- Error correcting learning (perceptron)
- Delta rule (AdaLine, Backprop)
- Competitive Learning (Self Organizing Maps)

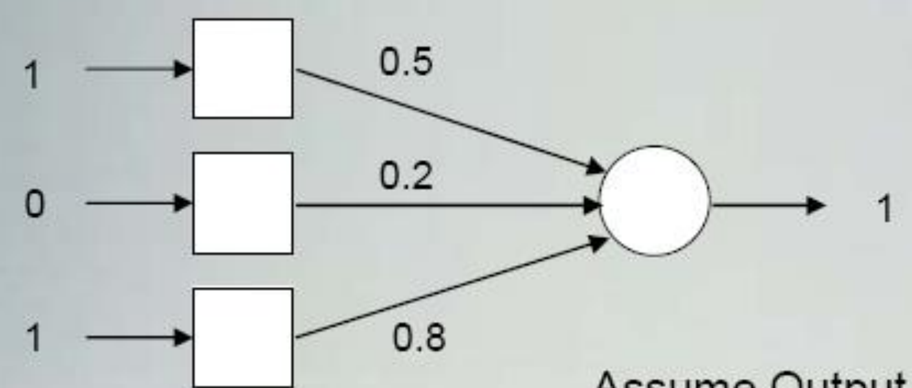
Perceptron



x1
x2
xn
w1
w2
wn
X0 = 1
w0
Σ
$\sum_{j=0}^{n} w_j x_j$
O = 1 if $\sum_{j=0}^{n} w_j x_j > 0$
-1 otherwise

How Do Perceptrons Learn?

- Uses supervised training
- If the output is not correct, the weights are adjusted according to the formula:
  - $W_{new} = W_{old} + \alpha(\text{desired} - \text{output})*\text{input}$ α is the learning rate



$1 * 0.5 + 0 * 0.2 + 1 * 0.8 = 1.3$

Assuming Output Threshold = 1.2

$1.3 > 1.2$

Assume Output was supposed to be 0
→ update the weights

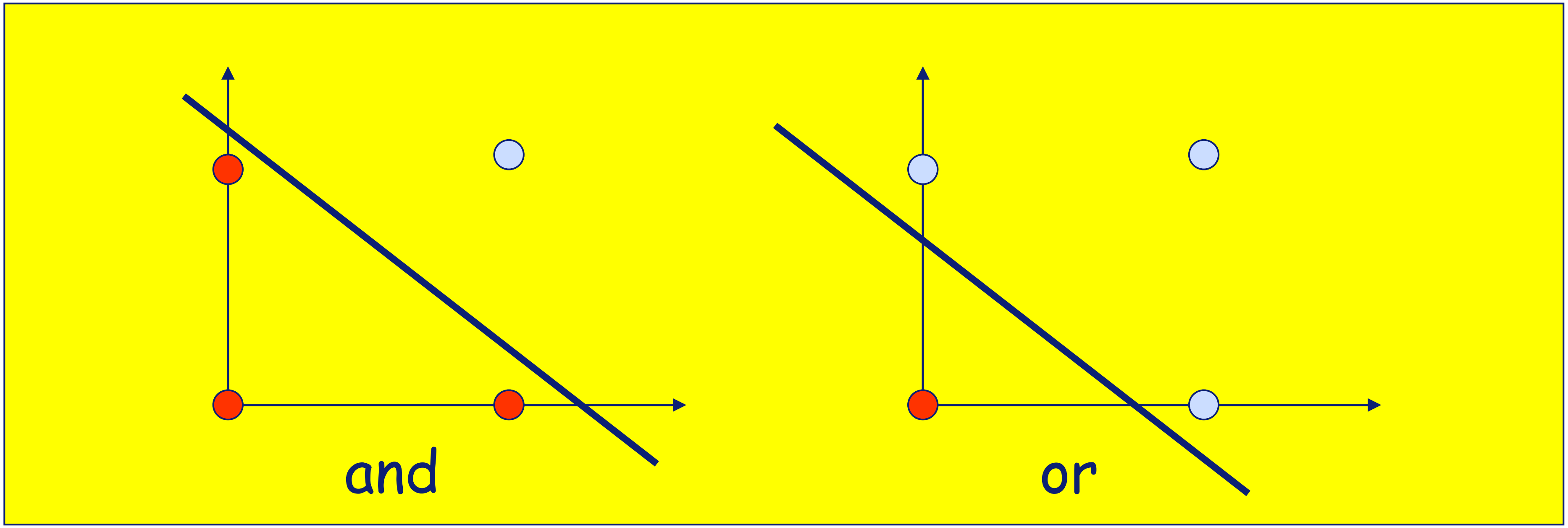
Assume $\alpha = 1$

$W_{1new} = 0.5 + 1*(0-1)*1 = -0.5$
$W_{2new} = 0.2 + 1*(0-1)*0 = 0.2$
$W_{3new} = 0.8 + 1*(0-1)*1 = -0.2$

Linear Separable

some functions not representable - e.g., (b) not linearly separable

So what can be represented using perceptrons?



Representation theorem: 1 layer feedforward networks can only represent linearly separable functions. That is, the decision surface separating positive from negative examples has to be a plane.

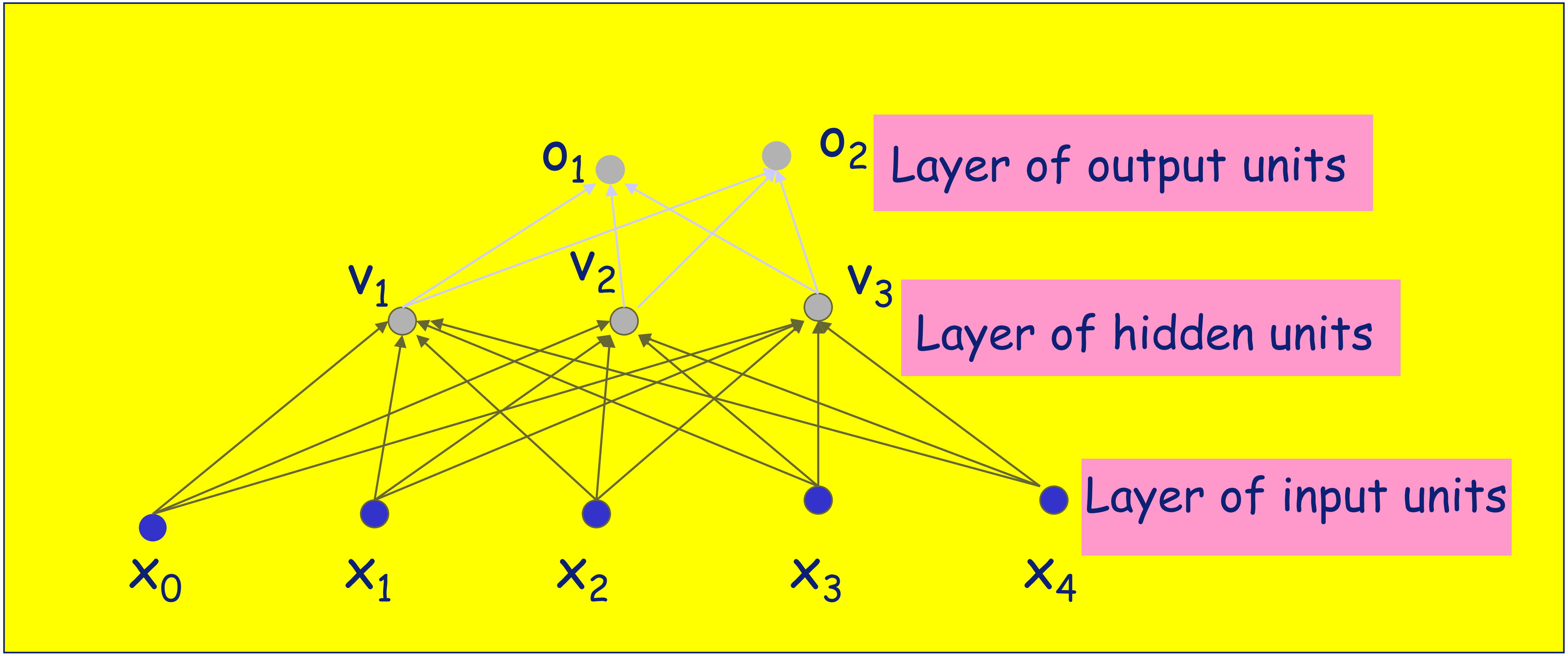
The perceptron learning algorithm

- Inputs: training set $\{(x_1,x_2,\ldots,x_n,t)\}$
- Method
  - Randomly initialize weights w(i), -0.5<=i<=0.5
  - Repeat for several epochs(periods of time) until convergence:
    - for each example
      - Calculate network output o.
      - Adjust weights:

learning rate error

$$\left.\begin{aligned} \Delta w_i &= \eta(t-o)x_i \\ w_i &\leftarrow w_i + \Delta w_i \end{aligned}\right\}$$ Perceptron training rule

Multilayer feed forward network



Weight updating in backprop

- Learning in backprop is similar to learning with perceptrons, i.e.,
  - Example inputs are fed to the network.
    - If the network computes an output vector that matches the target, nothing is done.
    - If there is a difference between output and target (i.e., an error), then the weights are adjusted to reduce this error.
    - The key is to assess the blame for the error and divide it among the contributing weights.

- **Estimating Error**
  - each hidden node contributes for some fraction of the error in each of the output nodes.

Back-propagation algorithm for updating weights in a multilayer network

1.Initialize the weights in the network (often randomly)
2.**repeat**
  for each example e in the training set **do**
   i.O = neural-net-output(network, e) ; forward pass
   ii.T = teacher output for e
   iii.Calculate error (T - O) at the output units
   iv.Compute **wj = wj +α * Err * Ij** for all weights from
    hidden layer to output layer;backward pass
   v.Compute **wj = wj +α * Err * Ij** for all weights from input layer
    to hidden layer; backward pass continued
   vi.Update the weights in the network
  end
3.**until** all examples classified correctly or stopping criterion met
4.**return**(network)

Back-propagation Using Gradient Descent

- Advantages
  - Relatively simple implementation
- Disadvantages
  - Slow and inefficient

Number of training pairs needed?

Difficult question. Depends on the problem, the training examples, and network architecture. However, a good rule of thumb is:

Where W = No. of weights; P = No. of training pairs, e = error rate

$$\frac{w}{p} = e$$

For example, for e = 0.1, a net with 80 weights will require 800 training patterns to be assured of getting 90% of the test patterns correct (assuming it got 95% of the training examples correct).

How long should a net be trained?

- correct responses for the training patterns and correct responses for new patterns
- If you train the net for too long, then you run the risk of overfitting.

Determining optimal network structure??

Weak point of fixed structure networks: poor choice can lead to poor performance

Too small network: model incapable of representing the desired Function

Too big a network: will be able to memorize all examples but forming a large lookup table, but will not generalize well to inputs that have not been seen before.

Thus finding a good network structure is another example of a search problems.
Some approaches to search for a solution for this problem include Genetic algorithms
But using GAs is very cpu-intensive.

Neural Networks: Advantages

- Distributed representations
- Simple computations
- Robust with respect to noisy data
- Robust with respect to node failure
- Empirically shown to work well for many problem domains
- Parallel processing
- Disadvantages
- Training is slow
- Interpretability is hard
- Network topology layouts ad hoc
- Can be hard to debug
- May converge to a local, not global, minimum of error
- May be hard to describe a problem in terms of features with numerical values

Brain vs. Digital Computers (1)

Computers require hundreds of cycles to simulate a firing of a neuron

The brain can fire all the neurons in a single step.

➤ Parallelism

Serial computers require billions of cycles to perform some tasks but the brain takes less than a second

e.g. Face Recognition

Training Back Prop Net: Feedforward Stage??

1. Initialize weights with small, random values
2. While stopping condition is not true
   - for each training pair (input/output):
     - each input unit broadcasts its value to all hidden units
     - each hidden unit sums its input signals & applies activation function to compute its output signal
     - each hidden unit sends its signal to the output units
     - each output unit sums its input signals & applies its activation function to compute its output signal

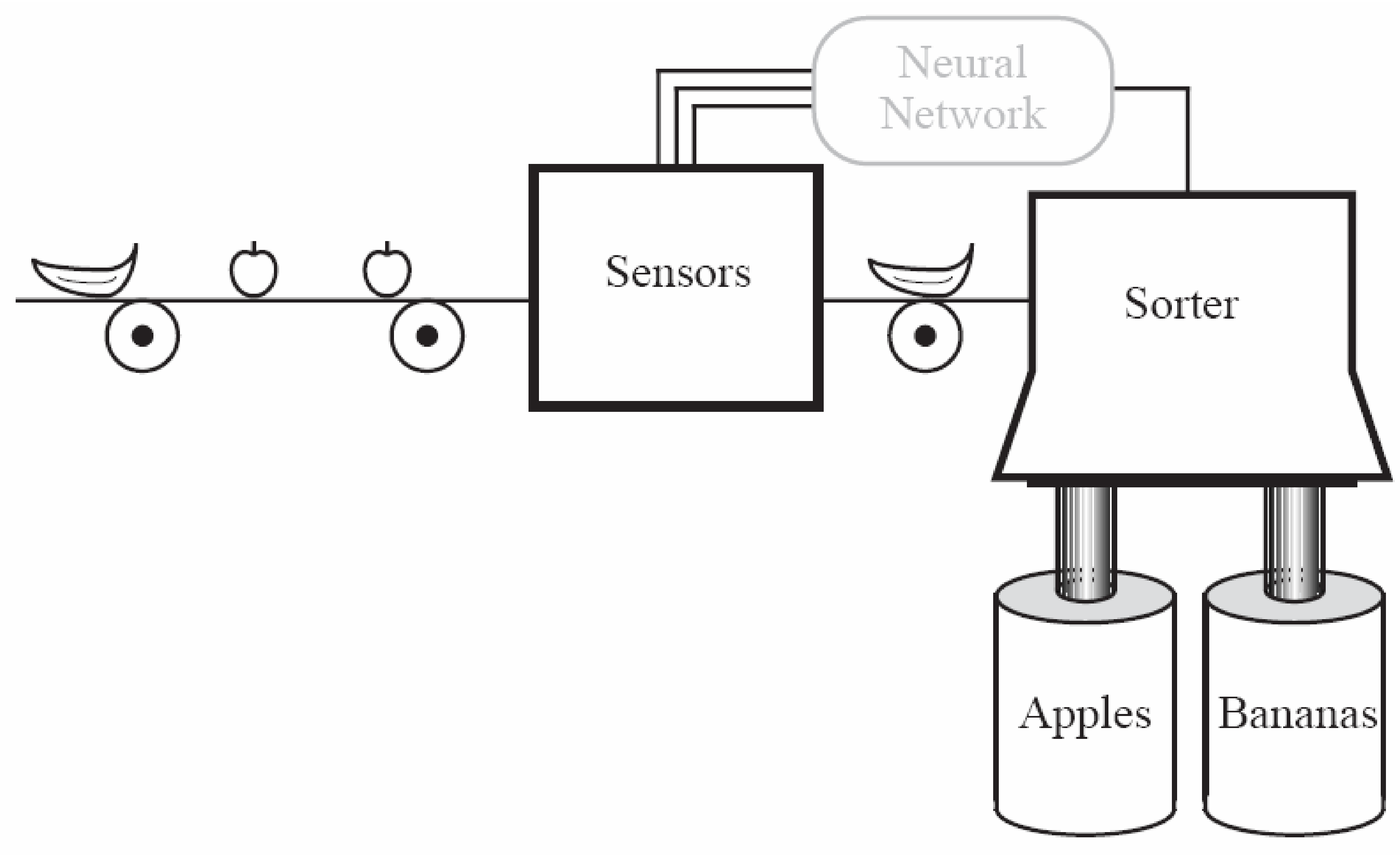
Training Back Prop Net: Backpropagation

3. Each output computes its error term, its own weight correction term and its bias(threshold) correction term & sends it to layer below
4. Each hidden unit sums its delta inputs from above & multiplies by the derivative of its activation function; it also computes its own weight correction term and its bias correction term

Training a Back Prop Net: Adjusting the Weights

5. Each output unit updates its weights and bias
6. Each hidden unit updates its weights and bias

Apples/Bananas Sorter



Questions? Suggestions ?