MONASH University

Information Technology

FIT5186 Intelligent Systems

Lecture 2

Neuron Learning and Perceptrons

Learning Objectives

- Understand
  - the fundamental properties of intelligent systems
  - the concept and process of learning in intelligent systems
  - the perceptron learning algorithm for classification
  - the limitations of perceptrons
- Be able to
  - appreciate the range of neural networks and data mining applications
  - perform simple character recognition tasks using perceptrons

Biological Neurons

- Ramóny Cajál (1911) introduced the idea of a neuron as the structural building block of the brain.
  - Billions of highly interconnected neurons.
  - Estimated 100 billion neurons and 60 trillion synapses or connections in human cortex!
- Neurons communicate with each other by sending electrical impulses along connecting synapses.
- Each neuron accepts input from other neurons, and if enough active inputs are received at once, then the neuron will be activated and “fire”.

  If not it remains inactive.

Biological Neurons (continued)

- Nervous system is a 3 stage process:
  - Input provided by sensory receptors;
  - Information processed;
  - Effectors are controlled, give human response.
- The second stage “information processed” is the most interesting one.
- Information is evaluated and compared with stored information (memory).
- With the right mechanisms in place, learning and thinking can then be achieved in this environment.

What is Learning Anyway?

- Learning is the modification of behaviour through experience.
- Something has to have changed as a result of this experience.
- There is a biological change that occurs in the brain.
- In the brain, synapses connecting certain parts of the brain get stronger and others get weaker to reflect the learning process.

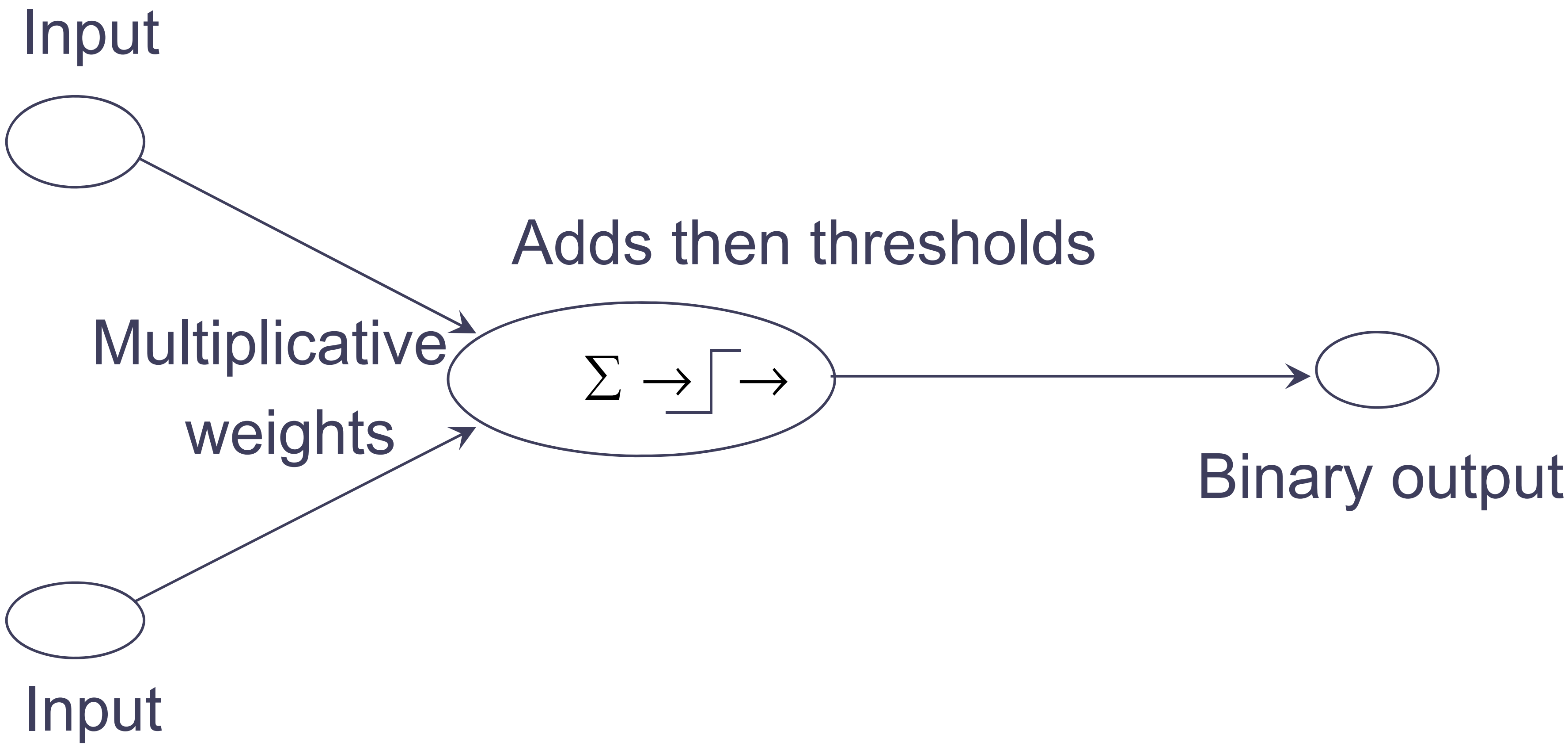
Simple Artificial Neuron Models

- We use a similar principle in the design of artificial neural network models.
- By using simple models of the brain's structure and behaviour, neural networks combine advantages of serial computers (speed and power) with thinking and learning ability of humans.

Simple Artificial Neuron Models (continued)

- Several important features of the brain which need to be captured:
  - Highly interconnected network of simple processing elements.
  - Output from neurons is either on or off.
  - Output depends on inputs.
  - Total input must reach a certain level to make a neuron fire.
  - Amount of input signals received by a neuron depends on the strength of connection.

Outline of A Basic Model

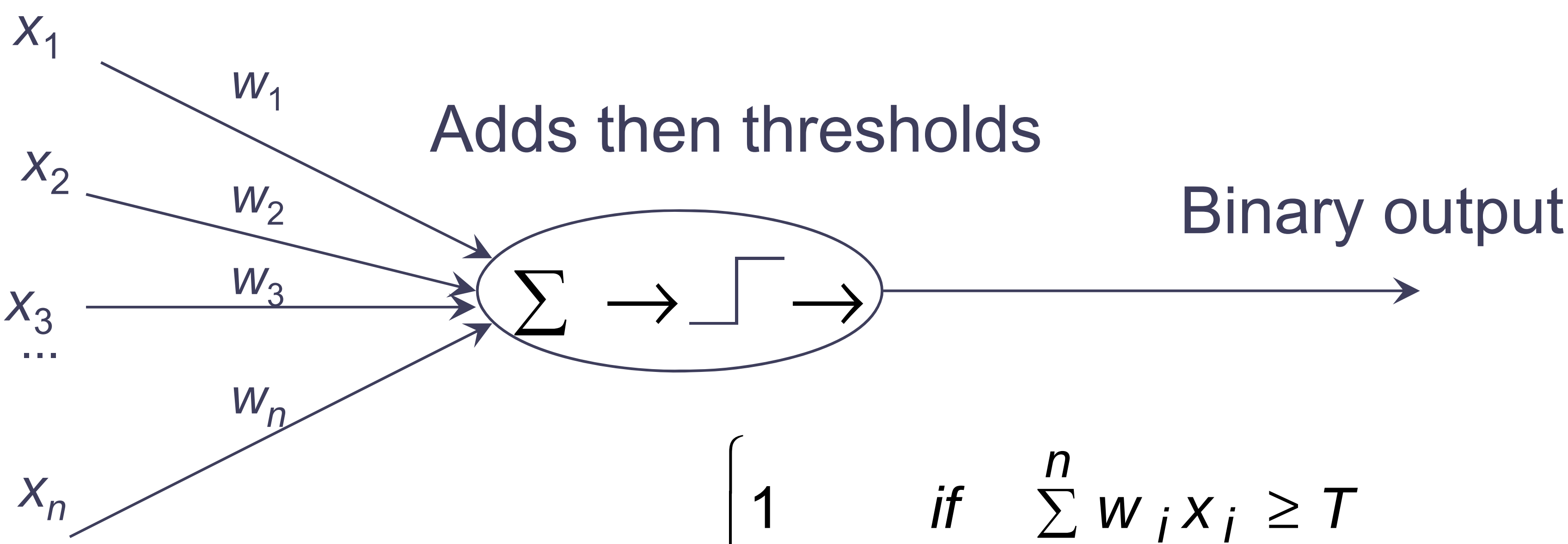


The McCulloch-Pitts Model

- This simple model is known as the McCulloch-Pitts model.
  - Inputs are multiplied by weights and added together (= net input). $\longrightarrow$ Weighted Sum
  - If this net input exceeds a threshold value, the neuron fires.
  - This is the McCulloch-Pitts model of the neuron (1943).
  - Notation:

$$\sum_{i=1}^{n} x_i w_i = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n$$

Detail of the McCulloch-Pitts Model



Output: $$O = \begin{cases} 1 & \text{if} \quad \sum\limits_{i=1}^{n} w_i x_i \geq T \\ 0 & \text{if} \quad \sum\limits_{i=1}^{n} w_i x_i < T \end{cases}$$

T is the threshold value.

Nature of the McCulloch-Pitts Model

- A highly simplified model.
- No complex patterns and timings of actual neuron activity in brain are attempted.
  - Excitatory synapses are given a weight value of +1.
  - Inhibitory synapses are given a weight value of -1.
  - Neurons are only allowed a state of 0 or 1.
  - All neurons are assumed to calculate simultaneously or synchronously (together).
  - No interaction between neurons is modelled.

Logic Operators of the McCulloch-Pitts Model

- Despite these limitations, this simplified model can perform basic logic operations (NOT, OR, AND).
- Recall: “1” is like True, “0” is like False.

x	NOT(x)
1	0
0	1

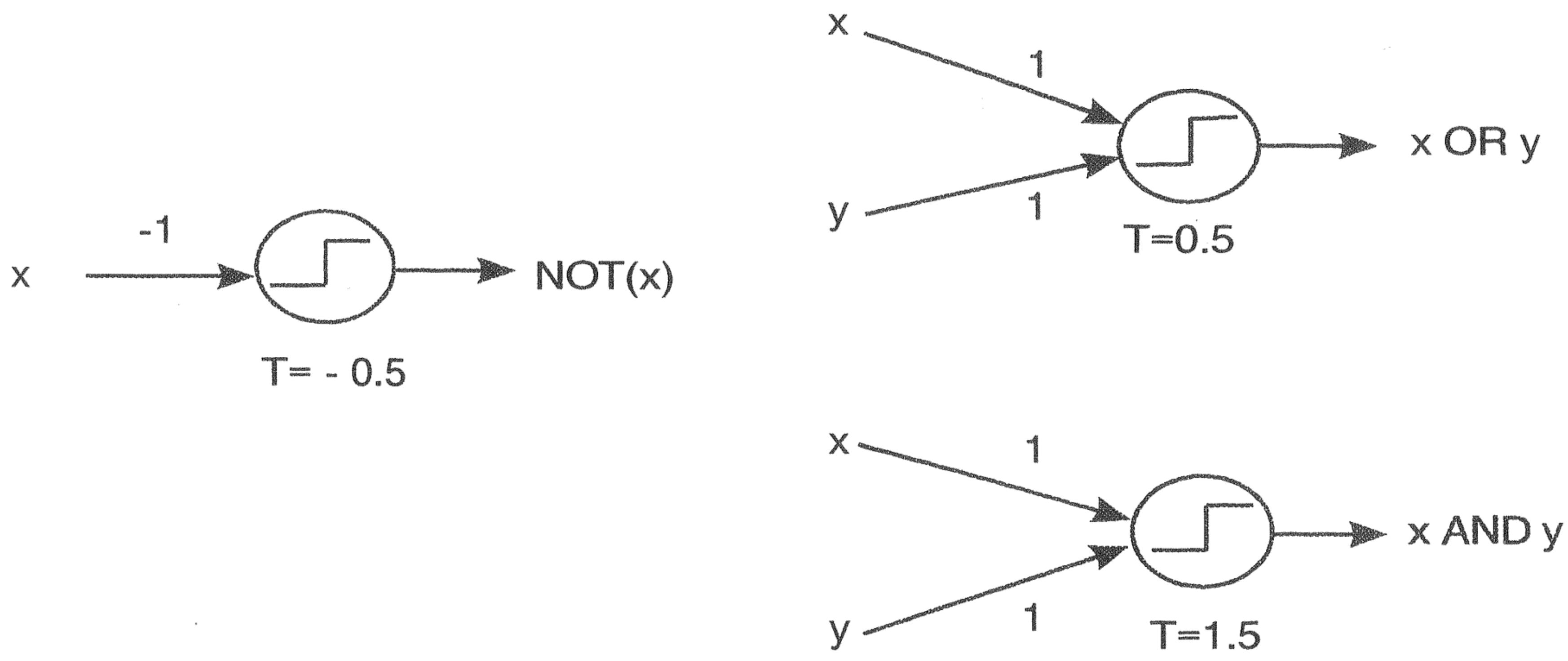
x	y	x OR y
1	1	1
1	0	1
0	1	1
0	0	0

x	y	x AND y
1	1	1
1	0	0
0	1	0
0	0	0

- Also for 3 inputs (x,y,z): e.g. 1 OR 0 OR 0 = 1
  1 AND 0 AND 0 = 0

Logic Operators of the McCulloch-Pitts Model

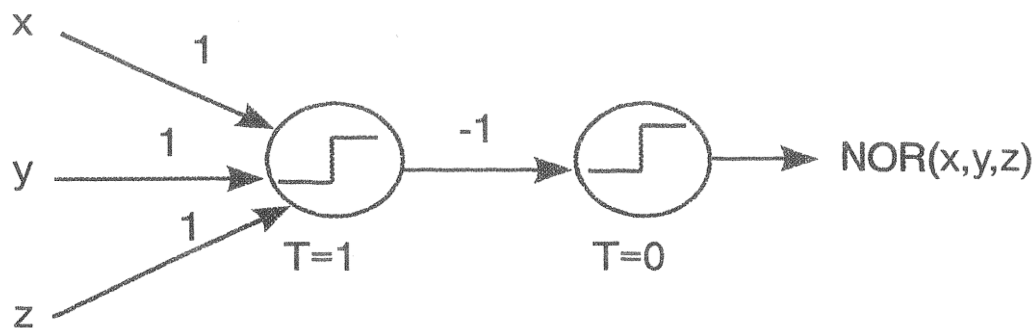
Examples of logic operations NOT, OR, and AND.

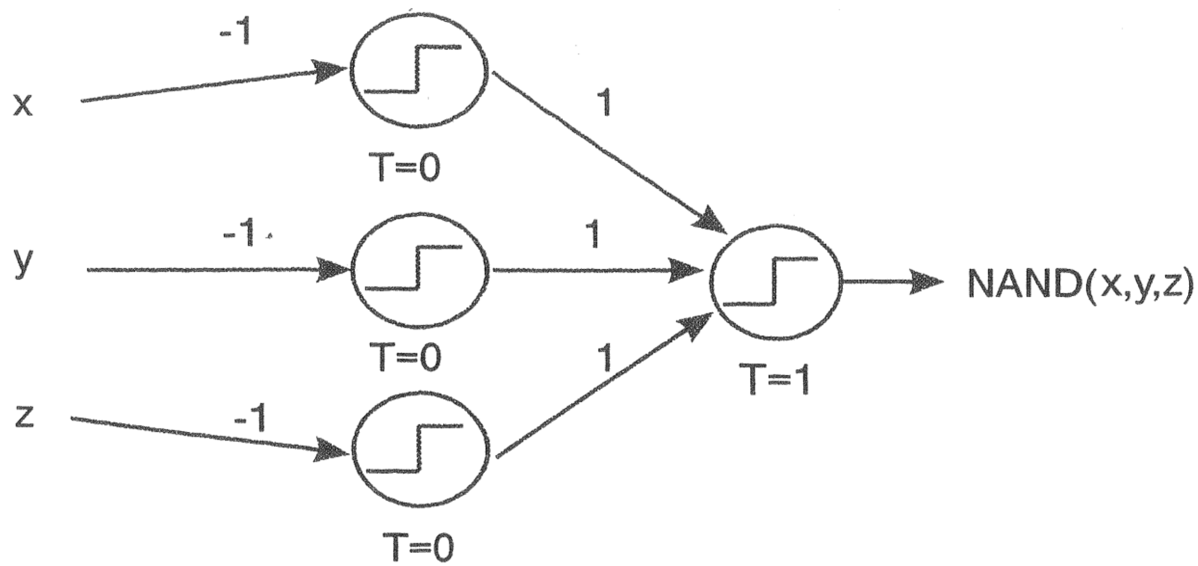


- Note that no **"learning"** is occurring here - It is simply implementing something it has been designed for.
  - i.e. by ***adapting the weights*** for each case, the network becomes dedicated to that task.

Logic Operators of the McCulloch-Pitts Model

- NOR(x, y, z) = NOT(x OR y OR z)
- NAND(x, y, z) = NOT(x AND y AND z)

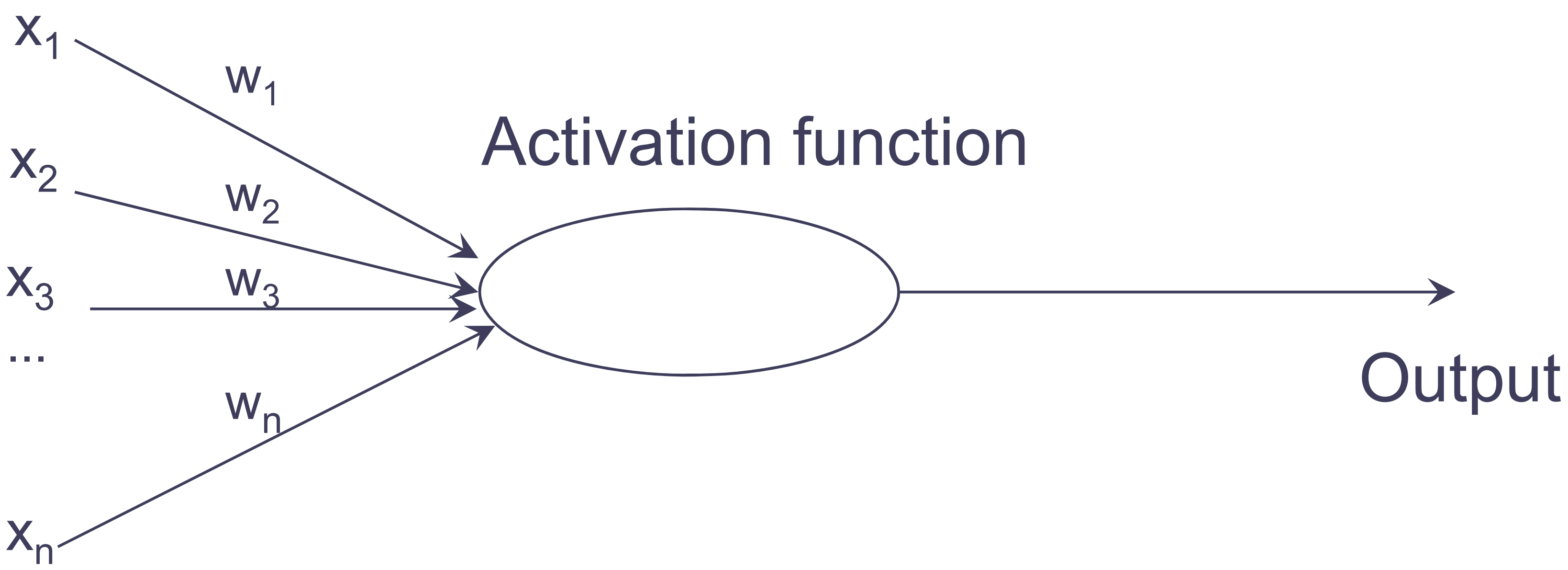




A More General Model

- Let's remove simplifications (limitations) of the McCulloch-Pitts model:
  - Allow weights to take on any values;
  - A more general "activation function" than a threshold function;
  - Allow continuous output of neuron;
  - Allow asynchronous firing of neurons (this depends on the architecture chosen);
  - Allow neurons to interact fully.

Detail of A General Model



Output: $o = f(\mathbf{w}^t \mathbf{x}) = f(\sum_{i=1}^{n} w_i x_i)$

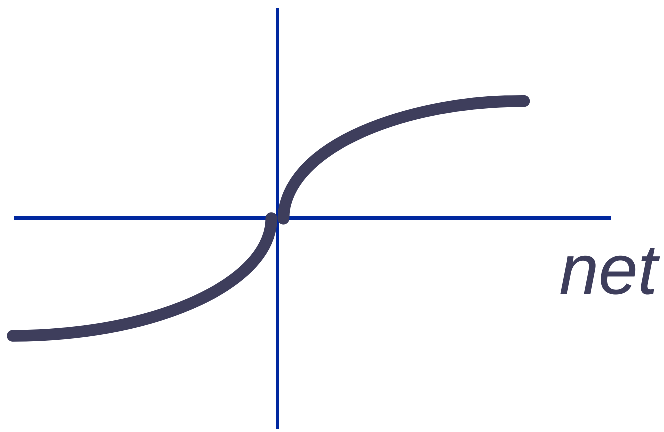
Detail of A General Model (continued)

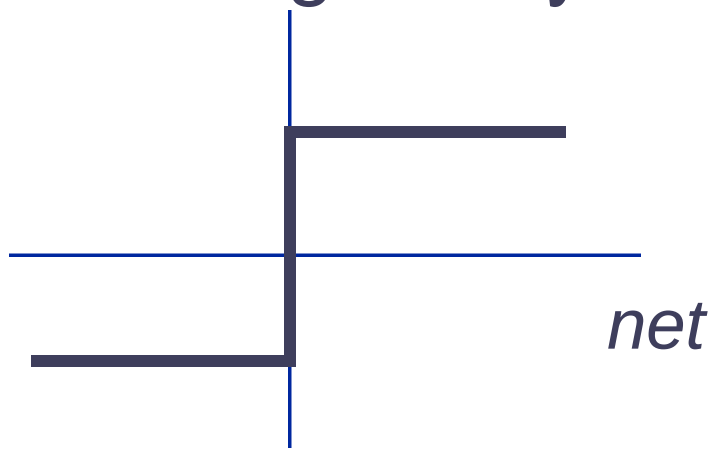
- Actual output depends on the nature of activation function f(.).

- Let the total input to a neuron be called ***net***

$$net = \mathbf{w}^t \mathbf{x} \quad or \quad net = \sum_{i=1}^{n} w_i x_i$$

- f(*net*) can be sigmoidal or thresholding or anything.

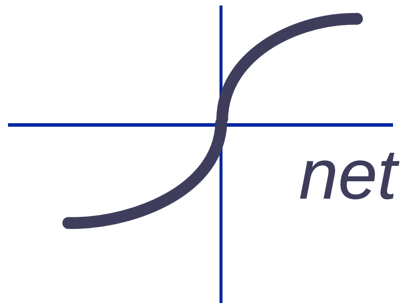




Detail of A General Model (continued)

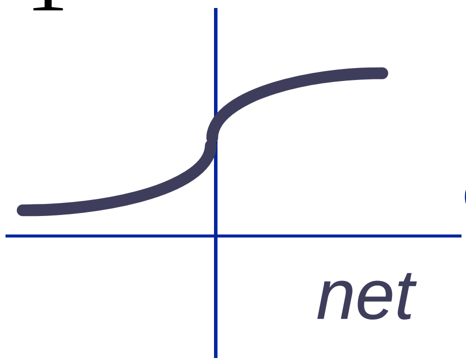
Assume threshold value of T = 0

$$f(net) = \frac{2}{1+\exp(-\lambda net)} - 1$$



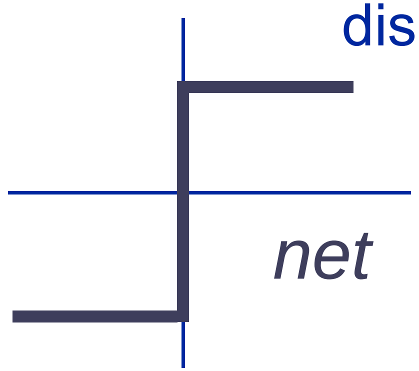
Bipolar continuous

$$f(net) = \frac{1}{1+\exp(-\lambda net)}$$



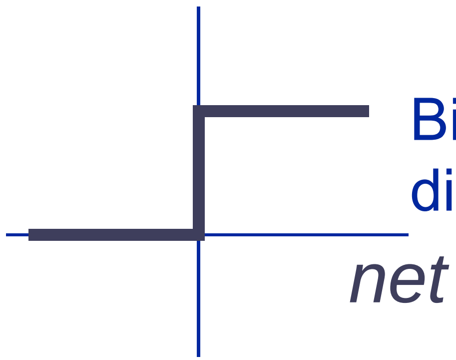
Binary continuous

$$f(net) = \begin{cases} +1, & net > 0 \\ -1, & net < 0 \end{cases}$$



Bipolar discrete

$$f(net) = \begin{cases} 1, & net > 0 \\ 0, & net < 0 \end{cases}$$



Binary discrete

where λ is a parameter which controls the steepness of the function.

Detail of A General Model (continued)

- So activation functions can be discrete or continuous, and bipolar (1, -1) or binary (0, 1).
- Not all problems can be modelled with T = 0, so we can add another neuron (n+1) with

$$x_{n+1} = -1 \text{ and } w_{n+1} = T$$

- *Example*: 2 input neurons with *x* values 1 and -2, weights 3 and -1, threshold T = 4, discrete binary activation function:

a) T = 4: $(1 \times 3) + (-2 \times -1) = 5$

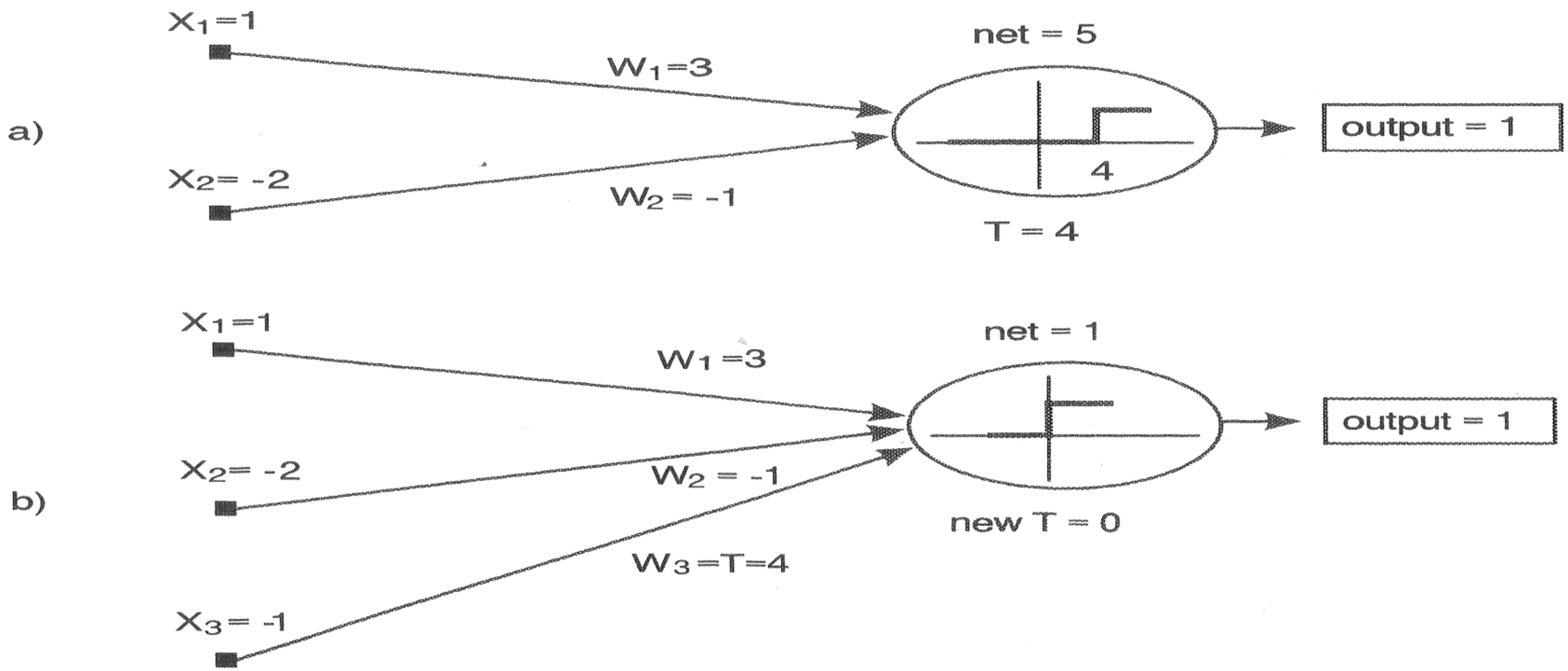
5 >= 4? Yes: output = 1

b) New T = 0: $(1 \times 3) + (-2 \times -1) + (-1 \times 4) = 1$

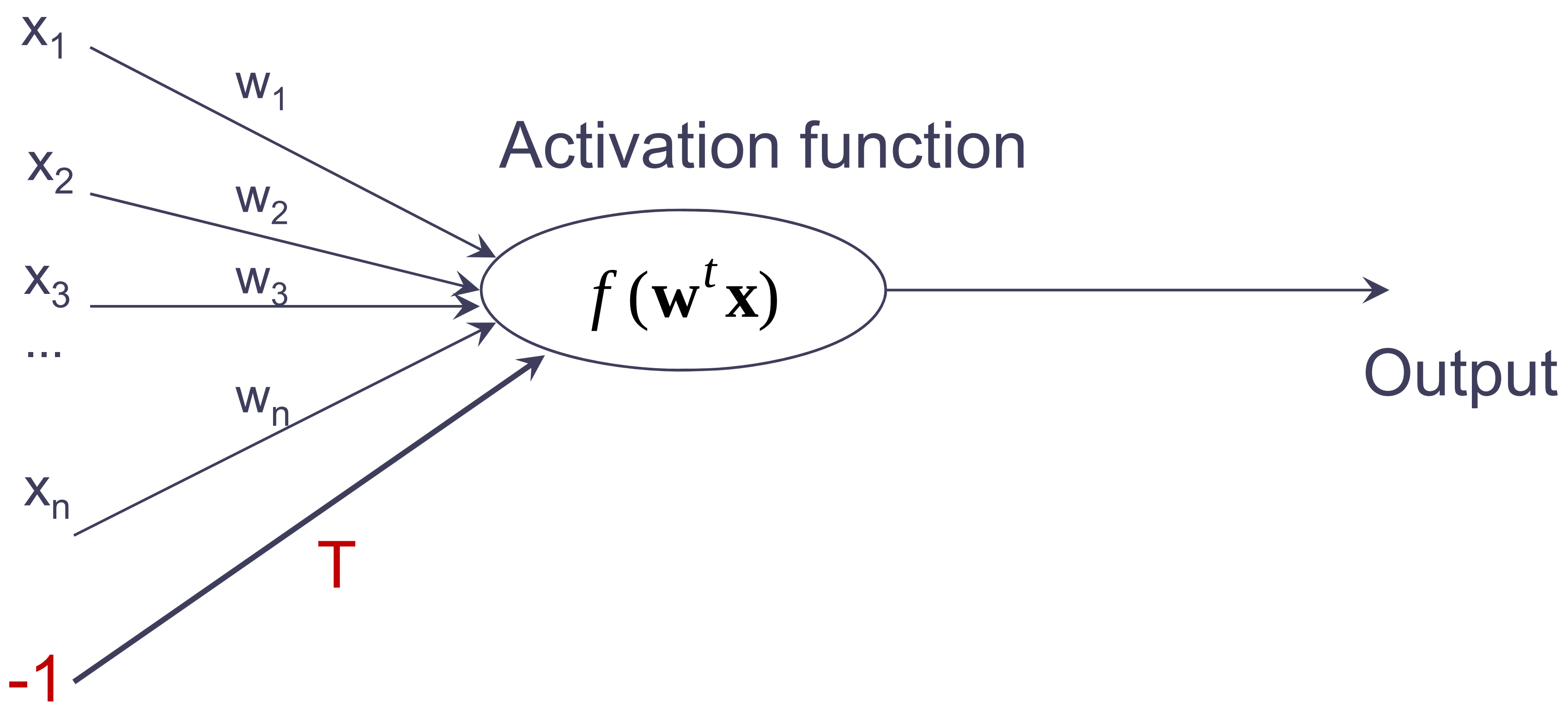
1 >= 0? Yes: output = 1

Detail of A General Model (continued)

- *Example*: 2 input neurons with *x* values 1 and -2, weights 3 and -1, threshold T = 4, discrete binary activation function:
  a) T = 4: $(1 \times 3) + (-2 \times -1) = 5$
     5 >= 4? Yes: output = 1
  b) New T = 0: $(1 \times 3) + (-2 \times -1) + (-1 \times 4) = 1$
     1 >= 0? Yes: output = 1



So the General Model can be like



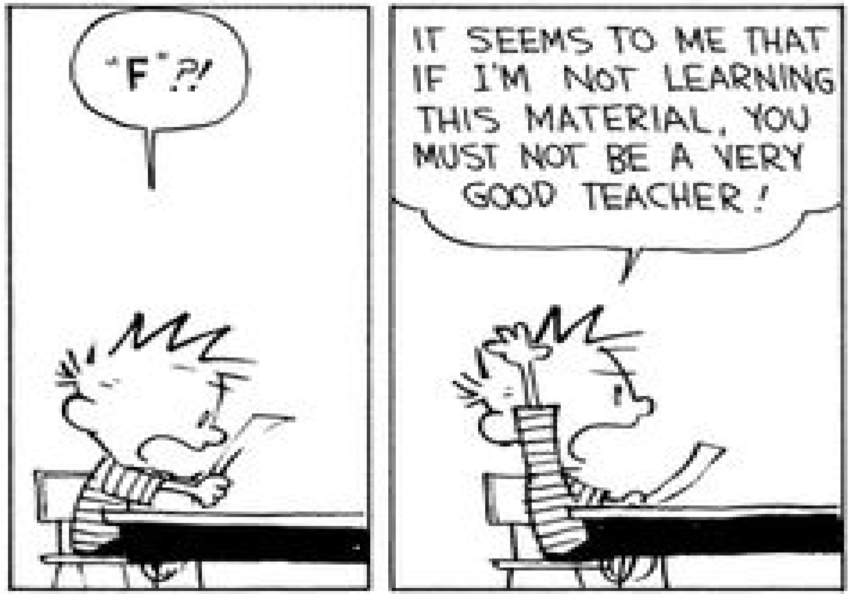
Learning

- Note that so far we have just considered networks with fixed weights.
  - We are simply calculating the output for a given input vector (i.e. OR, AND, NOR, NAND).
- Such networks can be used to classify or predict things, once the weights are known.
- But how do we work out what the weights should be to achieve a task?

Learning (continued)

- The good thing about neural networks is that they can **"learn"** relationships between inputs and outputs.
- Learning is shown by the weights adapting themselves to reflect some experience.
- There are basically two types of learning:
  - ***Supervised learning*** where we know what the output should be and force it through weights;
  - ***Unsupervised learning*** where the network finds patterns itself.
- Learning type used depends on the chosen architecture (which depends on the problem type).

Supervised Learning



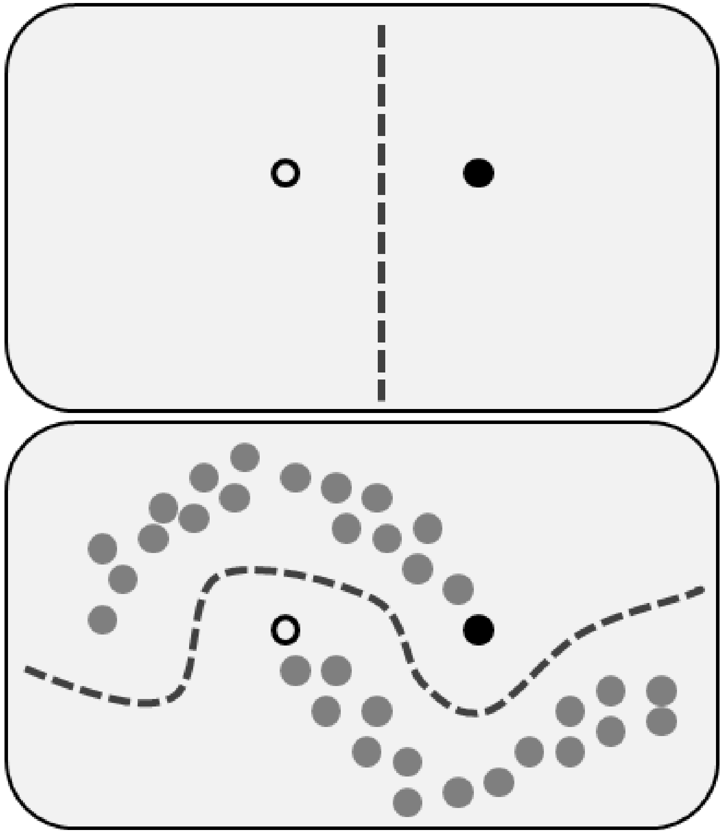
- Imagine the learning process in the classroom:
  - A teacher asks a question.
  - The student answers.
  - If the answer is correct, what the student believed true is now strengthened in her/his mind.
  - If the answer is incorrect, the teacher tells the student the correct answer, and the student has learnt.
  - It may take several repetitions for the student to permanently remember the correct answer.

Unsupervised Learning

- The student sorts through information herself/himself to discover relationships between the information.
  - The student learns that a relationship is true if she/he repeatedly encounters it.
- "Learning without a teacher".

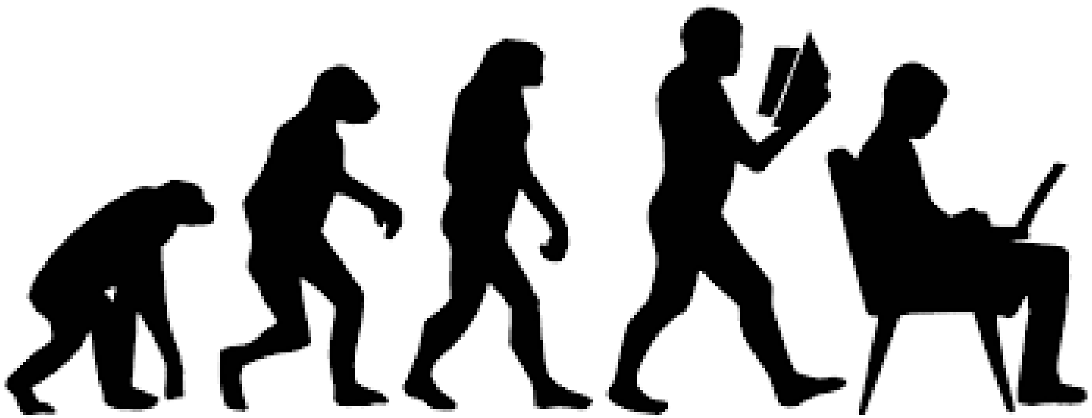
Semi-supervised Learning

- The teacher can only tell or know partially the correct answer.
- The student learns the correct answer gradually from the partial information.
- “Learning partially from a teacher”.
- A mix of supervised and unsupervised learning.



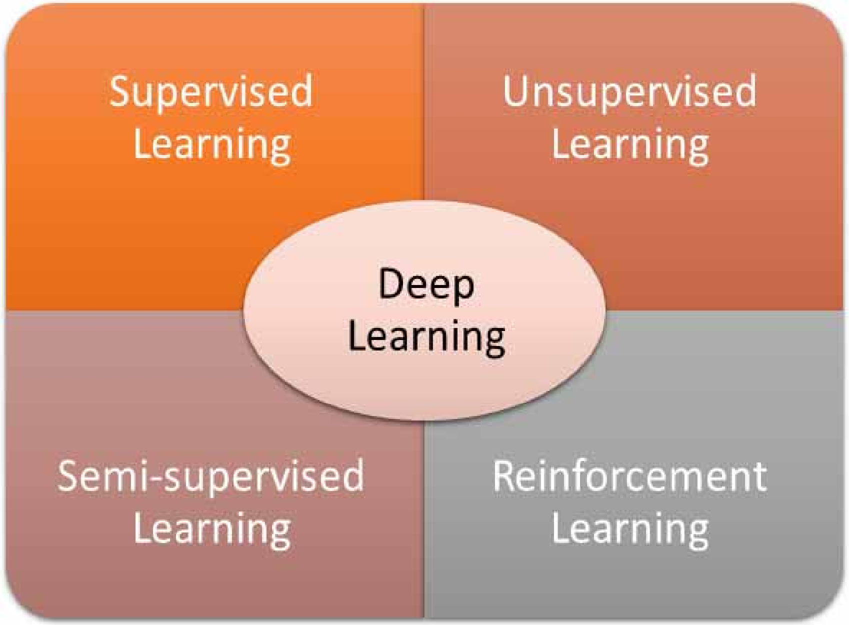
Reinforcement Learning

- The teacher does not tell the student the correct answer or there is no teacher available.
- The student works out the answer herself/himself.
- If the student’s answer is or is closer to the correct answer than her/his previous answer, get a reward, otherwise a penalty.
- The student learns to find the right answer through continuous adjustments.



Deep Learning

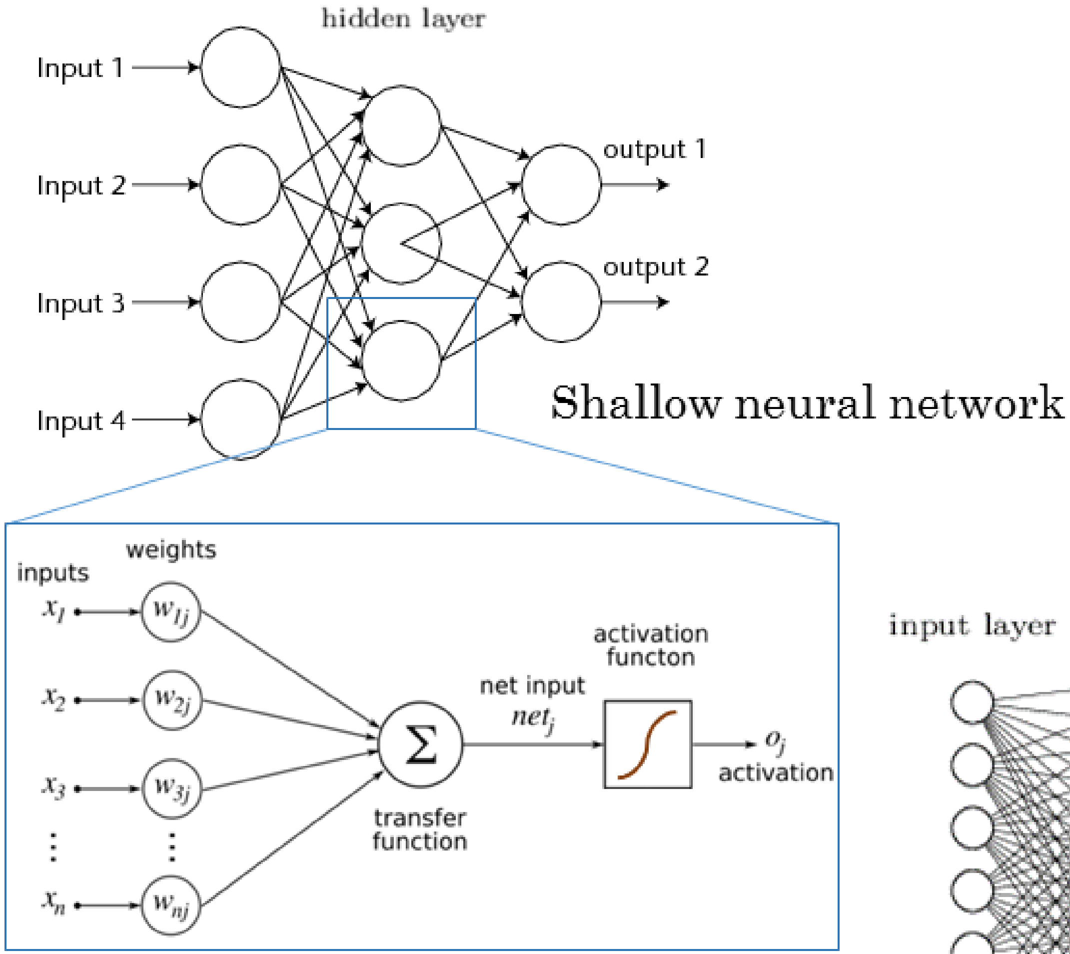
- A collection of statistical machine learning techniques.
- Used to learn feature hierarchies automatically.
- Usually based on artificial neural networks.

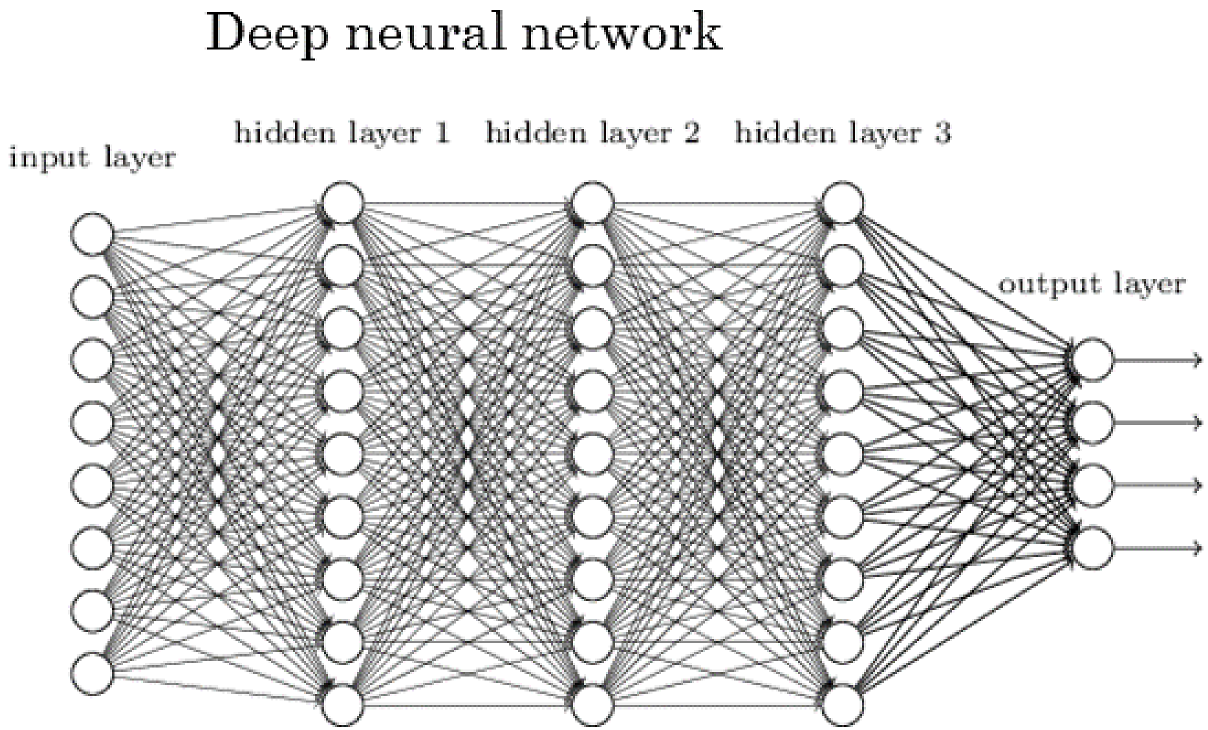


- Deep neural networks have more than one hidden layer (of data representation).
- A Primer on Deep Learning

  https://www.datarobot.com/blog/a-primer-on-deep-learning/

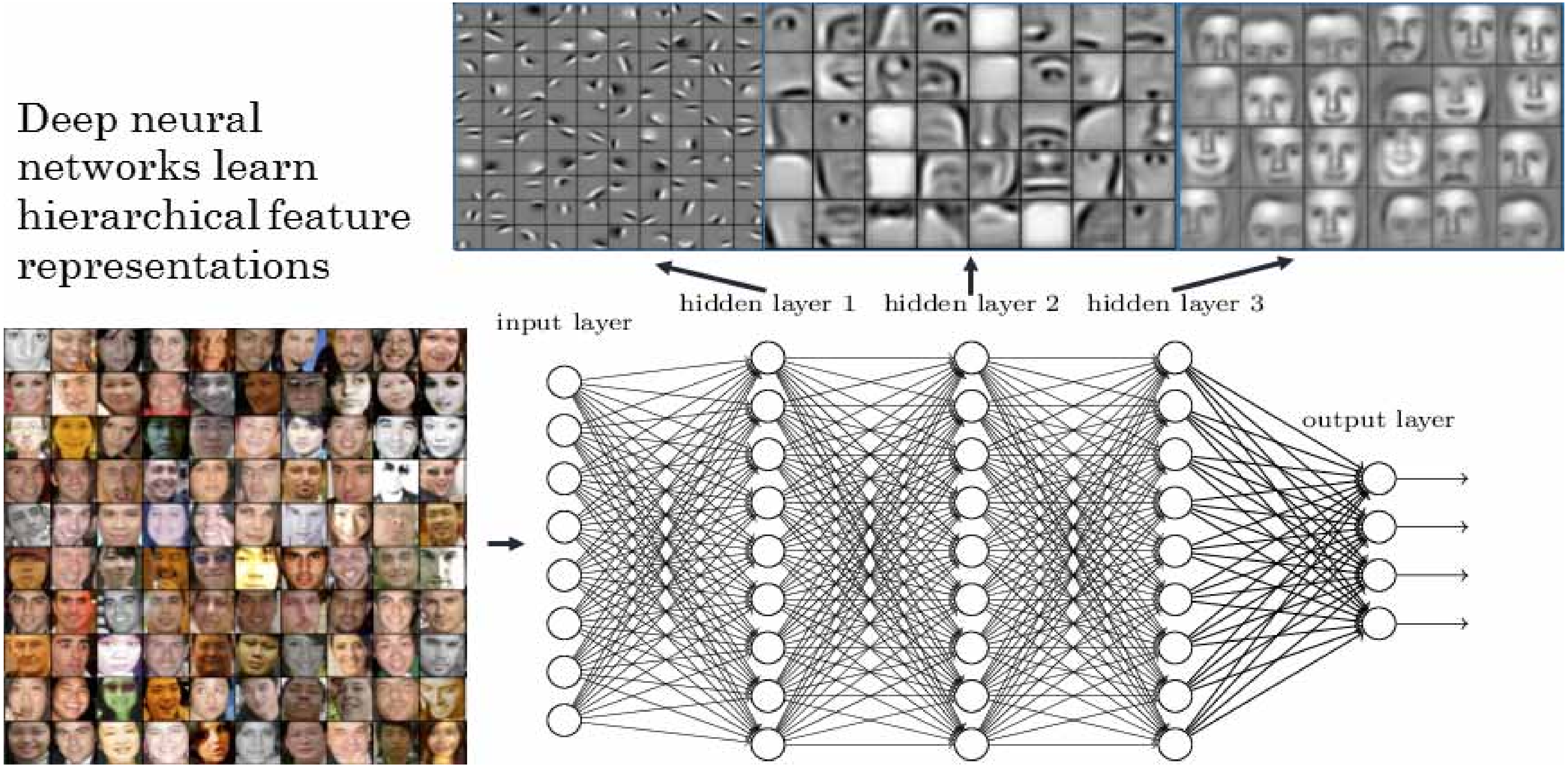
Deep Neural Networks (Continued)





- http://www.rsipvision.com/exploring-deep-learning/

Deep Neural Networks (Continued)

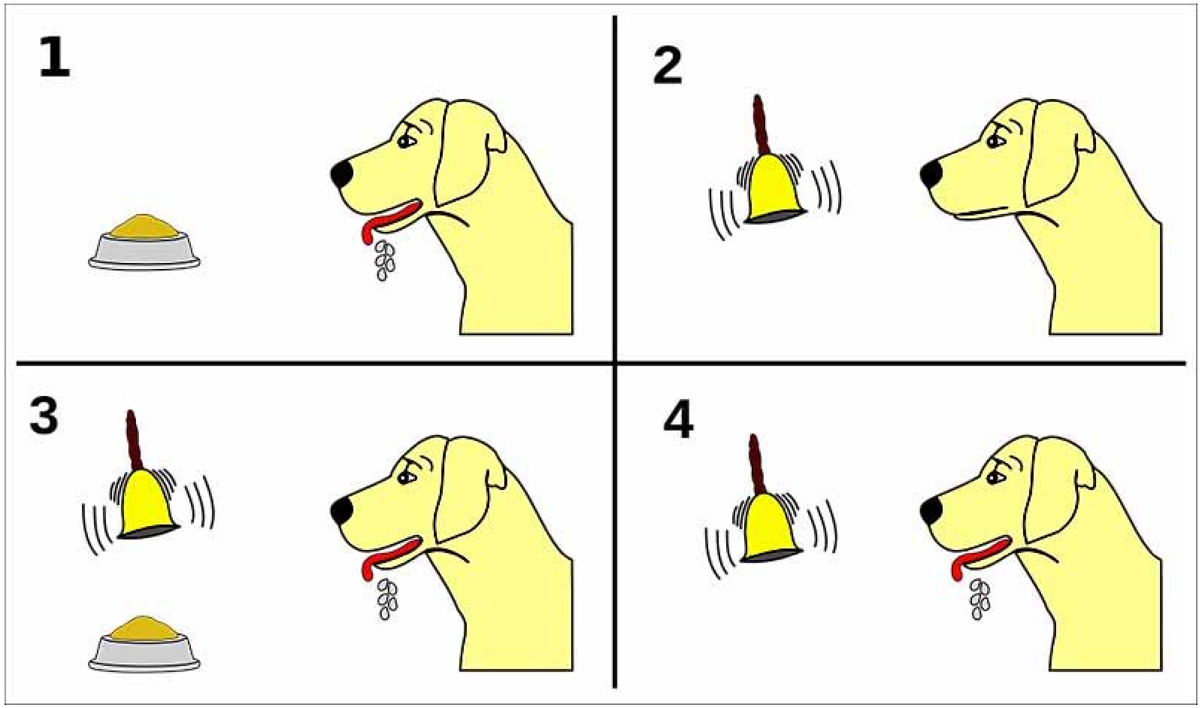


- "The takeaway is that deep learning excels in tasks where the basic unit, a single pixel, a single frequency, or a single word/character has little meaning in and of itself, but a combination of such units has a useful meaning." - Dallin Akagi

How Does Learning Work?

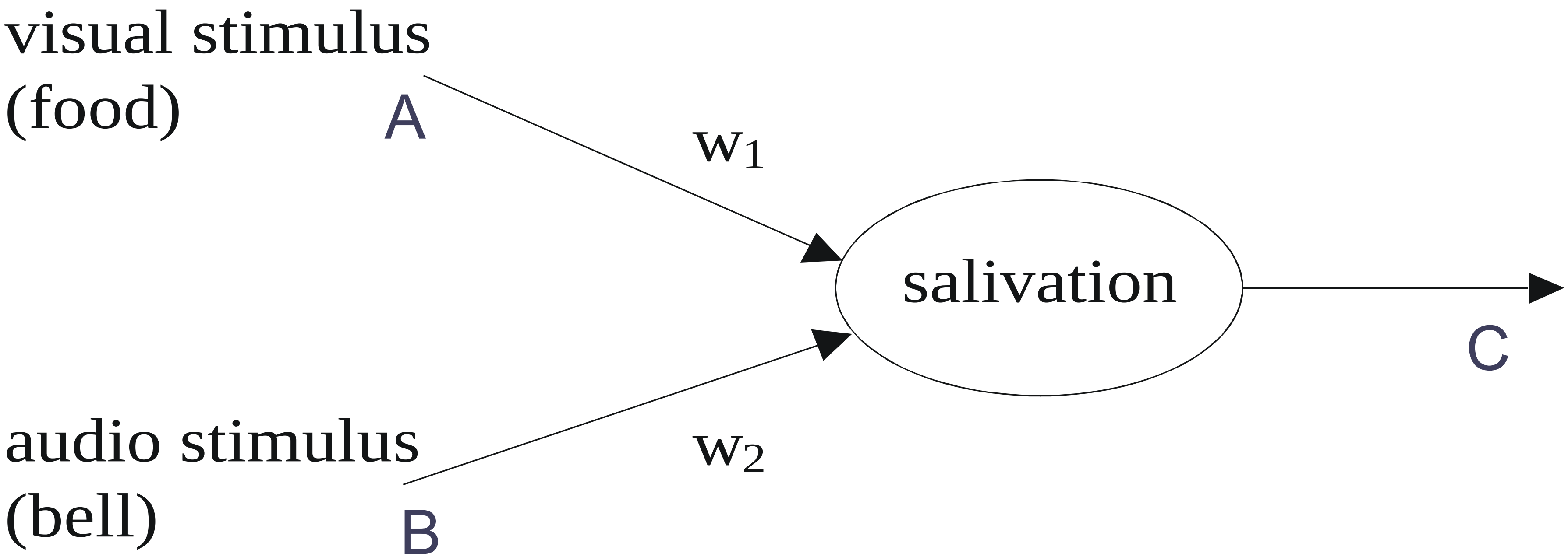
- The multiplicative weights in the neural network model are modified to reflect the learning process.

- Consider the Pavlov's dog example.



Pavlov's Dog Example

- Suppose excitation of A due to sight of food is sufficient to excite C causing salivation.
- Excitation of B due to hearing bell is not sufficient to cause firing of C (initially).
- Initially weights (w_1, w_2) might be (2, 0); ring bell with food (2, 0.9); keep doing this (2, 1.8). Now don't need food.



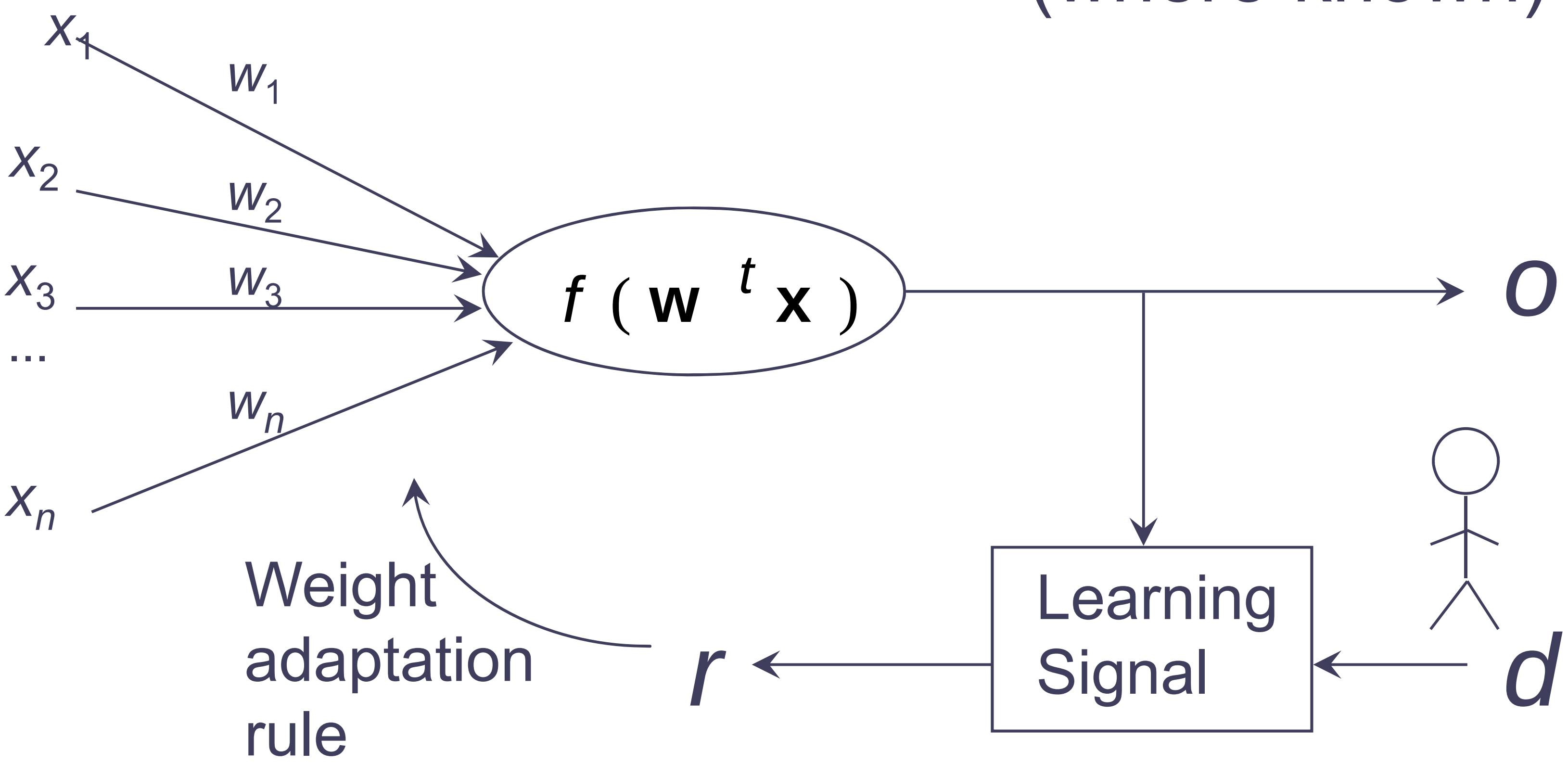
Pavlov’s Dog Example (continued)

- Show food (excite A), so C will fire.
- While C is still firing, ring bell to excite B.
- B is now participating in the excitation of C.
- B’s influence on C is now increased by increasing the weights connecting B and C.
- If this experiment is repeated often enough, eventually B will be able to cause C to fire, even in the absence of visual input from A!
- The network has learnt a relationship.

- How are the weights updated?
  - There are many different rules for learning ...

Process of Learning

o is the observed output
d is the desired output
(where known)



r is the learning signal (a function of x, w, and d).

Learning Rules

- The learning signal generated is a function of *x*, *w* and maybe *d* if it's known (supervised learning)
- It is used to modify the weights.
- What is this function?
  - It depends on the type of learning rule used.
- Generally, the weights are adapted like:

$w(t+1) = w(t) + c\,\mathbf{r}(w(t),x(t),d(t))\,x(t)$

c is the learning constant (determines the **rate of learning**).

$r(.)$ is the **learning signal** (determines the type of learning).

$x(t)$ is the current input.

$w(t)$ is the current weight.

Learning Rules (continued)

- In the following lectures, we will be looking at different neural network architectures, and learning rules.
  - For classification, prediction, clustering, and data mining;
  - Lots of case studies;
  - With hands on experience for you in the tutorials.

- We'll start by looking at how McCulloch-Pitts type neurons can learn (Perceptron model).

Perceptrons

- Devised by Frank Rosenblatt (psychologist/neurophysiologist) in late 1950's.
- An attempt to "illustrate some of the **fundamental properties of intelligent systems** in general, without becoming too deeply enmeshed in the special, and frequently unknown, conditions which hold for particular biological organisms"

(Rosenblatt's "The Perceptron: A probabilistic model for information storage and organization in the brain", 1958)

Perceptrons for Classification

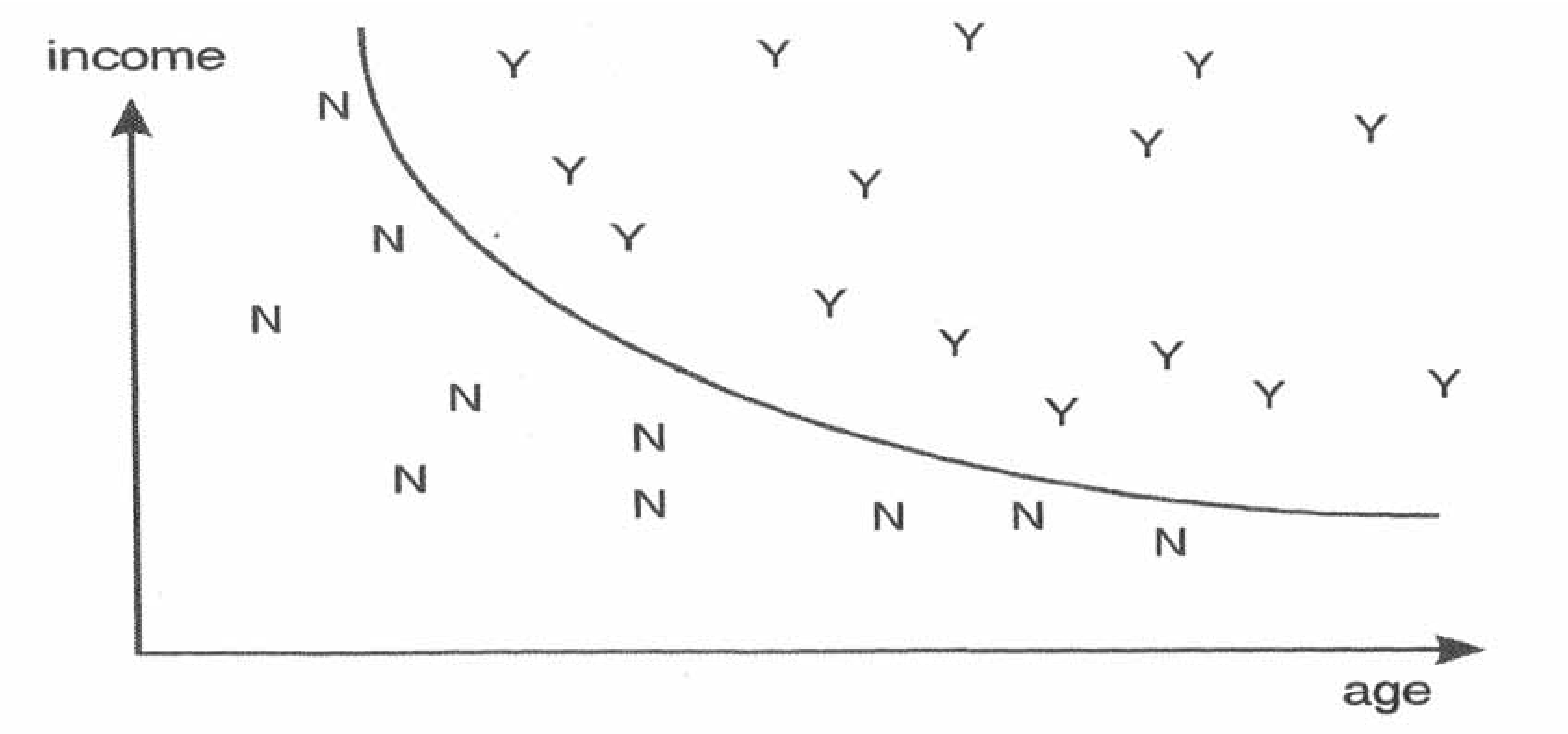
- Based on the McCulloch-Pitts model of neuron.
- Basic principle: through ***training***, Perceptrons could be used to classify inputs patterns (data) into categories.
- Classification is the search for common features within data.
  - Done for centuries: e.g. classification of living species, plants, weather conditions, etc.
  - More recently: fingerprint identification, radar and signal detection, printed and written character recognition, medical diagnosis, speech recognition, bank cheque processing.

Perceptrons for Classification (continued)

- The ability to classify is learnt over time as a result of repetitive inspecting and classifying examples.
- Inferences are made from experience.
  - e.g. We know there are categories for animals such as mammals, reptiles, birds, etc.
    - Each has distinctive features that uniquely categorise the class, i.e. blood temperature, body covering, etc.
    - Sometimes the decision will be obvious, other times it isn't (i.e. a cold-blooded animal with feathers).

Discriminant Functions

- Discriminant functions are lines (or curves) in the input space which separates regions of data according to their classification.
- The following figure shows an example of a discriminant function which separates 15 granted loan customers from 10 rejected customers based on only two inputs (age and income).



Perceptrons as Classifiers

- Consider an example to demonstrate how a Perceptron can learn to find a "**discriminant function**" or "**decision boundary**" based on some data examples.
- Example:

  Suppose there are 6 points in the 2 dimensional (2D) input space, separated into 2 distinct classes

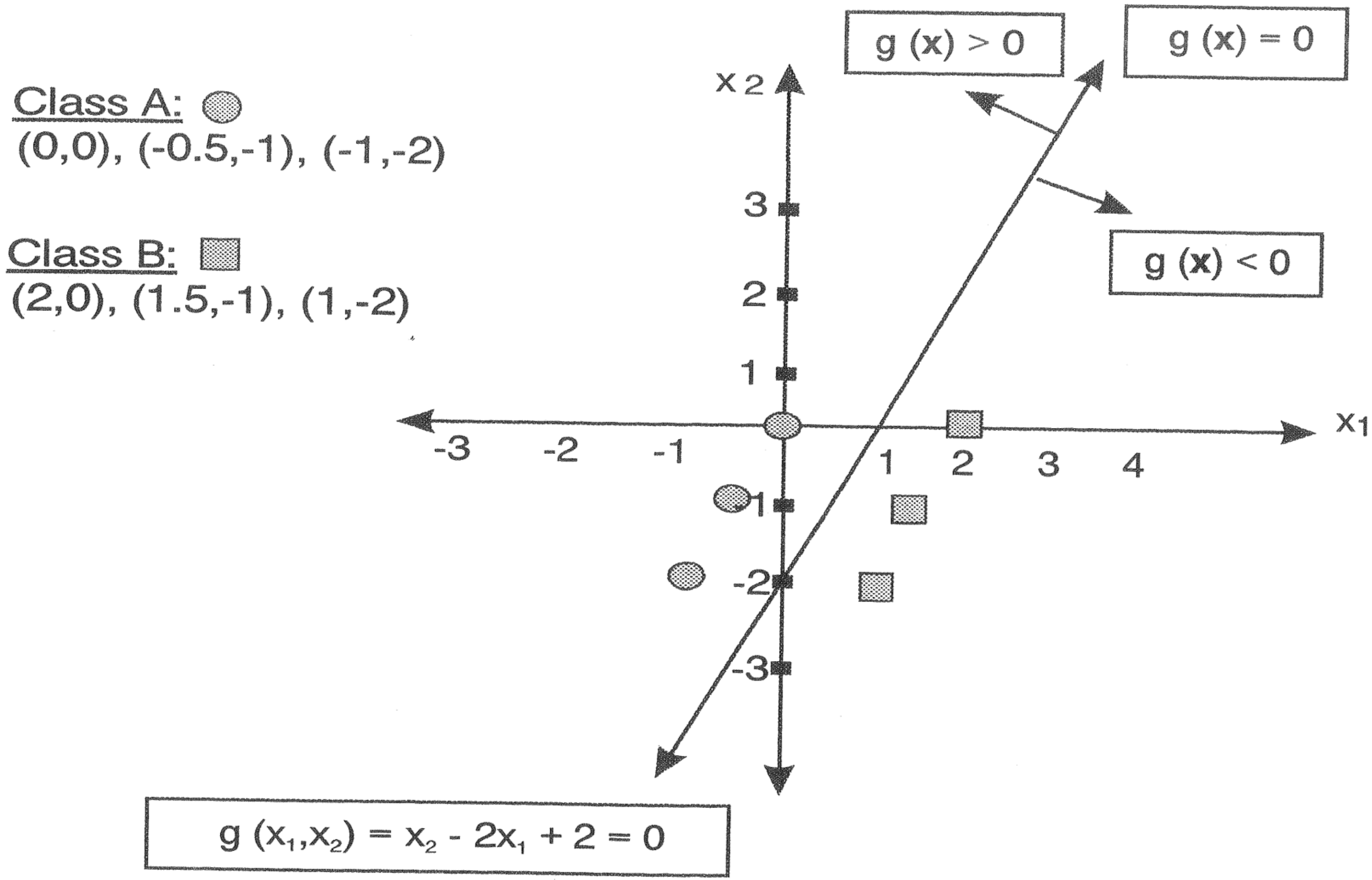
  (0,0), (-0.5, -1) and (-1, -2) are in CLASS A.

  (2,0), (1.5, -1) and (1, -2) are in CLASS B.
- Can we draw a line in the input space to separate the A's from the B's?
  - Answer: Yes! (There are infinitely many such lines)

  $g(x_1, x_2) = -2x_1 + x_2 + 2 = 0$ is one line (see the next slide)

A Discriminant Function (Dichotomiser) for a Small Classification Example

- The following figure shows the six points and one such discriminant function $g\ (x_1,x_2)$.



Perceptrons as Classifiers (continued)

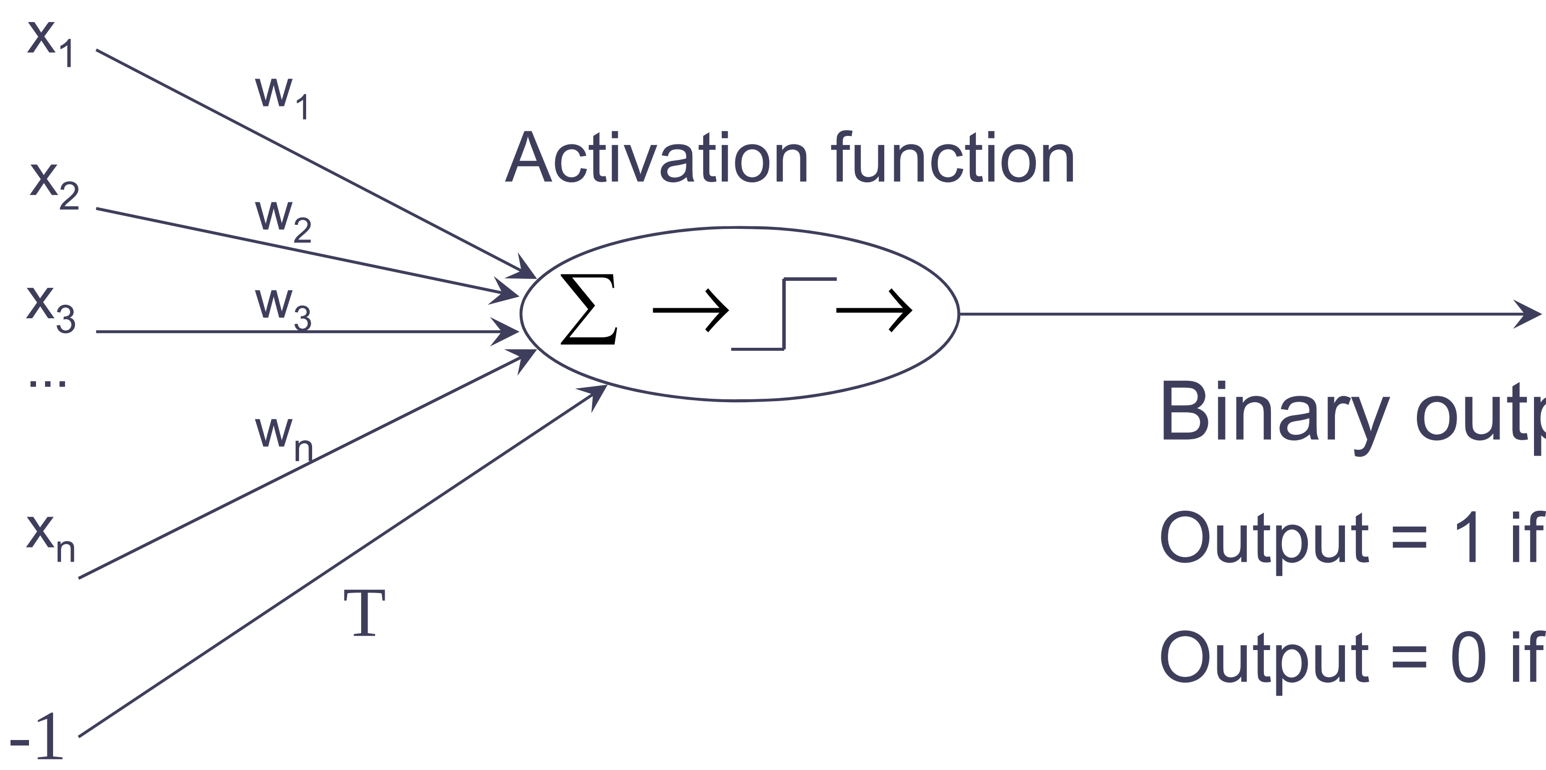
- The example demonstrates a discriminant function as a *dichotomiser*

  (a separation of the input space into 2 regions):

  – From Ancient Greek: *dicha* meaning “in two”
    tomia meaning “cut”

- How can we get a Perceptron to find a line which will dichotomise the input space based on some data?

Perceptron Model



Binary output

Output = 1 if *net* ≥ 0

Output = 0 if *net* < 0

Weights of the Perceptron

- The **weights** of the Perceptron are what determine the equation of the discriminant function or decision boundary:
- In fact:

$$w_1 x_1 + w_2 x_2 - T = 0$$

is the equation (remember T is the threshold value).

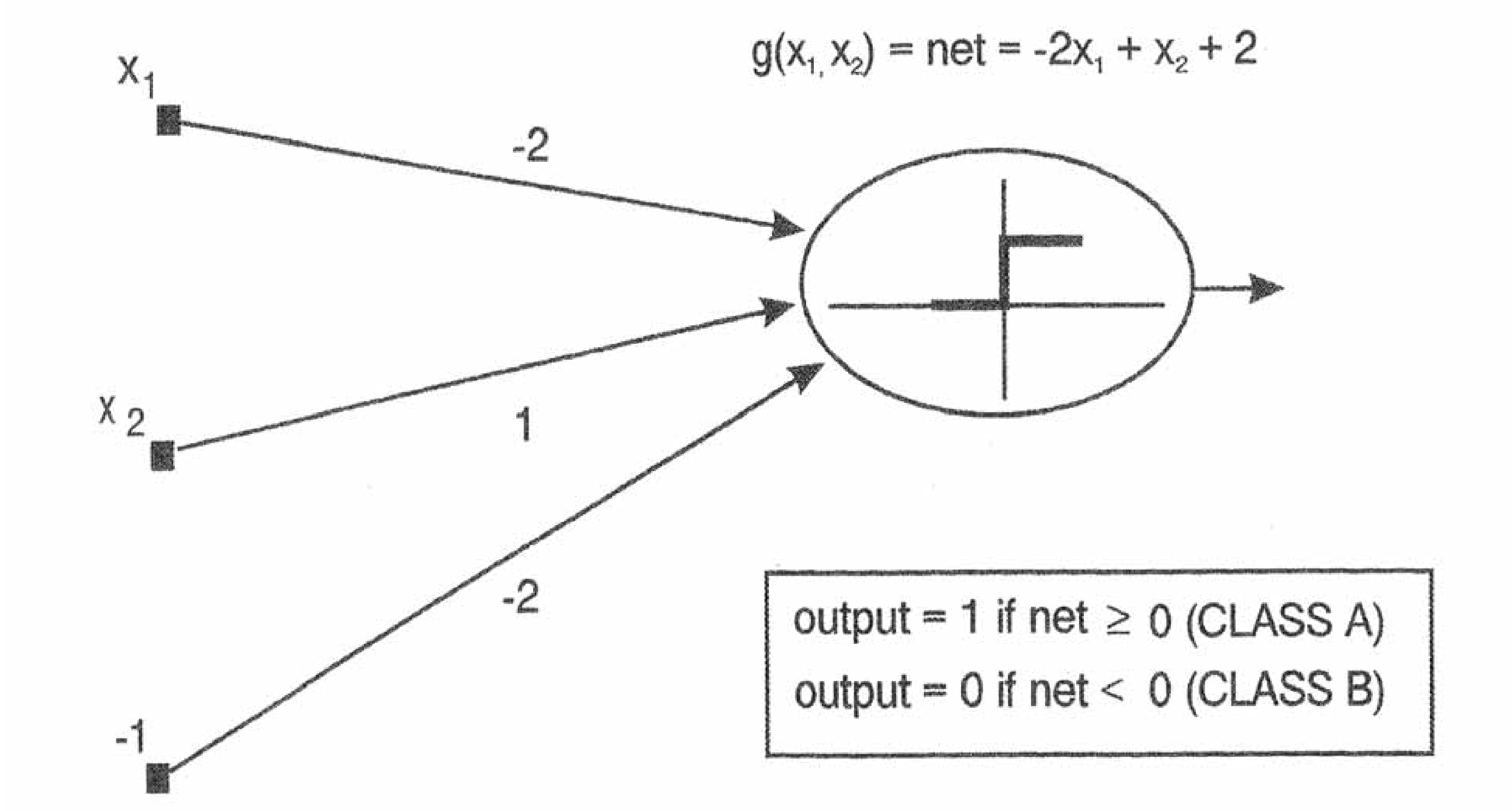
- So for the example, g $(x_1, x_2) = -2x_1 + x_2 + 2 = 0$, the Perceptron has weights

$$w_1 = -2,\ w_2 = 1, \text{ and } T = -2$$

  – Use this to check the classification for other inputs.

A Perceptron for a Small Classification Example

- The following figure shows the Perceptron.



Weights of the Perceptron (continued)

- So the weights of the Perceptron represent the equation of the discriminant function or decision boundary.
- How does the Perceptron arrive at those weights?
  - Initially the weights are all zero (or random).
  - The Perceptron is ***trained*** using the inputs and the known classifications (this is supervised learning).
  - Training modifies the weights (learning).
  - The inputs are repeatedly presented until the weights stop changing (i.e. learning is finished).

Perceptron Learning Algorithm

- Step 1: Initialise all weights (including the threshold weight) to be zero (or small random values): call the weight vector w^0.
- Step 2: Present an input $(x_1, x_2, ..., x_n, -1)$.
- Step 3: Calculate the output where f(.) is the discrete binary activation function $$o = f\left(\sum_{i=1}^{n+1} w_i x_i\right)$$
- Step 4: Adapt the weights according to:

$$\mathbf{w} \leftarrow \mathbf{w} + c(d - o)\mathbf{x}$$

where c is a learning constant, and d is the desired output for that input.

Next input (until all inputs have been presented)

Perceptron Learning Algorithm (continued)

- The desired output for an input is
  1 if the input is in class A and
  0 if the input is in class B
- Two classes only at this stage.
- Weights are only changed if the output is incorrect (if $o = d$, no change).
- Weights along which there is zero input are not changed either (since they are not contributing to the incorrect output).
- Learning is proportional to the size of error.
- Learning rate is controlled by parameter c.

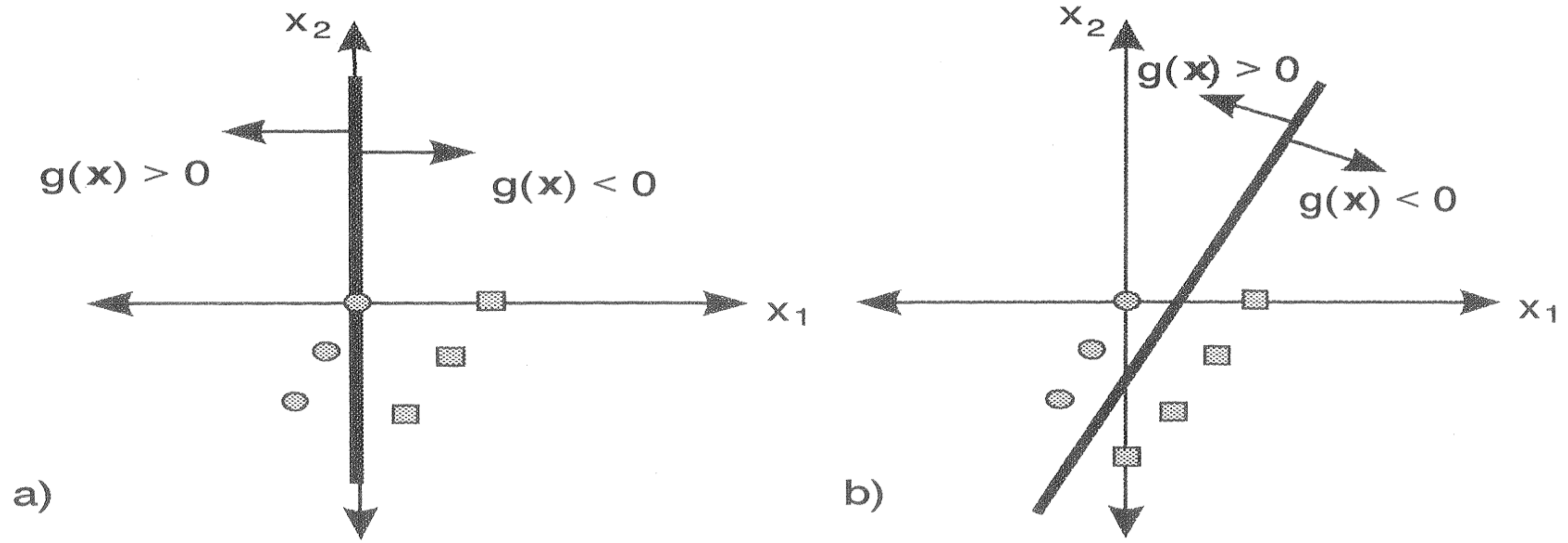
Perceptron Learning Algorithm (continued)

- See **additional material for Lecture 2 (pp. 1-2)** for an example of learning separation line for this example.
- Quite tedious by hand calculations
  - dichotomiser.exe (to be used in Tutorial 3).
- Different learning rates (and order of inputs, and initial weights) may result in different final weights (and hence equations for the decision boundary).
- x_1=0 (i.e. x_2 axis) separates the data OK.
- Add another point to Class B: (0,-3)

  What happens now?

Perceptron Learning Algorithm (continued)

- The line is forced to move, shown as follows:



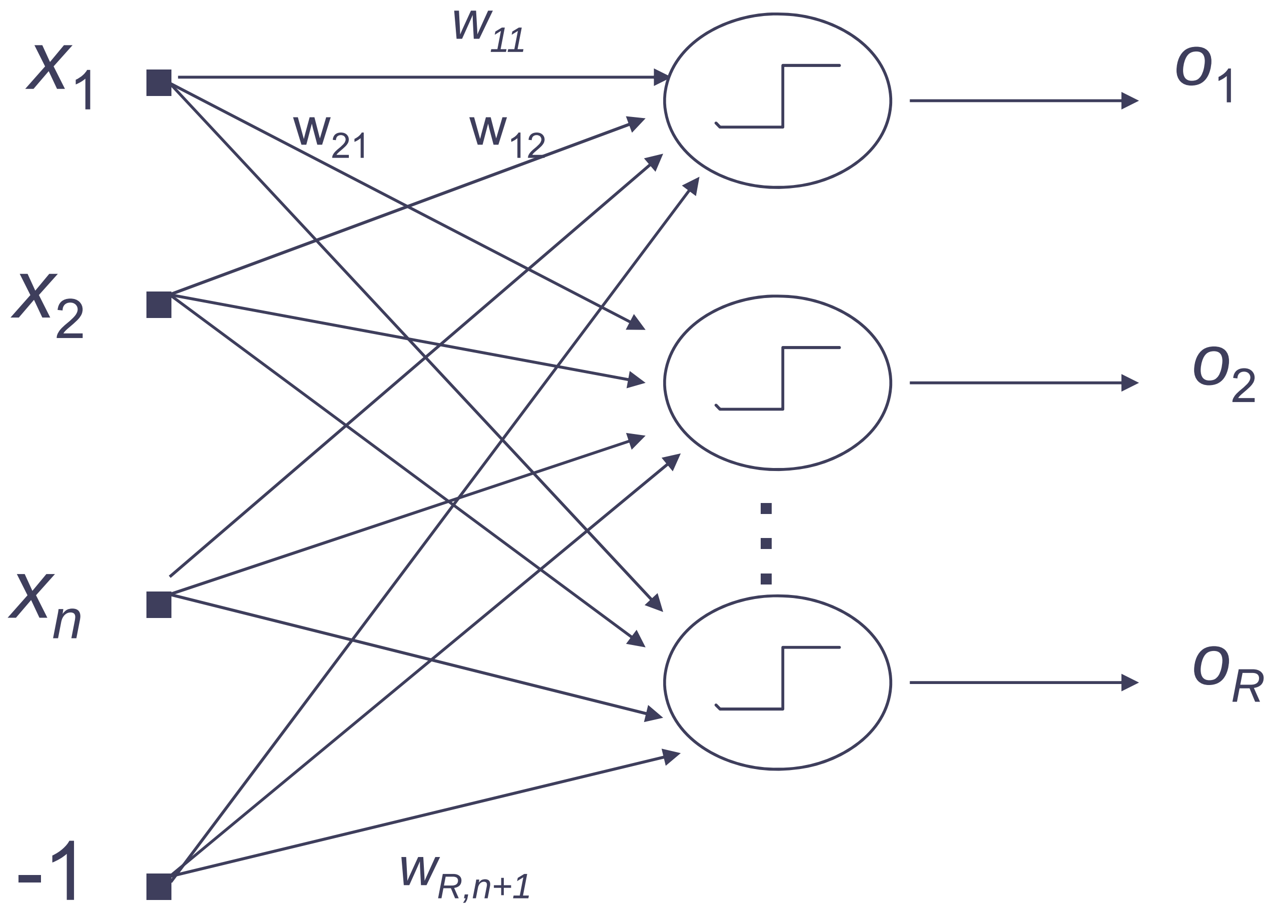
- Now that the classification has been learnt, so we can present new data to the network and it will decide its class based on this line.
- So far we have only examined the case where there are two classes (dichotomiser: output is 1 or 0 representing class A or B respectively).
- What happens if there are several classes in the data?

Multicategory Single Layer Discrete Perceptrons

- Suppose we have R classes in our training data.
- The classifier now consists of R discrete Perceptrons connected to the inputs.
- The output of each Perceptron is still 0 or 1;

  but for a given input, only one of the R outputs will be equal to 1, and the rest will be zero.
- The Perceptron whose output is 1 gives us the classification.

R-category Classifier



Only one output will be 1, the rest will be 0.

If o_j=1, then the input belongs to category *j*.

Learning Algorithm of R-category Classifier

- The learning algorithm is much the same as the dichotomiser algorithm:
  - Now using a weight matrix rather than a vector.
  - The rows of the weight matrix represent the weights for each Perceptron.
  - The desired output is now no longer a number, but a vector.
    - If the current input is in category 3 (out of 4 categories), the desired output is (0, 0, 1, 0).
  - The actual output is now also a vector (each number corresponding to a Perceptron).

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + c(d_i - o_i)\mathbf{x}$$ for each Perceptron *i*.

R-category Perceptrons

- Recall that a single Perceptron results in a single straight line to separate the data.
- R Perceptrons now result in R straight lines to separate the data.
- Example: suppose there are 3 points in the 2D plane, each from 3 different classes:

class 1: (10, 2)   class 2: (2, -5)   class 3: (-5, 5)

(see additional material for Lecture 2, pp. 3-5)

If initially the weights are →

$$\mathbf{W} = \begin{bmatrix} 1 & -2 & 0 \\ 0 & -1 & 2 \\ 1 & 3 & -1 \end{bmatrix}$$

What are the final weights?

What does the input space looks like after classification?

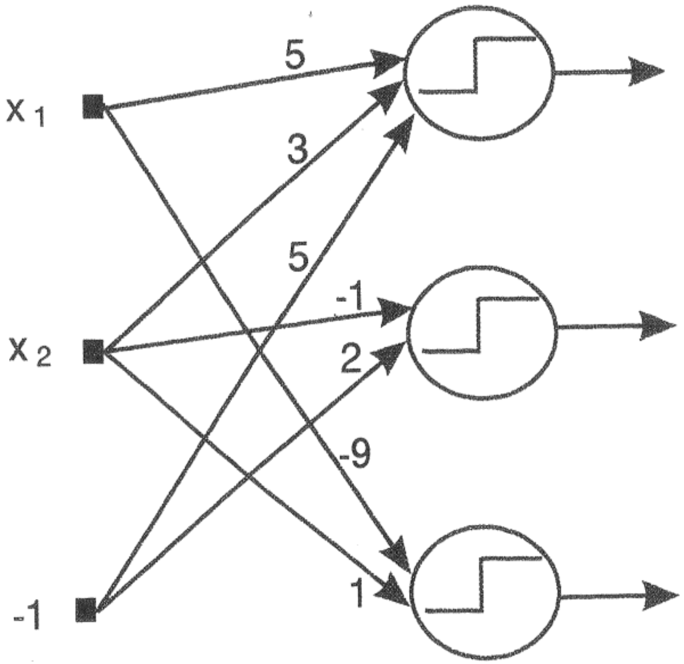
(see the next slide for indecision regions)

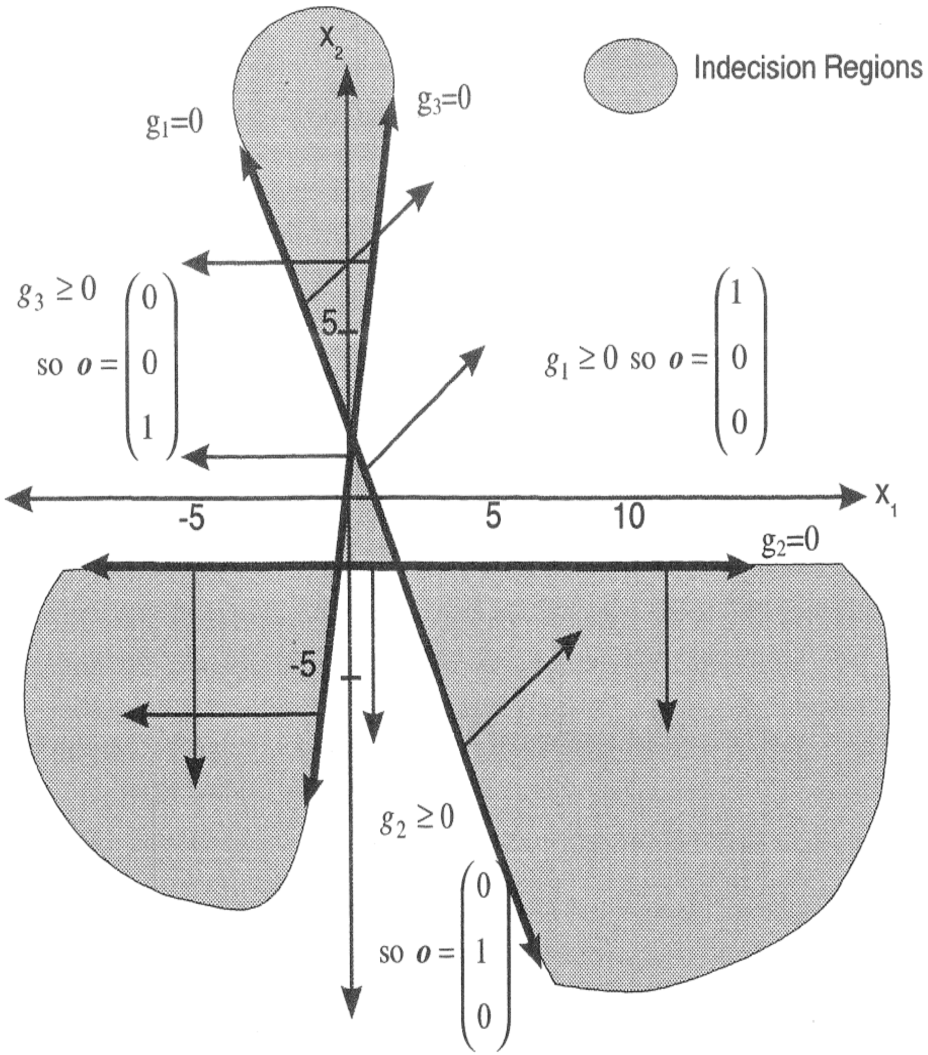
R-category Perceptron - Indecision Regions

$$\mathbf{W} = \begin{pmatrix} 5 & 3 & 5 \\ 0 & -1 & 2 \\ -9 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{w}_1^3 \\ \mathbf{w}_2^3 \\ \mathbf{w}_3^3 \end{pmatrix}$$

Where a new input gives $g_1 \geq 0$, $g_2 < 0$, $g_3 < 0$ then it belongs to CLASS 1.
Where a new input gives $g_1 < 0$, $g_2 \geq 0$, $g_3 < 0$ then it belongs to CLASS 2.
Where a new input gives $g_1 < 0$, $g_2 < 0$, $g_3 \geq 0$ then it belongs to CLASS 3.

'Indecision regions' form when more than one neuron outputs '1', or no neurons output '1': ie. $g_1 > 0$, $g_2 > 0$, $g_3 < 0$ (neurons 1 and 2 output '1') or $g_1 < 0$, $g_2 < 0$, $g_3 < 0$ (no neurons output '1').





Equations of boundary lines :

CLASS 1 : $5x_1 + 3x_2 - 5 = 0 = g_1$

CLASS 2 : $-x_2 - 2 = 0 = g_2$

CLASS 3 : $-9x_1 + x_2 = 0 = g_3$

Limitations of Perceptrons

- So single layer Perceptrons can be used to train and classify data into classes.
- Perceptron Convergence Theorem:

  If the classification *can* be learned by the Perceptron, then the procedure guarantees that it *will* be learned in a finite number of training cycles.

- How do we know if it *can* be learned?

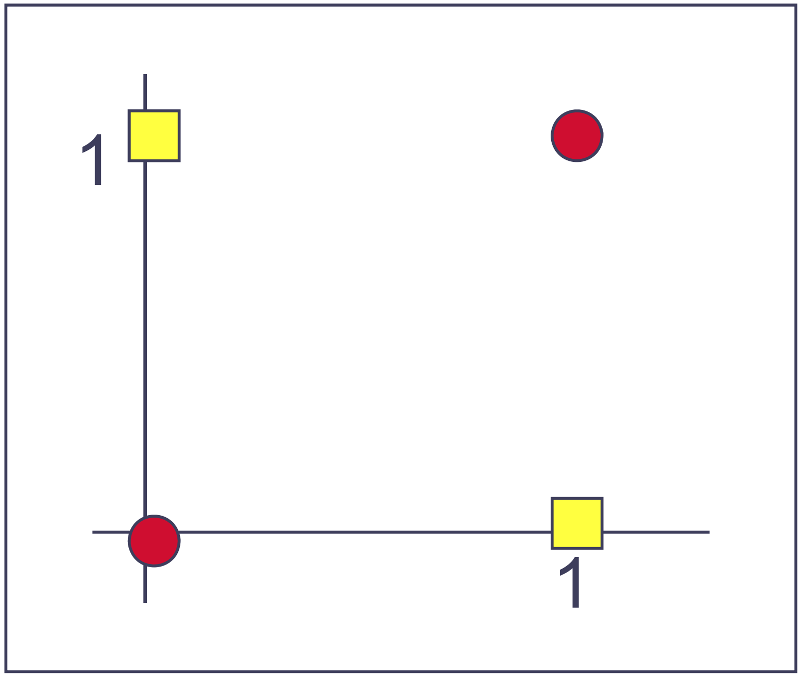
Limitations of Perceptrons (continued)

- In 1969, Minsky and Papert published *Perceptrons* which analysed the true capabilities and limitations of Perceptrons.
- They discovered that there are certain restrictions on the types of problems for which the Perceptron is suitable.
- Perceptrons can differentiate classes from data only if the data is ***linearly separable***.
- So far all the examples we have considered have been linearly separable (the data can be separated by drawing only straight lines).

The XOR Problem

- The XOR (exclusive OR) problem is an example where the data is not linearly separable.

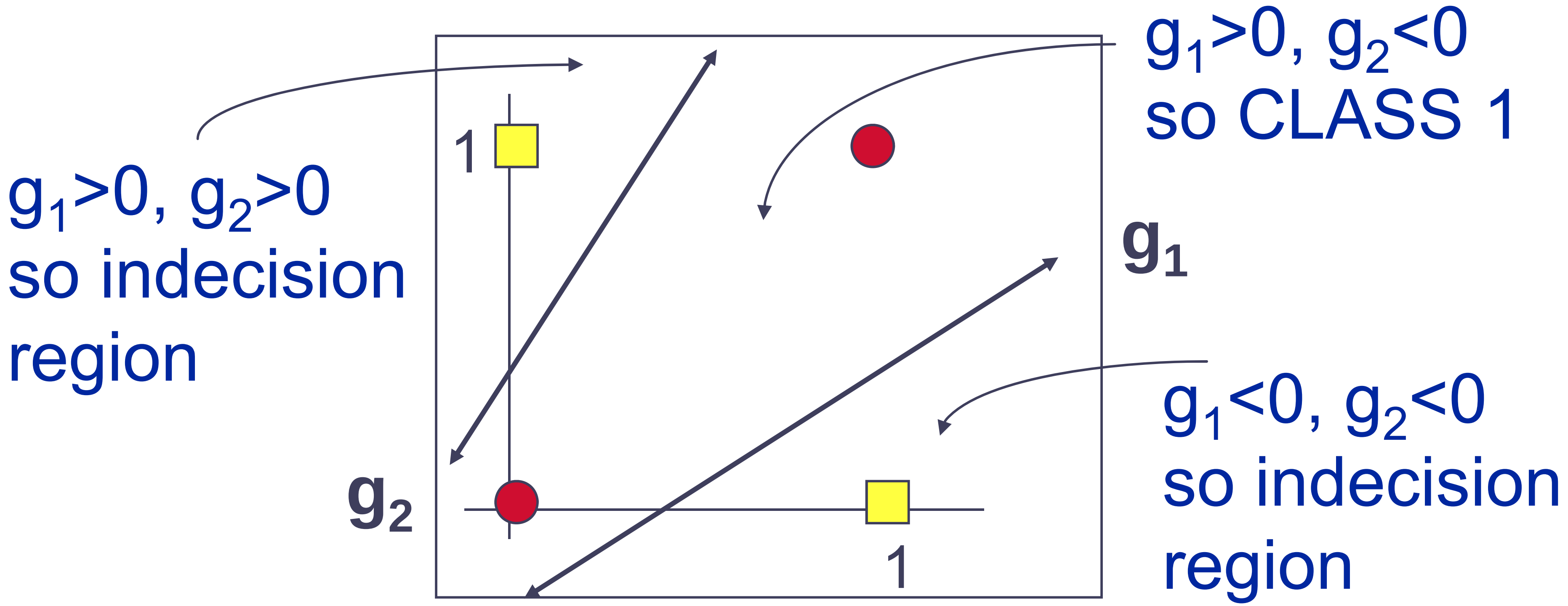
$x1$	$x2$	$x1 \oplus x2$
0	0	0
0	1	1
1	0	1
1	1	0



● Output 0 ■ Output 1

The XOR Problem (continued)

- It is impossible to find a straight line which dichotomises the two classes.
- How about using two straight lines (i.e. using 2 Perceptrons to classify the 2 classes?)
  - This won’t work either since the indecision regions will be too large.



The XOR Problem (continued)

- What you have learnt in this lecture will not work for the XOR problem (or any other linearly non-separable problem).
- The solution to this problem is to use Multilayered Perceptrons rather than Single-layered Perceptrons.
- We will look at these in the next lecture.

This Week's Tutorial

Tutorial 1

- Internet search for neural networks and data mining:
  - Tutorial sheet provides sites to look at.
- Appreciate the range of applications, and the excitement about the topic.
- These web sites will be a useful source of information for the unit.

This Week's Tutorial (continued)

Tutorial 2

- Character recognition using Perceptrons
  - three characters: C, L and I.
- Train the Perceptron to recognise these three characters using the Perceptron Learning Algorithm
  - by hand at first (to make sure you understand)
  - then use the "rclass-classifer.exe" program to produce final weights
  - Please read "Note for using rclass-classifier.pdf".
- Evaluate the trained Perceptron on corrupt characters ... How does it do? Is it robust?