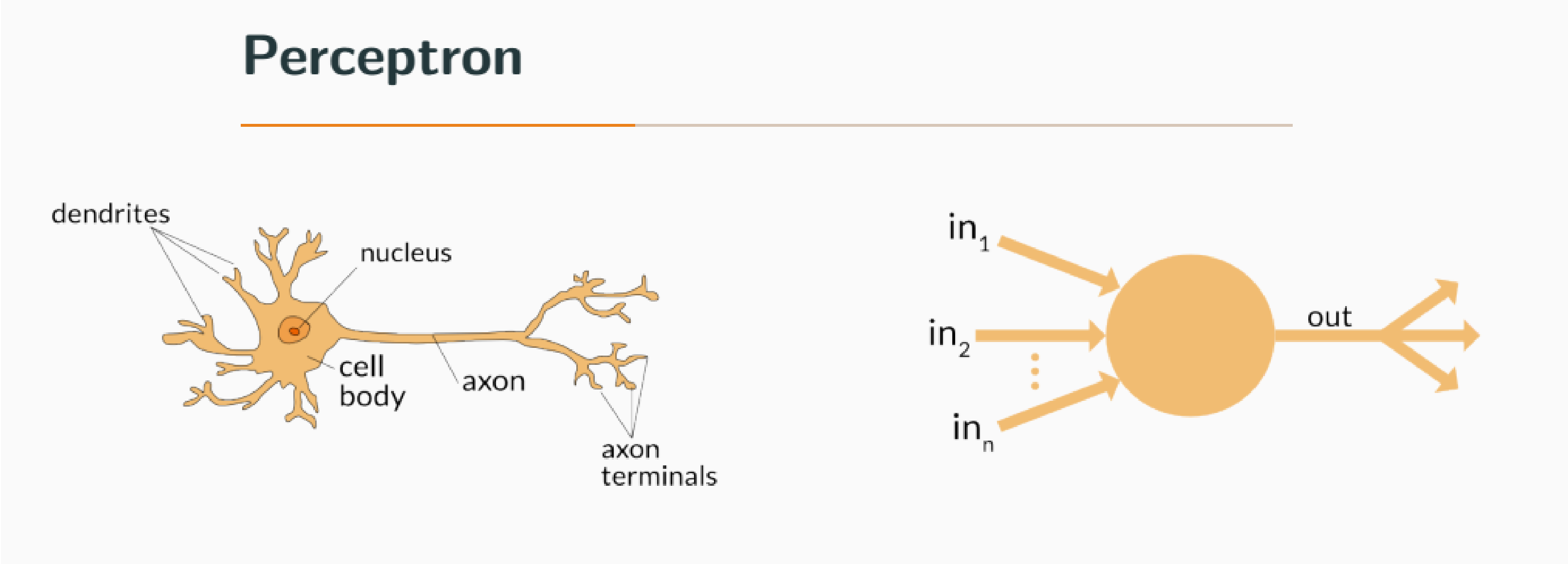
COMP417 Lecture 2

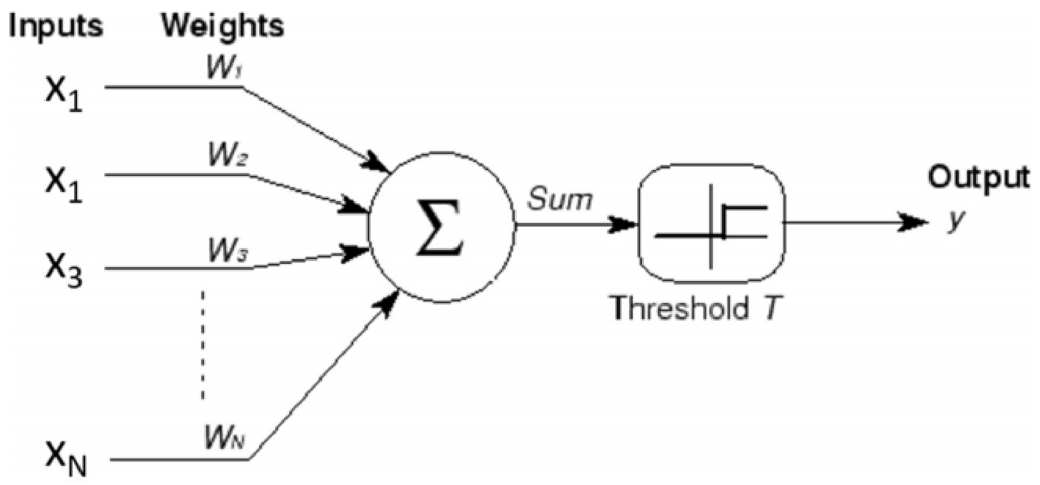
Artificial Neural Networks - The Basics

Dr. Hend Dawood

Neural Networks: The Perceptron



Perceptron: Simplified model



- Number of inputs combine linearly
  - Threshold logic: Fire if combined input exceeds threshold

$$Y = \begin{cases} 1 \;\; if \; \sum_i w_i x_i - T \geq 0 \\ 0 \qquad\qquad\qquad else \end{cases}$$

Neural Networks: The Perceptron

Also provided a learning algorithm

$$\mathbf{w} = \mathbf{w} + \eta(d(\mathbf{x}) - y(\mathbf{x}))\mathbf{x}$$

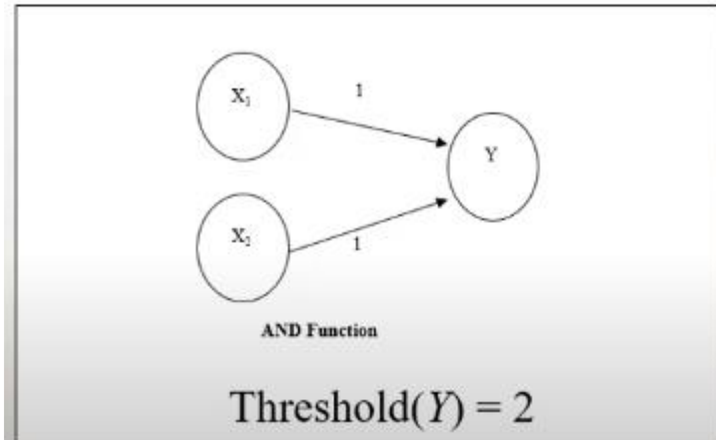
Sequential Learning:

$d(x)$ is the desired output in response to input $\mathbf{x}$

$y(x)$ is the actual output in response to $\mathbf{x}$

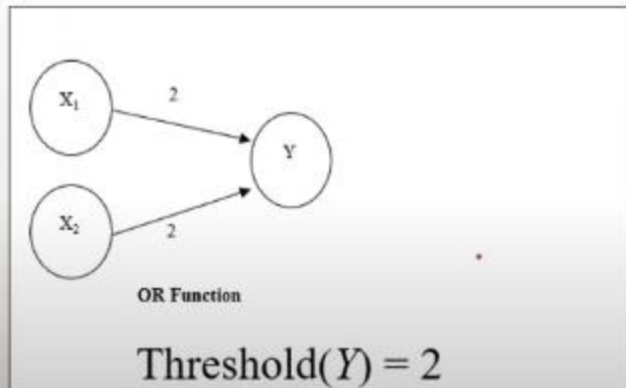
- Boolean tasks
- Update the weights whenever the perceptron output is wrong
  - Update the weight by the product of the input and the *error* between the desired and actual outputs
- Proved convergence for linearly separable classes

Neural Networks: The Perceptron



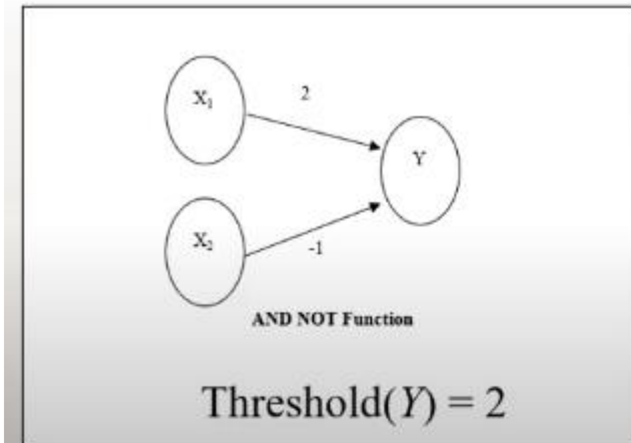
AND		
X1	X2	Y
1	1	1
1	0	0
0	1	0
0	0	0

Neural Networks: The Perceptron



OR		
X1	X2	Y
1	1	1
1	0	1
0	1	1
0	0	0

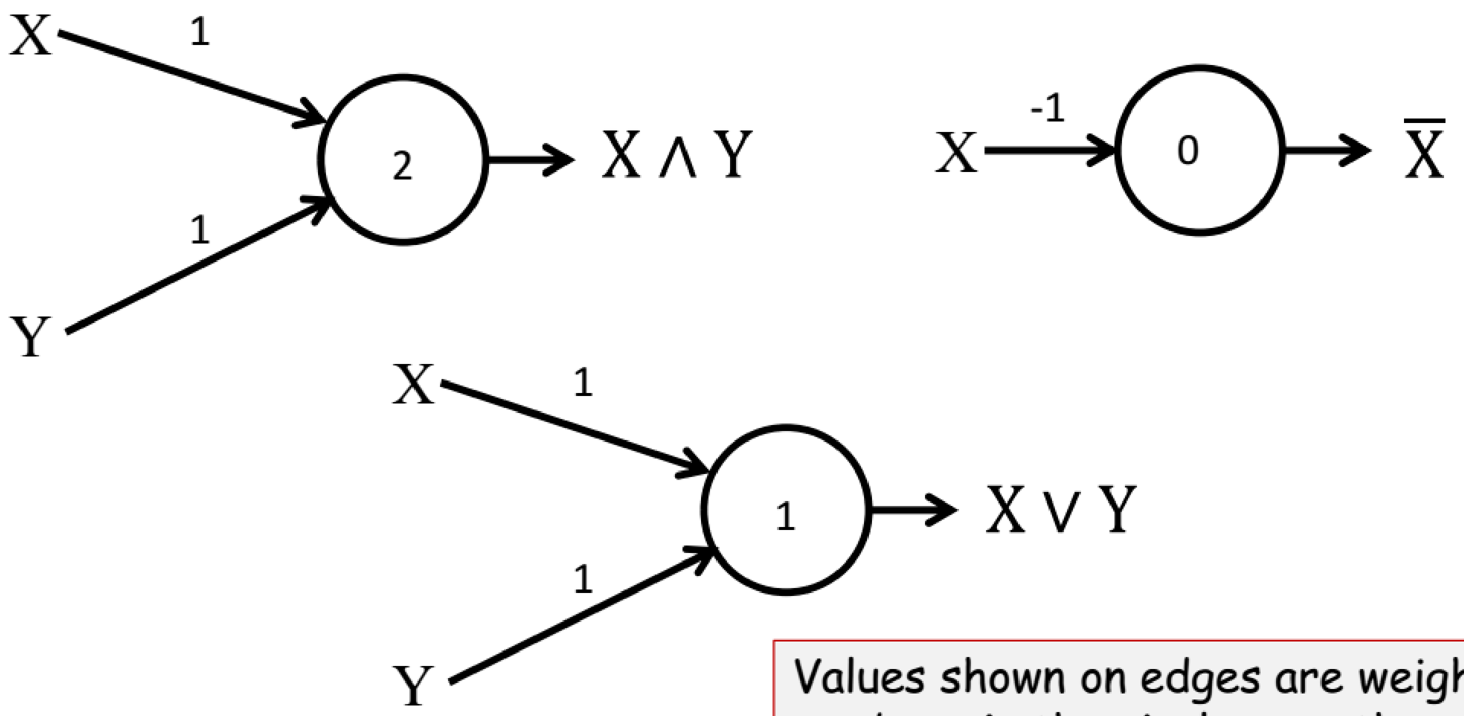
Neural Networks: The Perceptron



AND NOT		
X1	X2	Y
1	1	0
1	0	1
0	1	0
0	0	0

Neural Networks: The Perceptron

Perceptron

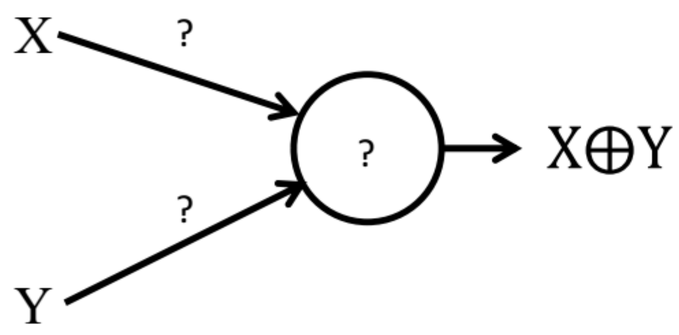


Values shown on edges are weights, numbers in the circles are thresholds

- Easily shown to mimic any Boolean gate
- But…

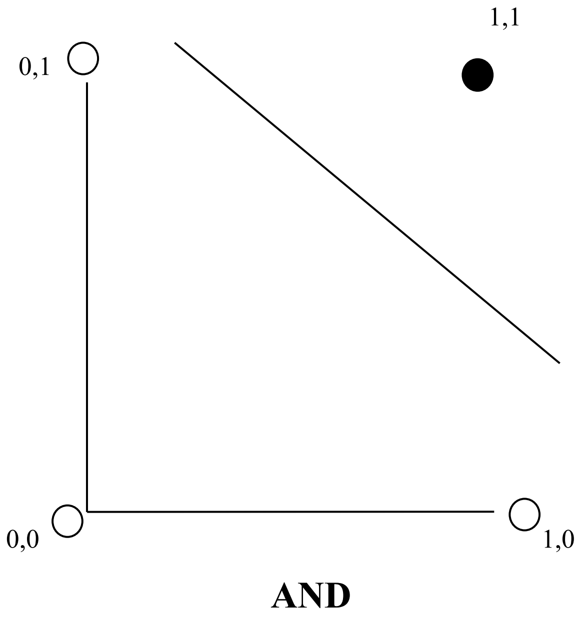
Individual units

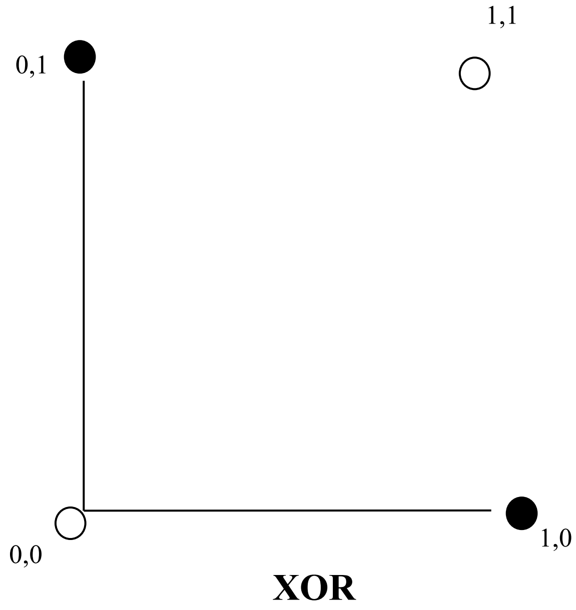
No solution for XOR!



Neural Networks: The Perceptron

- What can perceptrons represent?

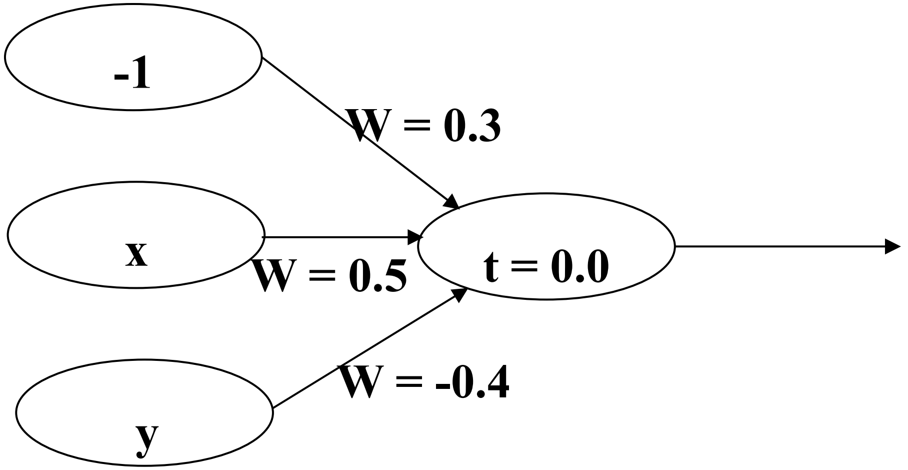




- Functions which can be separated in this way are called Linearly Separable.
- Only linearly Separable functions can be represented by a perceptron.

Neural Networks: The Perceptron

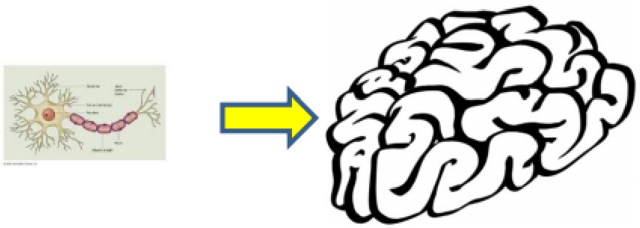
- Example: Training a Perceptron

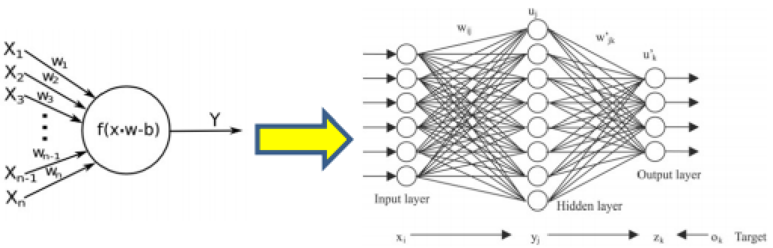


I_1	I_2	I_3	Summation	Output
-1	0	0	(-1*0.3) + (0*0.5) + (0*-0.4) = -0.3	0
-1	0	1	(-1*0.3) + (0*0.5) + (1*-0.4) = -0.7	0
-1	1	0	(-1*0.3) + (1*0.5) + (0*-0.4) = 0.2	1
-1	1	1	(-1*0.3) + (1*0.5) + (1*-0.4) = -0.2	0

Neural Networks: The Perceptron

A single neuron is not enough

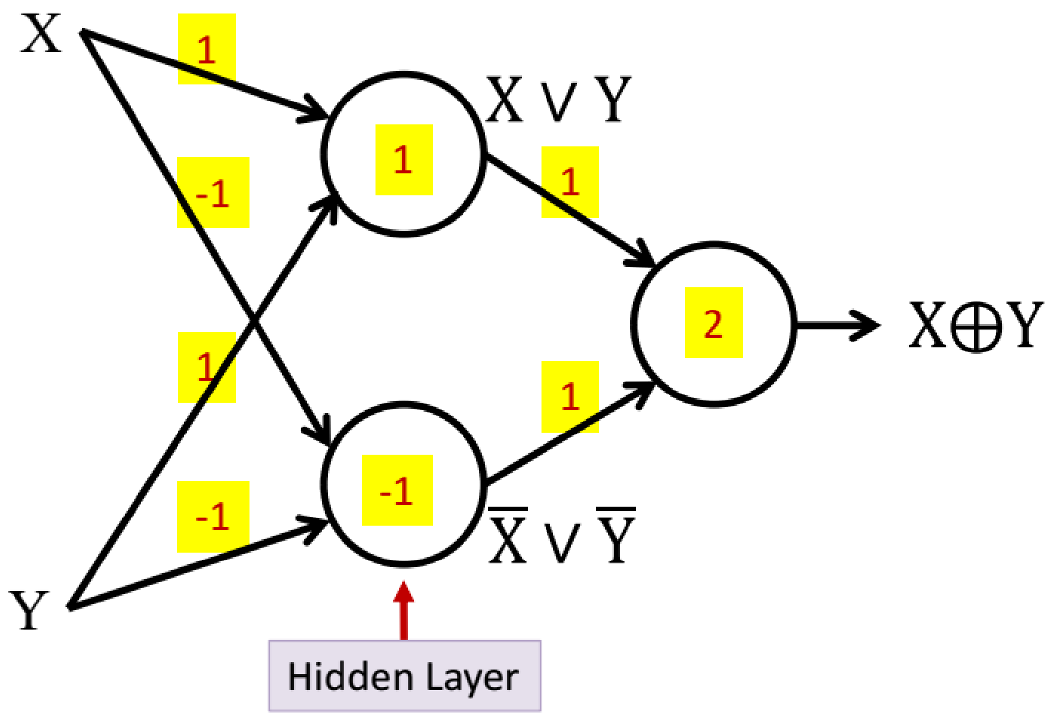




- Individual elements are weak computational elements
  - Marvin Minsky and Seymour Papert, 1969, *Perceptrons: An Introduction to Computational Geometry*
- *Networked* elements are required

Neural Networks: The Perceptron

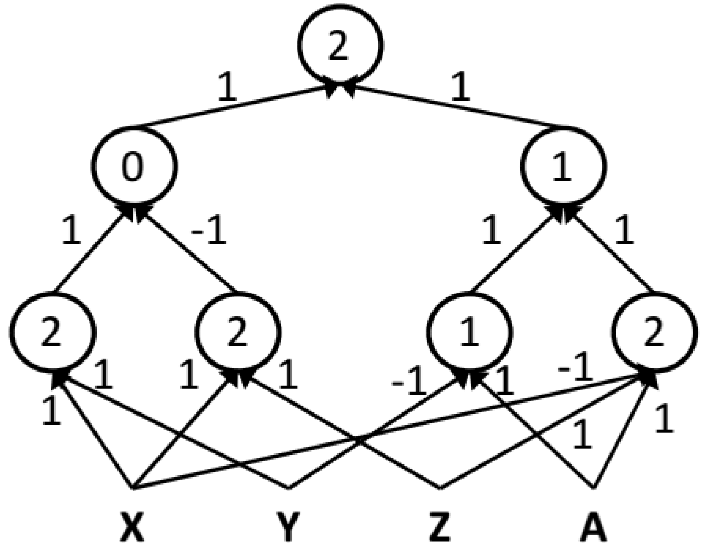
Multi-layer Perceptron!



- XOR
  - The first layer is a “hidden” layer

A more generic model

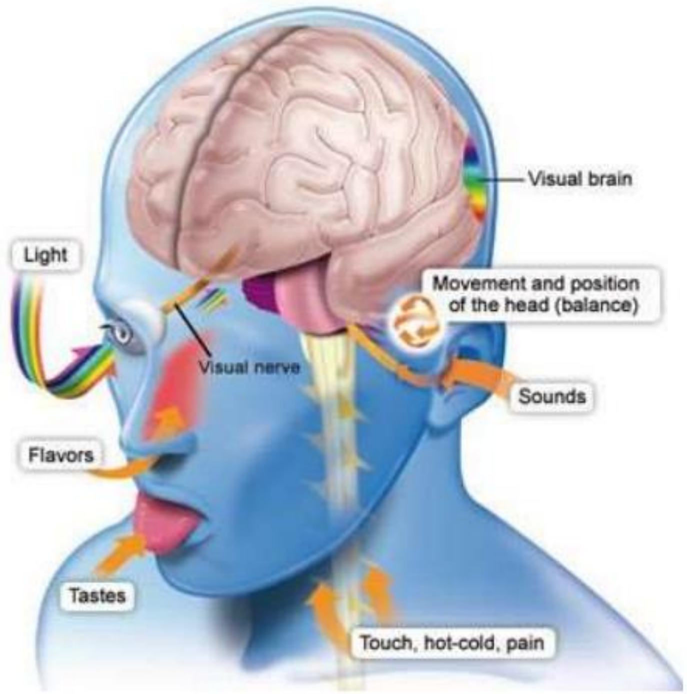
$$((A\&\bar{X}\&Z)|(A\&\bar{Y}))\&((X\ \&\ Y)|\overline{(X\&Z)})$$



- A “multi-layer” perceptron
- Can compose arbitrarily complicated Boolean functions!
  - In cognitive terms: Can compute arbitrary Boolean functions over sensory input
  - More on this in the next class

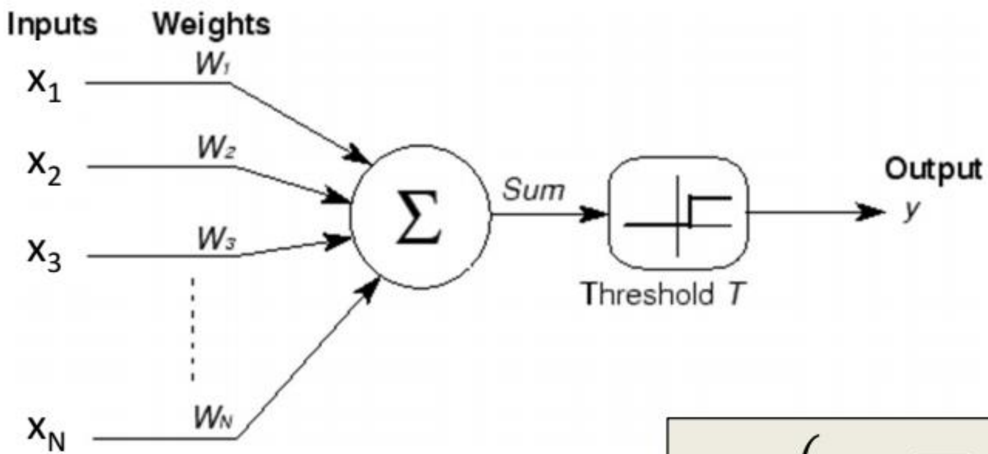
Neural Networks: The Perceptron

But our brain is not Boolean



- We have real inputs
- We make non-Boolean inferences/predictions

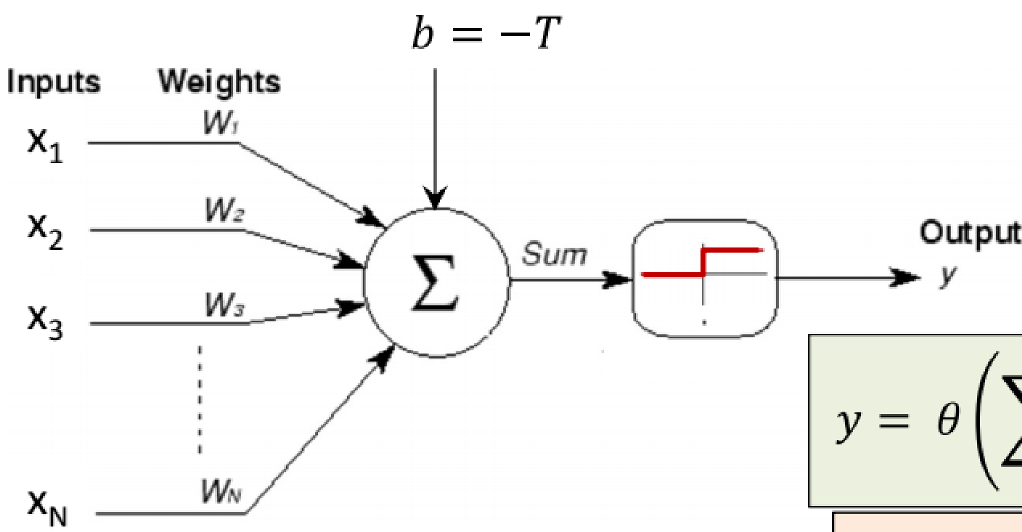
The perceptron with *real* inputs



$$y = \begin{cases} 1 \; if \; \sum_i w_i x_i - T \geq 0 \\ 0 \; else \end{cases}$$

- $x_1 \ldots x_N$ are real valued
- $w_1 \ldots w_N$ are real valued
- Unit "fires" if weighted input matches (or exceeds) a threshold

The perceptron with *real* inputs

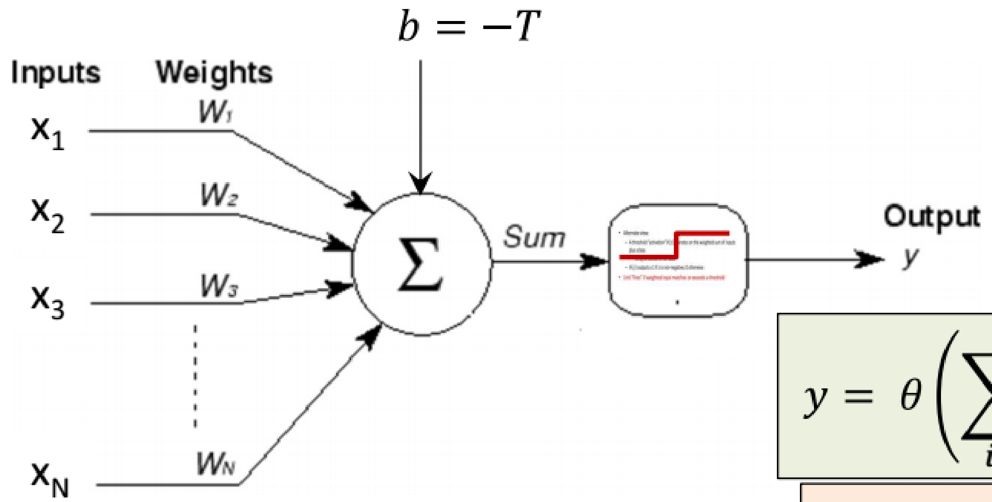


$$y = \theta\left(\sum_i w_i x_i + b\right)$$

$$\theta(z) = \begin{cases} 1 \; if \; z \geq 0 \\ 0 \; else \end{cases}$$

- Alternate view:
  - A threshold "activation" $\theta(z)$ operates on the weighted sum of inputs plus a bias
    - An *affine* function of the inputs
  - $\theta(z)$ outputs a 1 if z is non-negative, 0 otherwise
- Unit "fires" if weighted input matches or exceeds a threshold

The perceptron with *real* inputs



$$y = \theta\left(\sum_i w_i x_i + b\right)$$

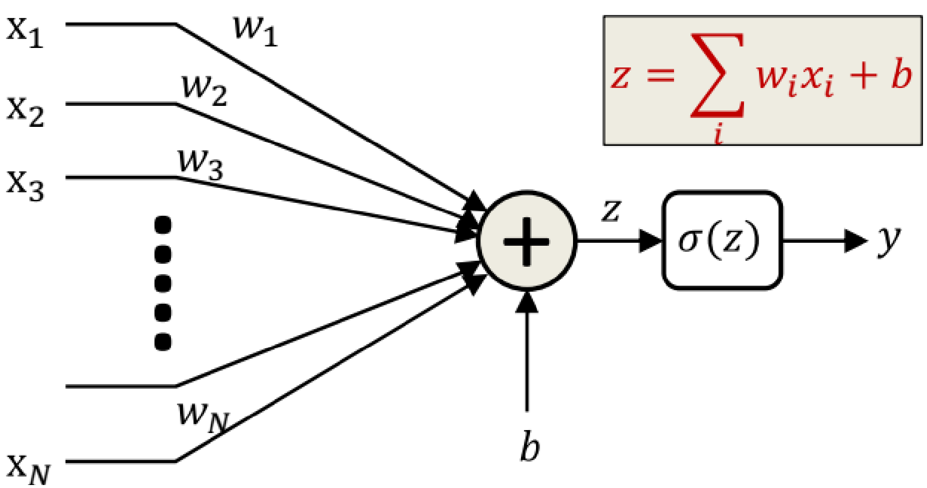
$$\theta(z) = \begin{cases} 1 \; if \; z \geq 0 \\ 0 \; else \end{cases}$$

- Alternate view:
  - A threshold "activation" $\theta(z)$ operates on the weighted sum of inputs plus a bias
    - An *affine* function of the inputs
  - $\theta(z)$ outputs a 1 if z is non-negative, 0 otherwise
- Unit "fires" if weighted input matches or exceeds a threshold

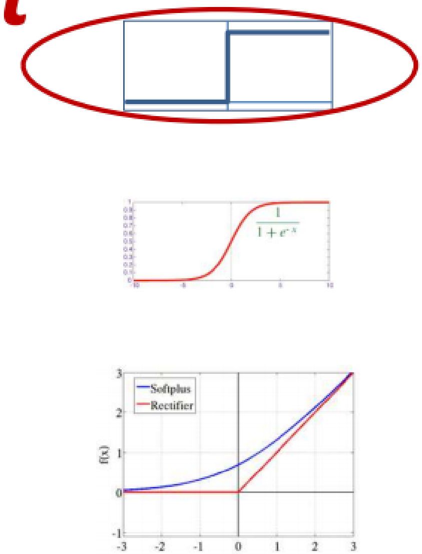
What is the difference between "linear" and "affine"?

Neural Networks: The Perceptron

The perceptron with *real* inputs and a real *output*



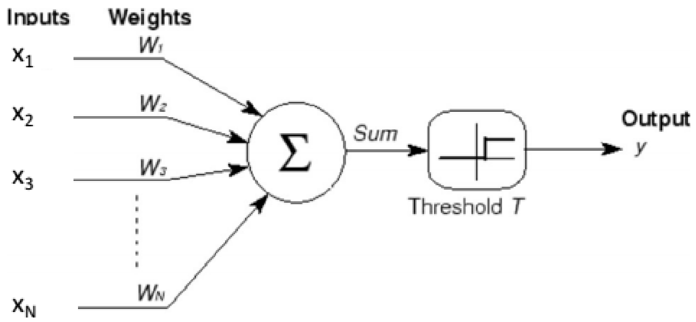
$$z = \sum_i w_i x_i + b$$



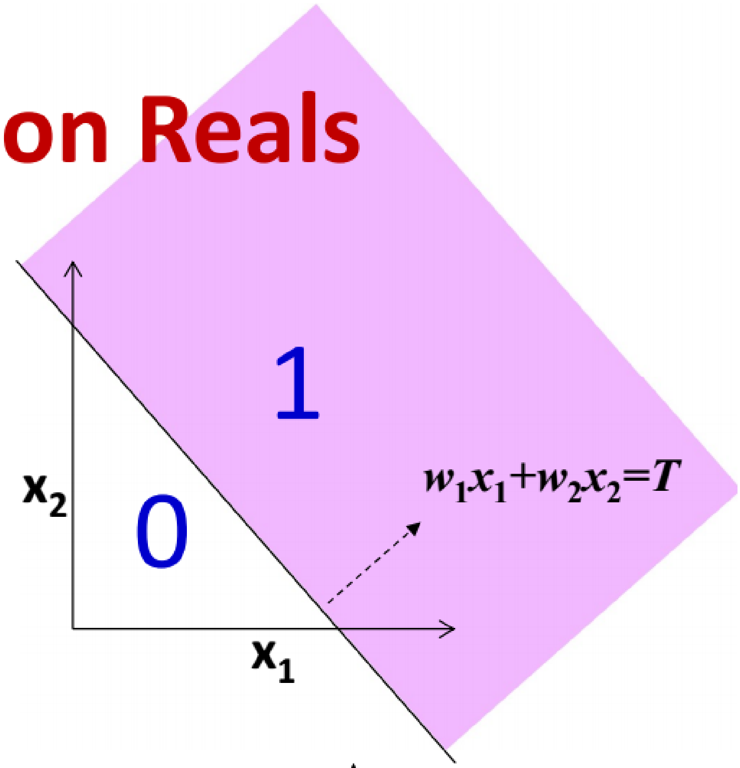
Activation functions $\sigma(z)$

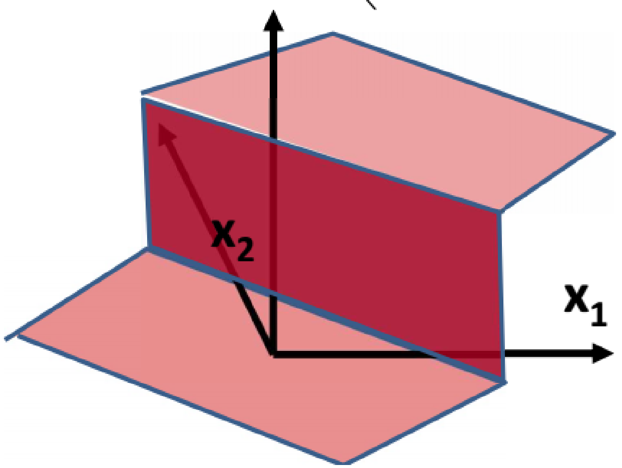
- $x_1 \ldots x_N$ are real valued
- $w_1 \ldots w_N$ are real valued
- The output y can also be real valued
- *For now we will continue to assume threshold activations*

A Perceptron on Reals



$$y = \begin{cases} 1 \; if \; \sum_i w_i x_i \geq T \\ 0 \; else \end{cases}$$

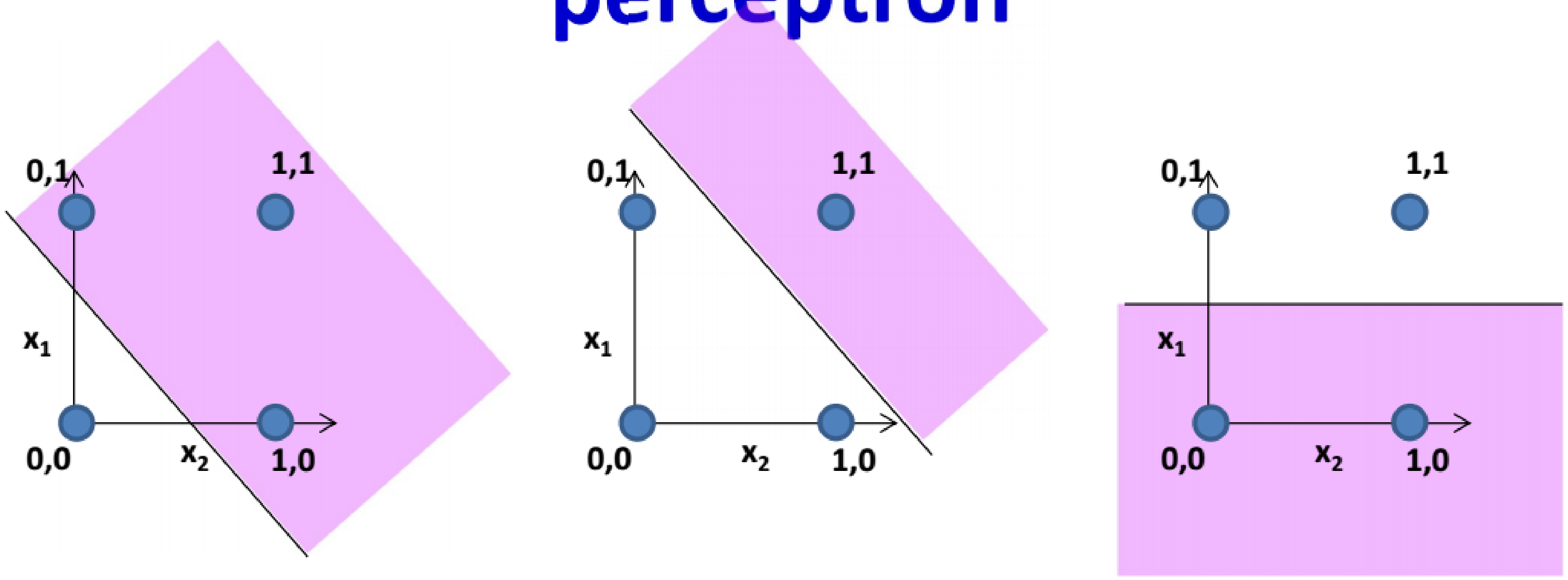




- A perceptron operates on *real*-valued vectors
  - **This is a *linear classifier***

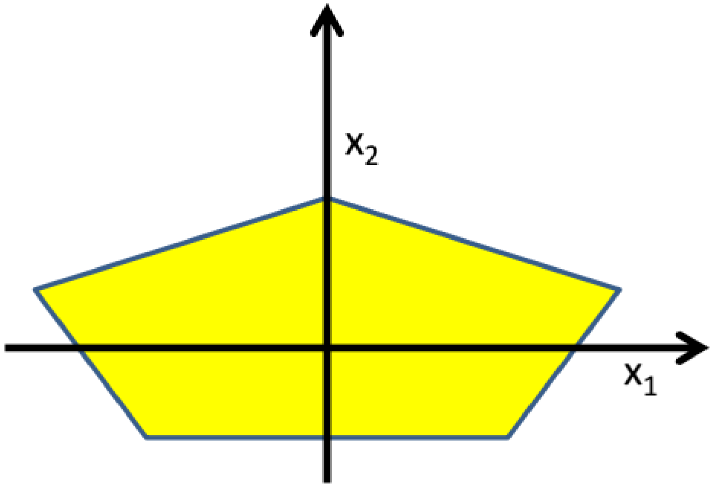
Neural Networks: The Perceptron

Boolean functions with a real perceptron



- Boolean perceptrons are also linear classifiers
  - Purple regions have output 1 in the figures
  - What are these functions
  - Why can we not compose an XOR?

Composing complicated "decision" boundaries

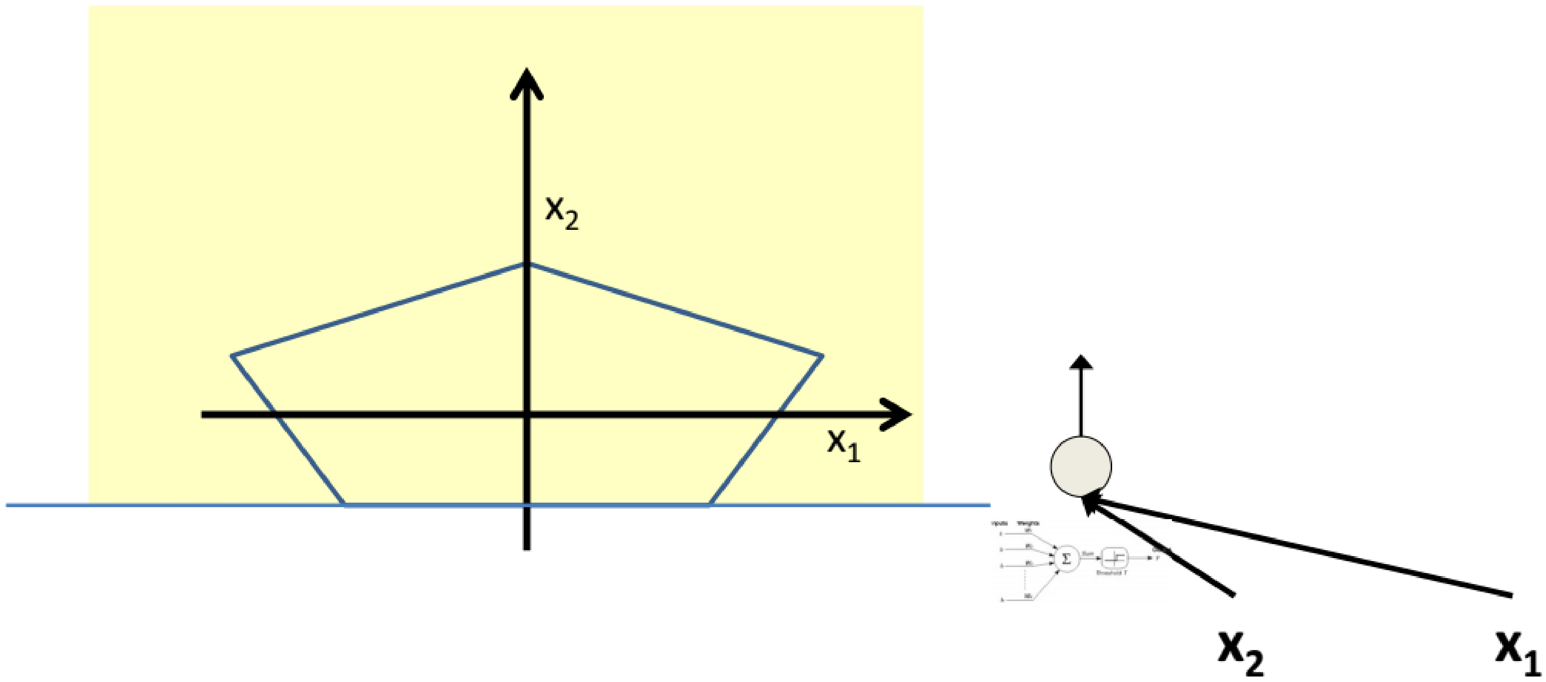


Can now be composed into "networks" to compute arbitrary classification "boundaries"

- Build a network of units with a single output that fires if the input is in the coloured area

Neural Networks: The Perceptron

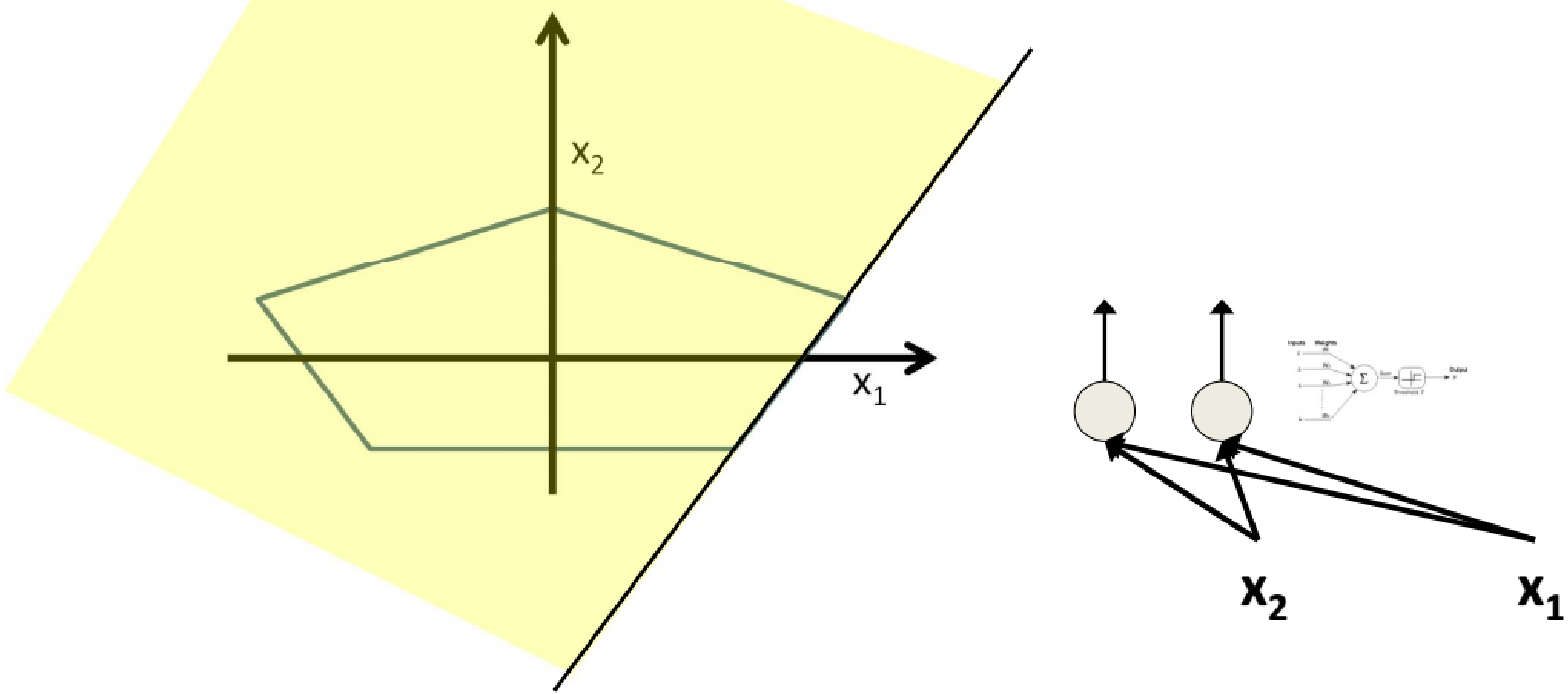
Booleans over the reals



- The network must fire if the input is in the coloured area

Neural Networks: The Perceptron

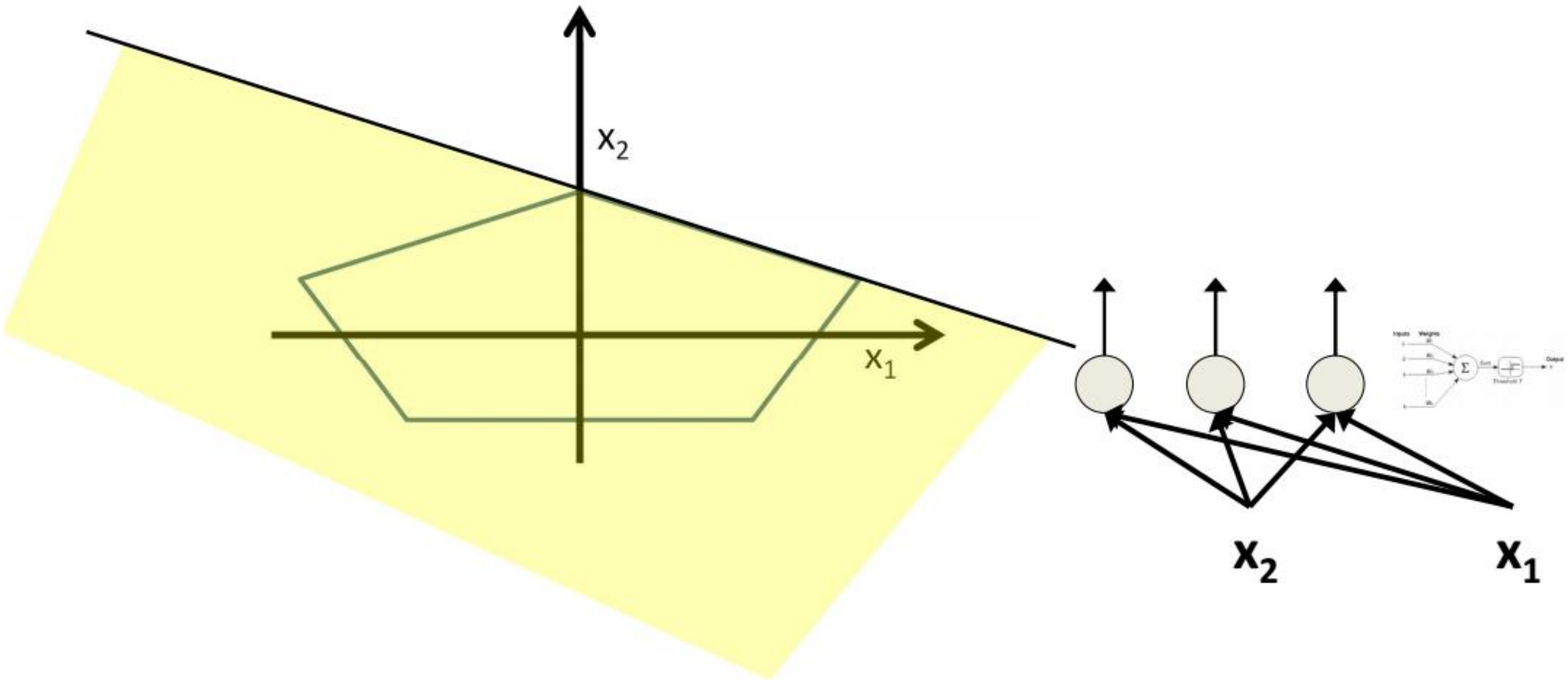
Booleans over the reals



- The network must fire if the input is in the coloured area

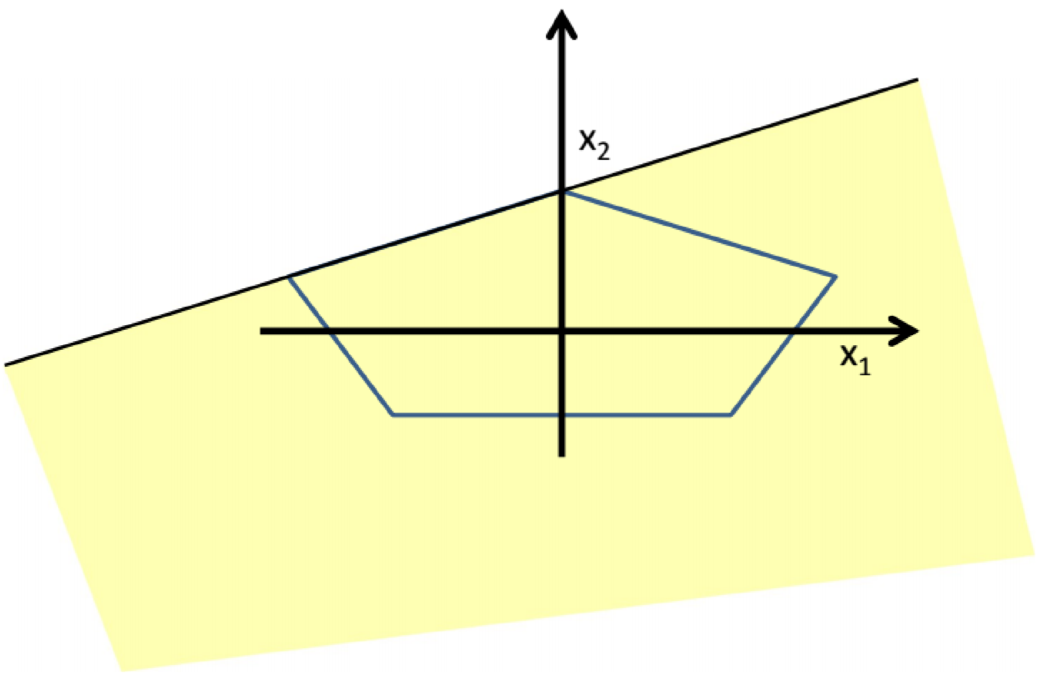
Neural Networks: The Perceptron

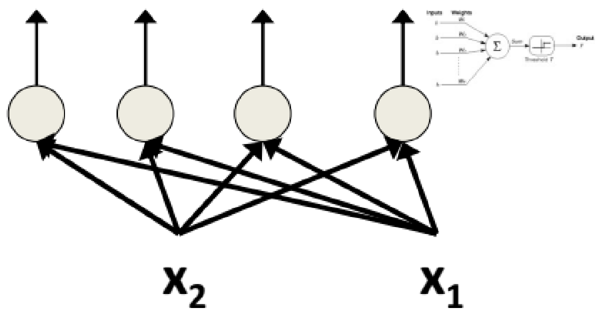
Booleans over the reals



- The network must fire if the input is in the coloured area

Booleans over the reals

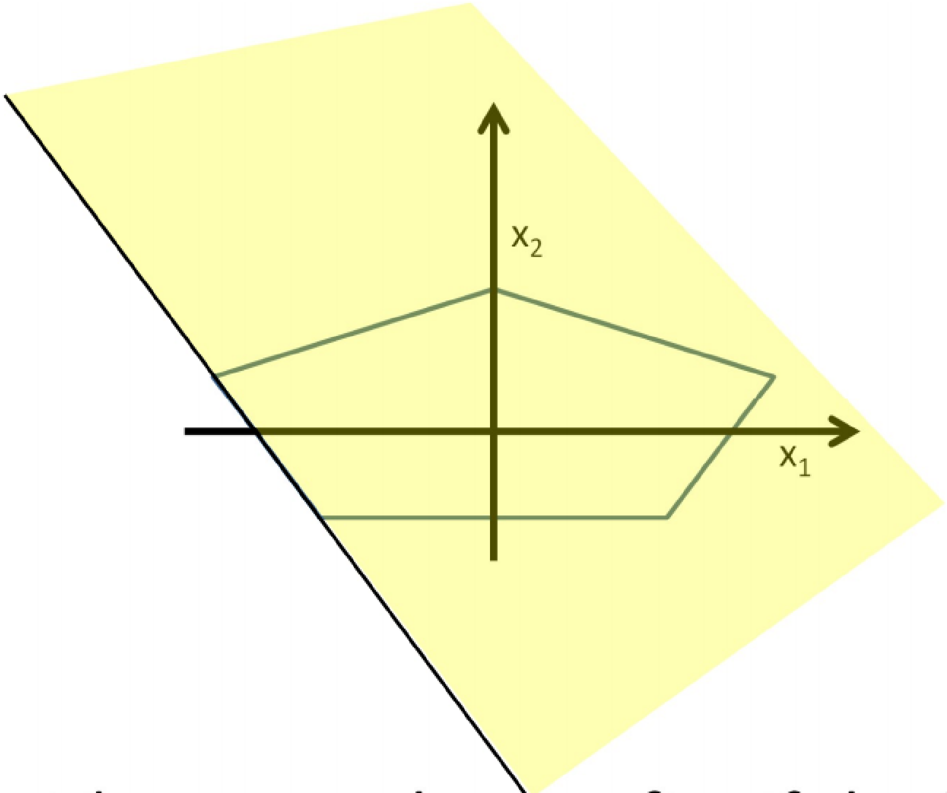


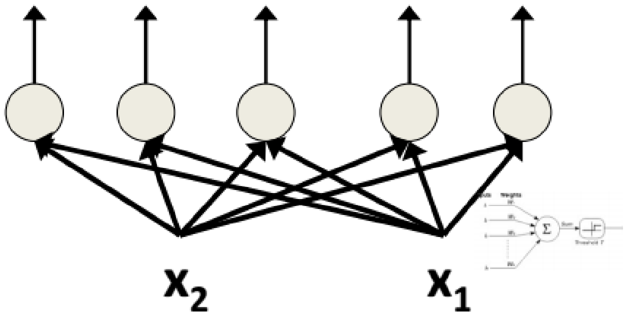


- The network must fire if the input is in the coloured area

Neural Networks: The Perceptron

Booleans over the reals

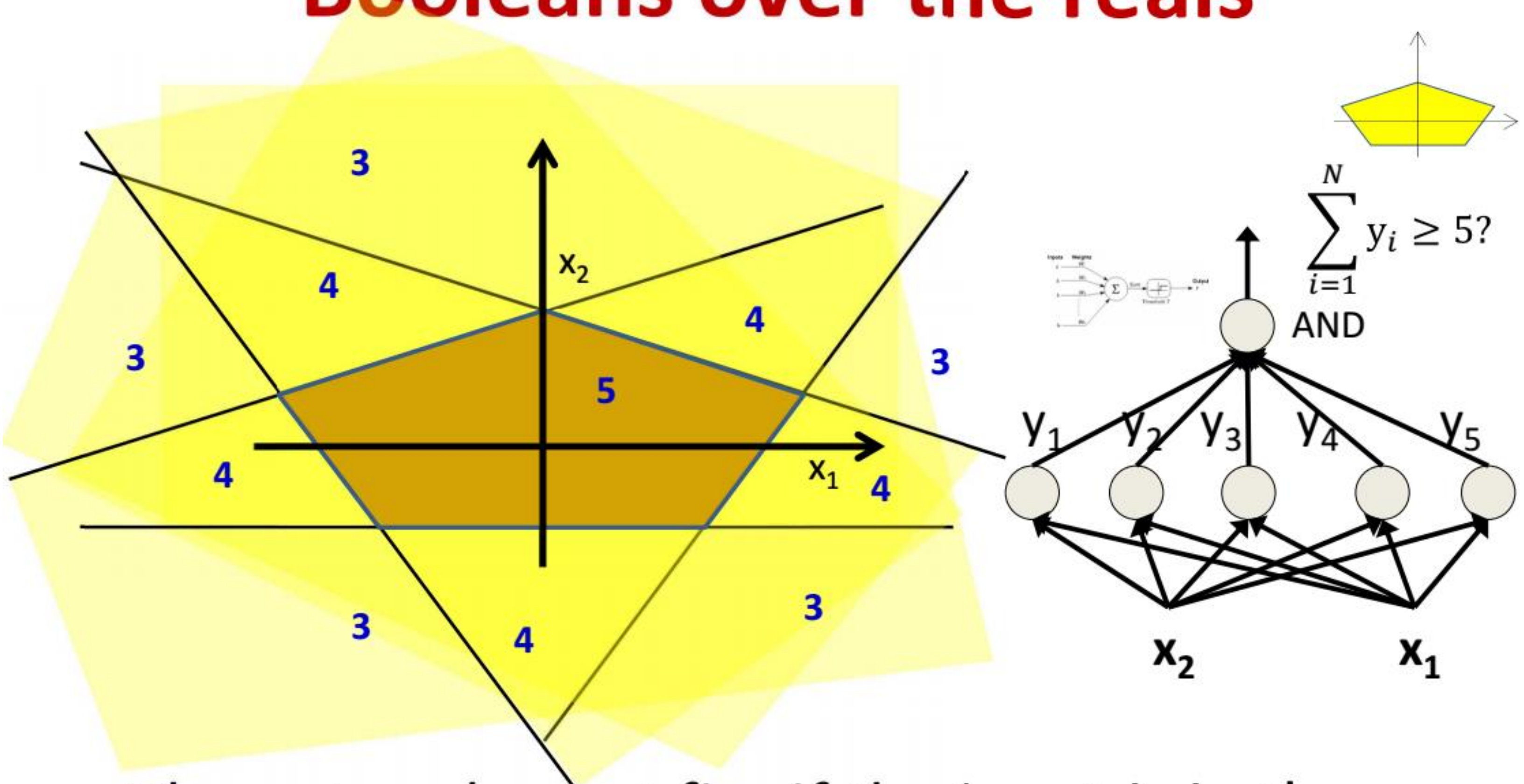




- The network must fire if the input is in the coloured area

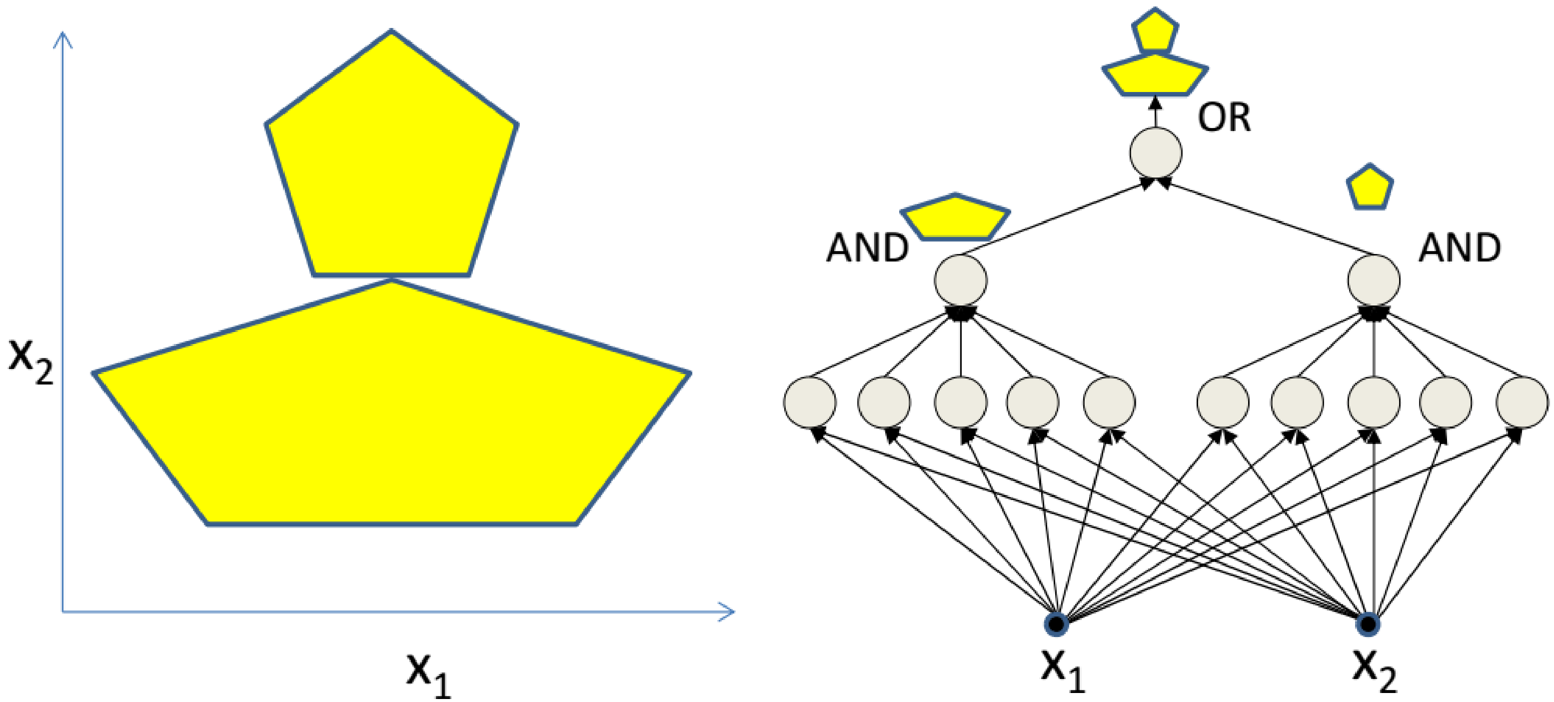
Neural Networks: The Perceptron

Booleans over the reals



- The network must fire if the input is in the coloured area

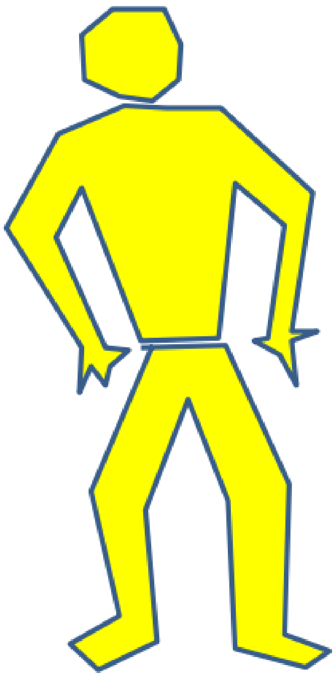
More complex decision boundaries



- Network to fire if the input is in the yellow area
  - "OR" two polygons
  - A third layer is required

Neural Networks: The Perceptron

Complex decision boundaries

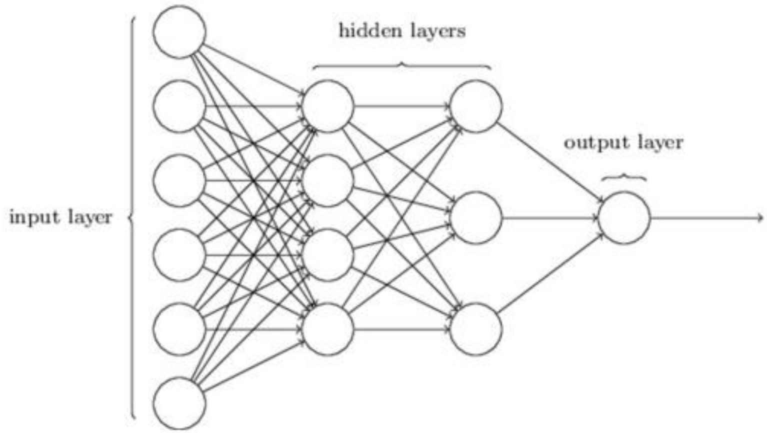


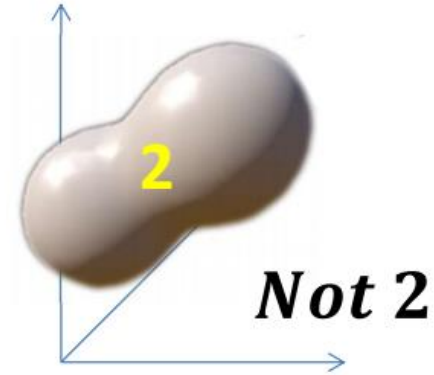
- Can compose very complex decision boundaries
  - How complex exactly? More on this in the next class

Neural Networks: The Perceptron

Complex decision boundaries

784 dimensions (MNIST)



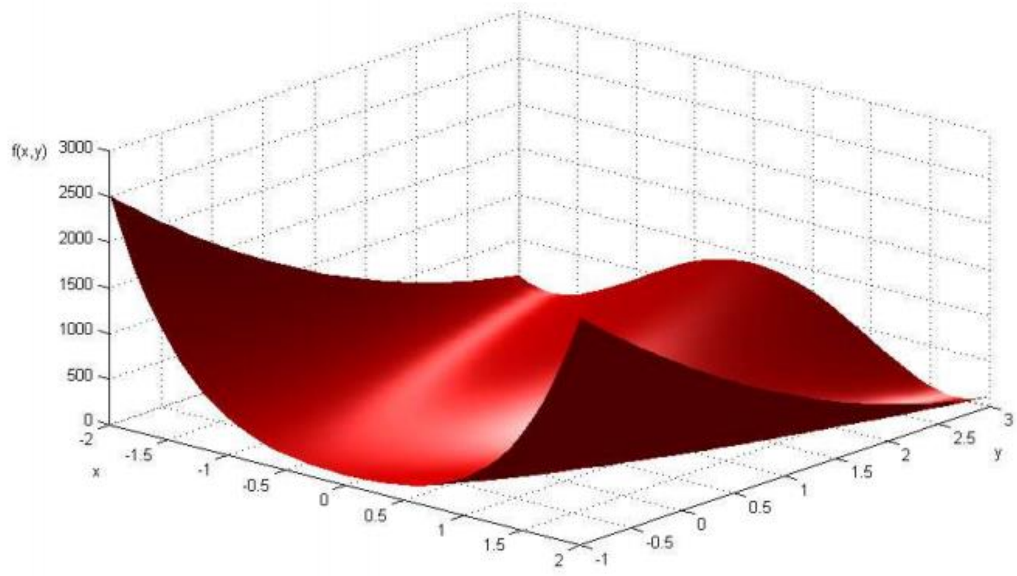


784 dimensions

- Classification problems: finding decision boundaries in high-dimensional space
  - Can be performed by an MLP
- MLPs can *classify* real-valued inputs
- They are universal classifiers
  - For any decision boundary, we can construct an MLP that captures it with arbitrary precision

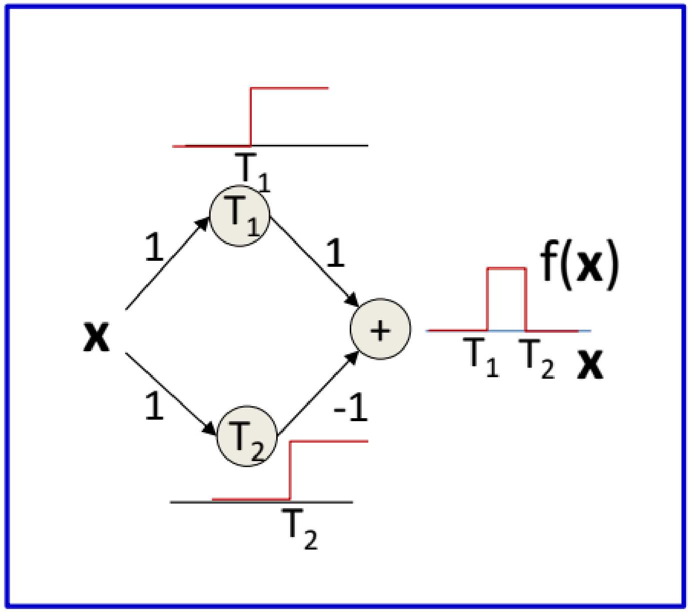
Neural Networks: The Perceptron

But what about continuous valued *outputs?*



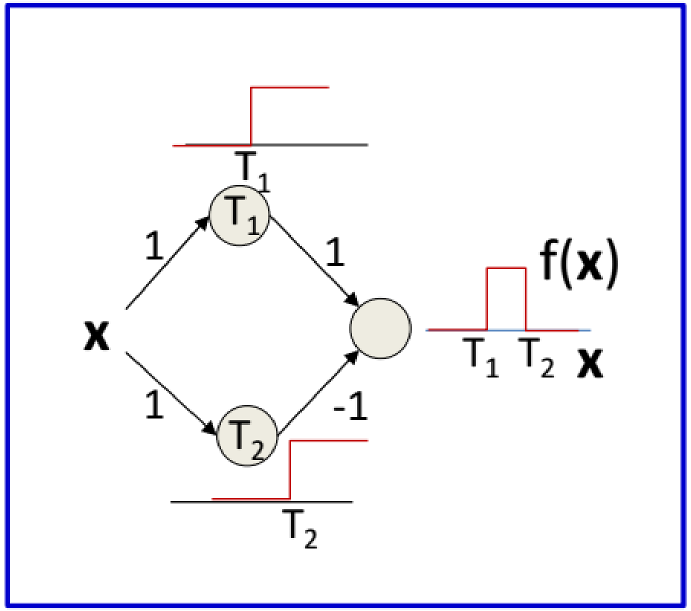
- Inputs may be real-valued
- Can outputs be continuous-valued too?

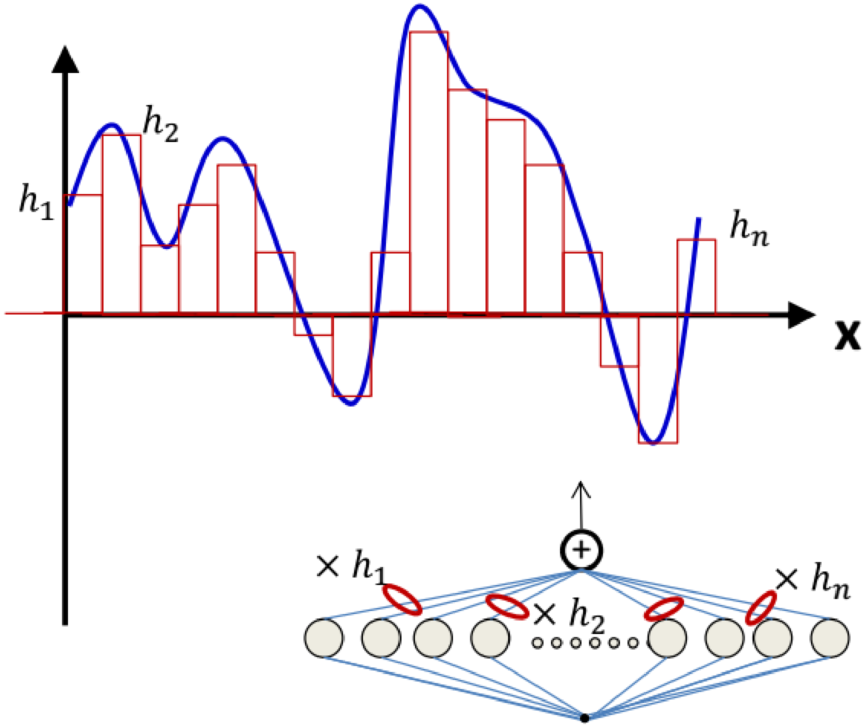
MLP as a continuous-valued regression



- A simple 3-unit MLP with a "summing" output unit can generate a "square pulse" over an input
  - Output is 1 only if the input lies between T_1 and T_2
  - T_1 and T_2 can be arbitrarily specified

MLP as a continuous-valued regression

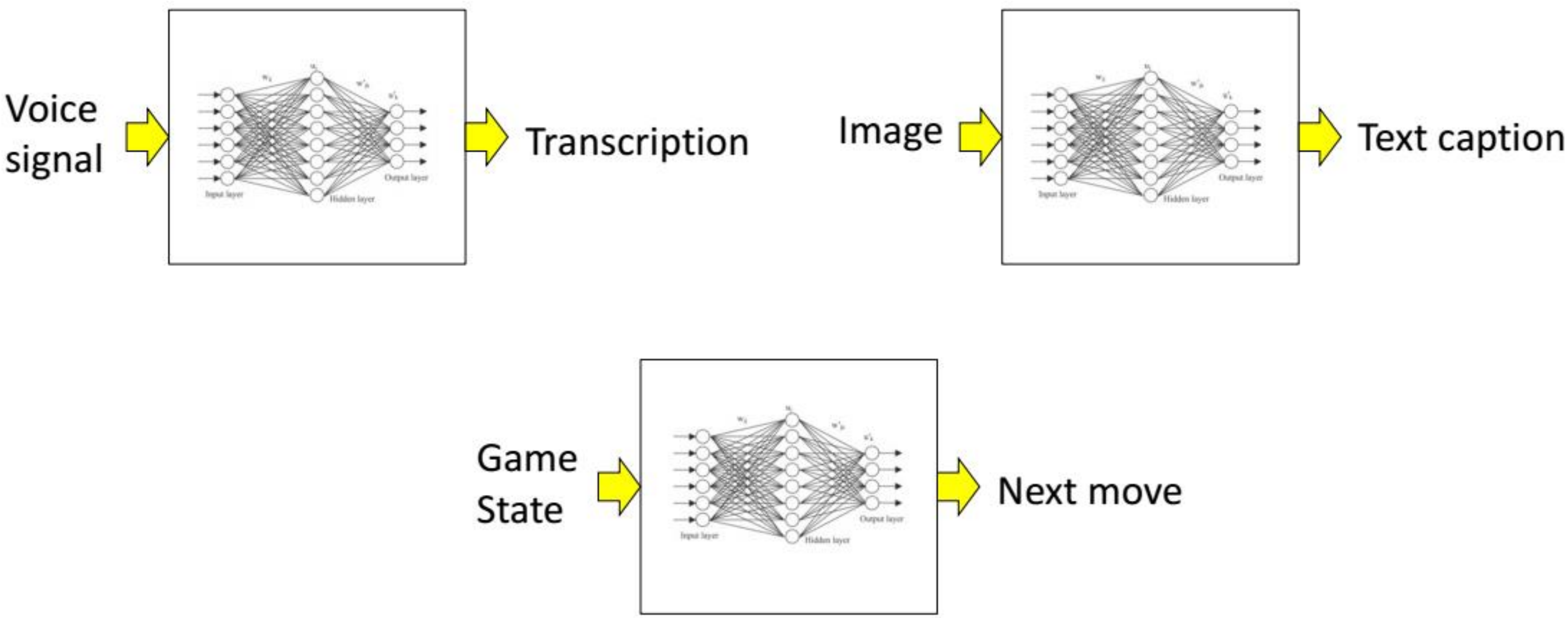




- A simple 3-unit MLP can generate a “square pulse” over an input
- **An MLP with many units can model an arbitrary function over an input**
  - To arbitrary precision
    - Simply make the individual pulses narrower
- This generalizes to functions of any number of inputs (next class)

Neural Networks: The Perceptron

These tasks are *functions*



- Each box is actually a function
  - E.g f: Image → Caption
  - It can be approximated by a neural network