CISC452/CMPE452/COGS400

Threshold Logic Units

Farhana Zulkernine

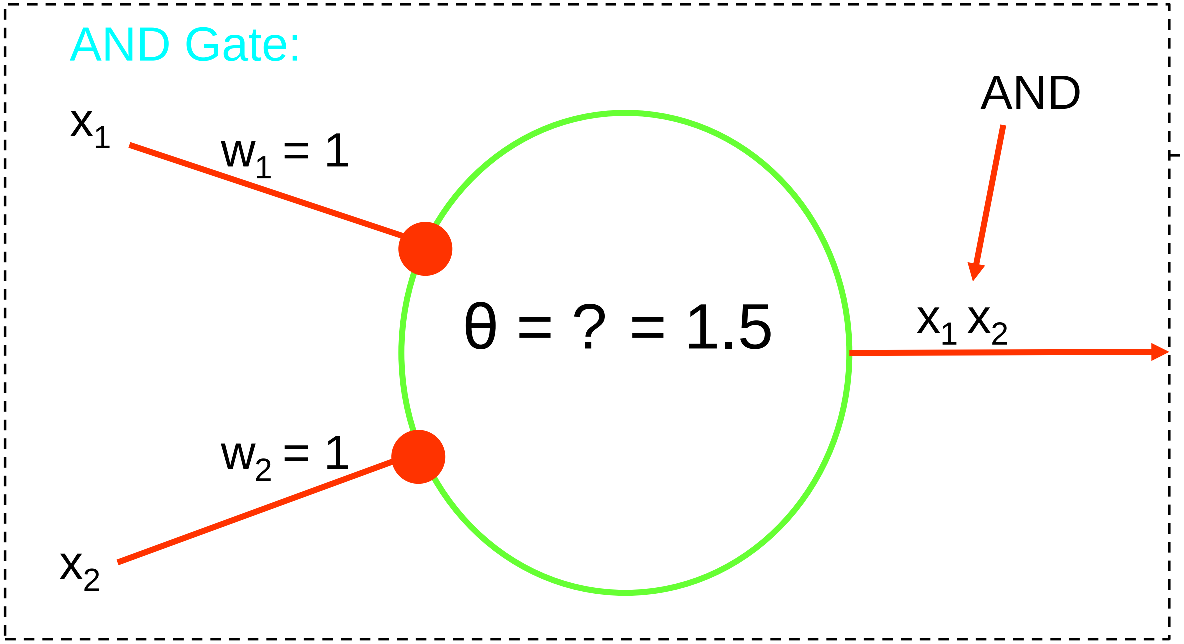
Threshold Logic Units (TLUs)

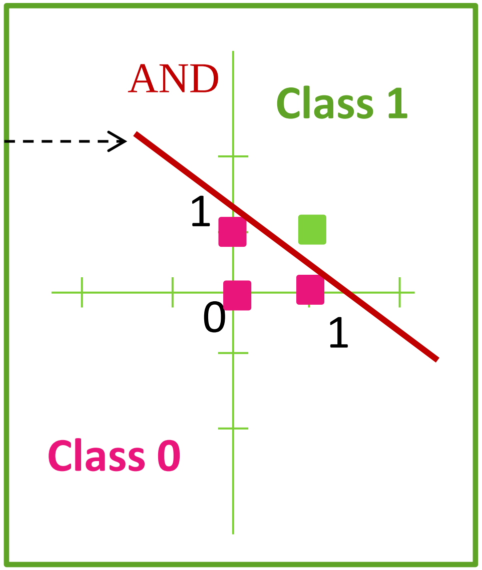
AND Gate

TLUs apply a threshold activation function and ***only accept binary inputs*** (0 or 1) – simulate logic gates.

AND Gate: 00, 01, 10= 0 and 11 = 1 (x,y)

Each point is represented by x_1x_2





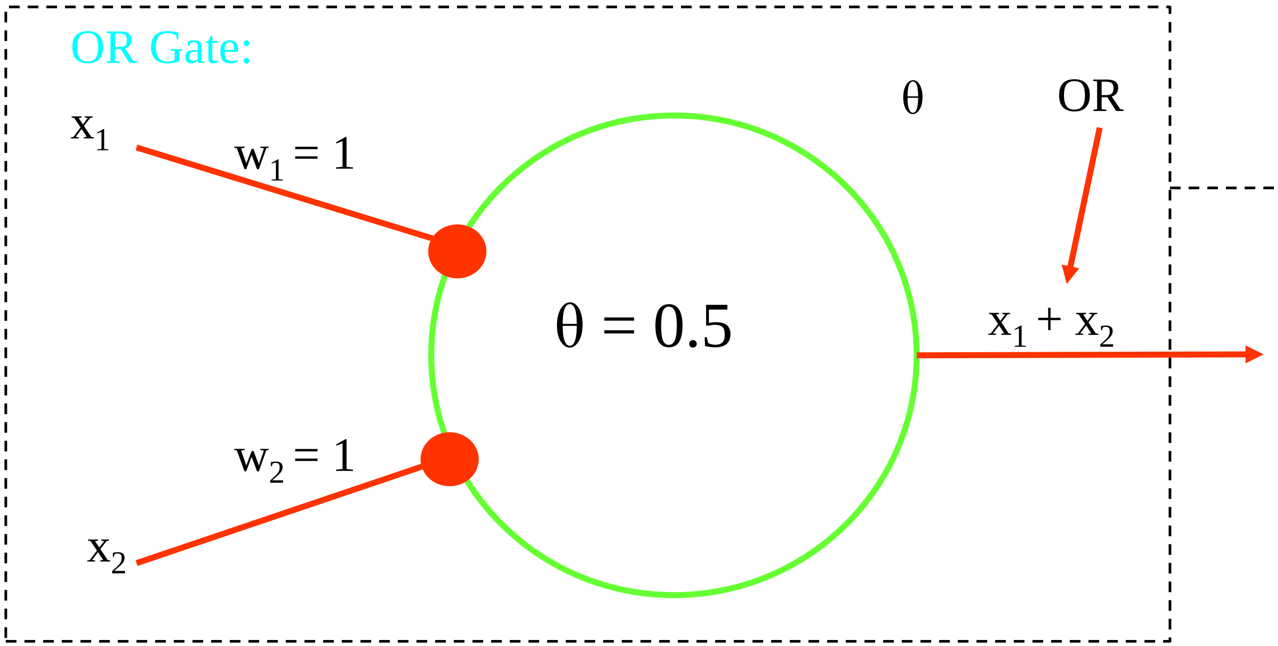
$w_1x_1 + w_2x_2 >= \theta$ **then output 1 else output 0**

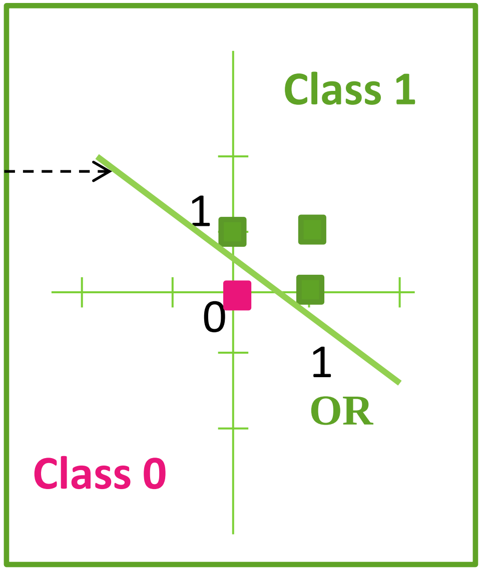
TLUs and Linear Separability – OR

OR Gate: **00 = 0** and **01, 10, 11 = 1**

TLU works as a linear separator for AND and OR gates.

Each point is represented by x_1x_2



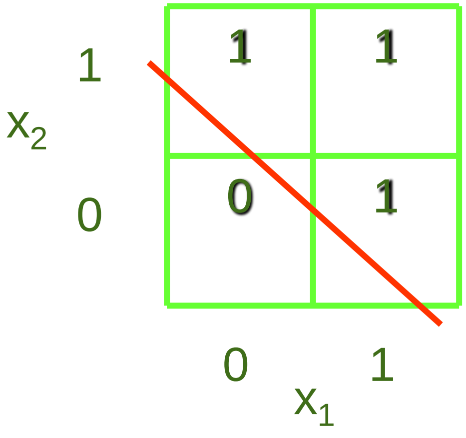


Input space

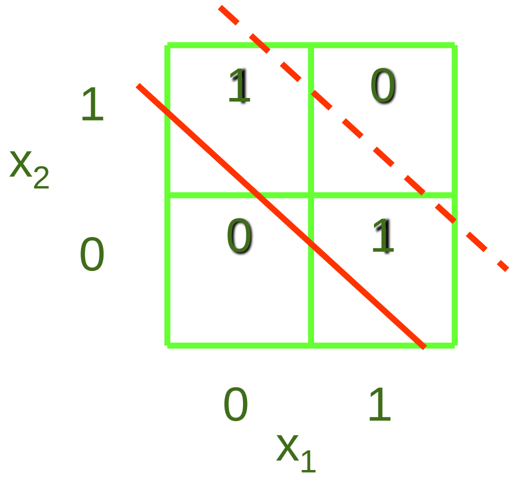
$w_1x_1 + w_2x_2 >= \theta$ then output 1 else output 0

What if the data are not linearly separable?

- A function f:$\{0, 1\}^n \to \{0, 1\}$ is linearly separable if the space of input vectors yielding 1 can be separated from those yielding 0 by a linear surface having (hyperplane) *n-1* dimensions.
- Examples: 2-D (dimensions) points: Separated by a 1-D line.



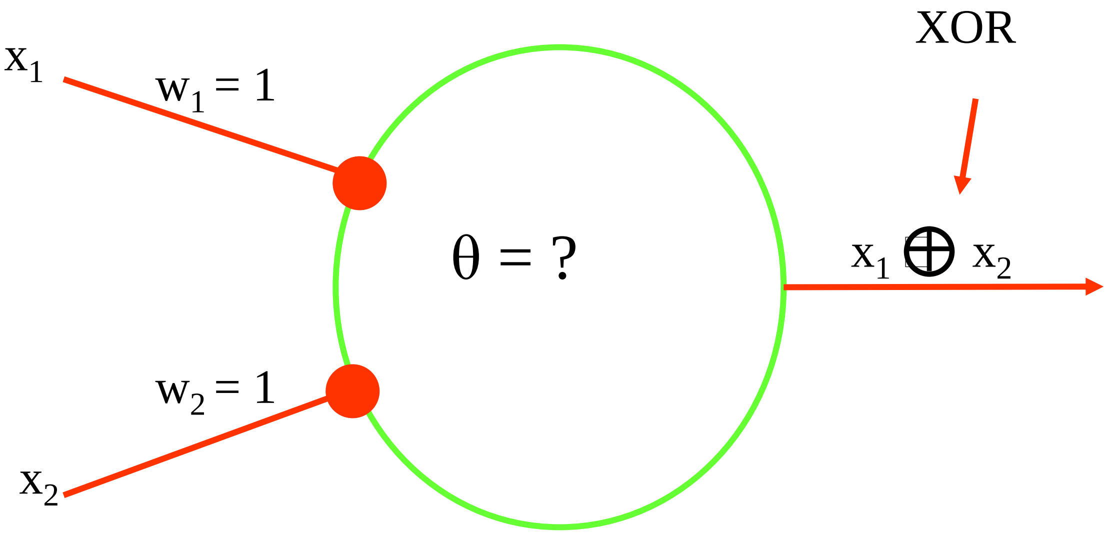
linearly separable (OR)



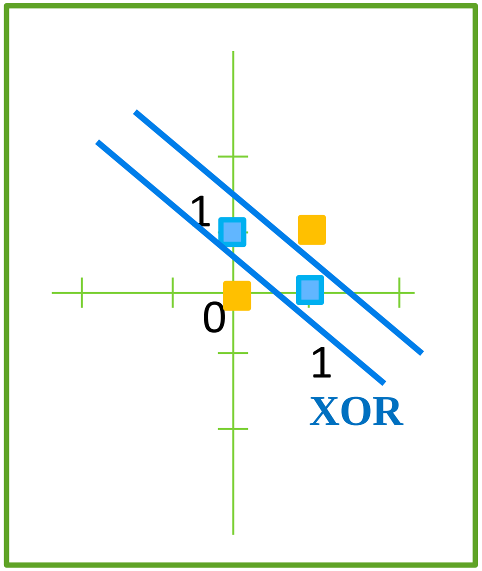
linearly inseparable (XOR)

TLUs cannot implement XOR

What about XOR?



Each point is represented by x_1x_2



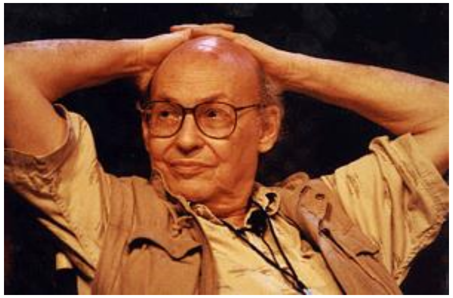
XOR Gate: **00,11=0** and **10,01=1**

NOT linearly separable by one line !!!

TLUs CANNOT realize functions that are NOT **linearly separable**.

Minsky and Papert

- In 1969, Marvin Minsky and Seymour Papert, two "PSS" researchers at MIT studied the ANNs and revealed that a two-layered (input and output) network cannot handle all logical relations – specifically the XOR.
  - It implies that ANNs lacked the power of a Turing machine.
- Federal funding for ANNs immediately stopped.



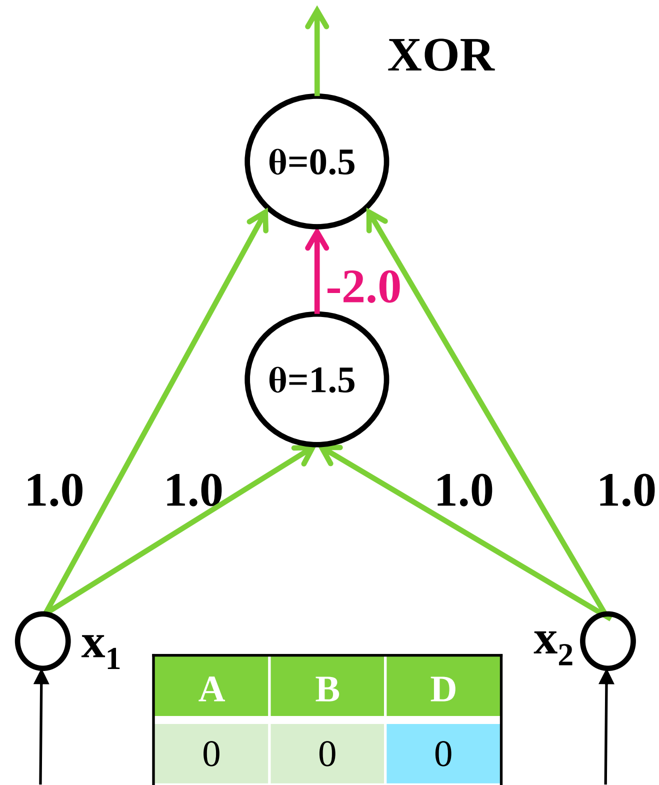
Minsky

The Big Breakthrough

- David Rumelhart and Jim McClelland developed Parallel Distributed Processing (PDP).

Multilayer Networks with *Inhibition*

- The solution that Rumelhart and McClelland propose is simple: Add a third layer between input and output.
- 3 layers enable creating an XOR gate and handle all logic.
- Note that middle layer neuron inhibits output layer neuron when $x_1 = x_2 = 1$.
  - New for ANN



A	B	D
0	0	0
0	1	1
1	0	1
1	1	0

Linear Separability as Functional Mapping

- To explain linear separability, let us consider the function f: $R^n \rightarrow \{0, 1\}$. θ is called the **bias**.

$$f(x_1, x_2, \ldots, x_n) = 1, \quad \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq \theta$$

$$= 0, \quad \text{otherwise}$$

where $x_1, x_2, \ldots, x_n$ represent real numbers (not TLU).

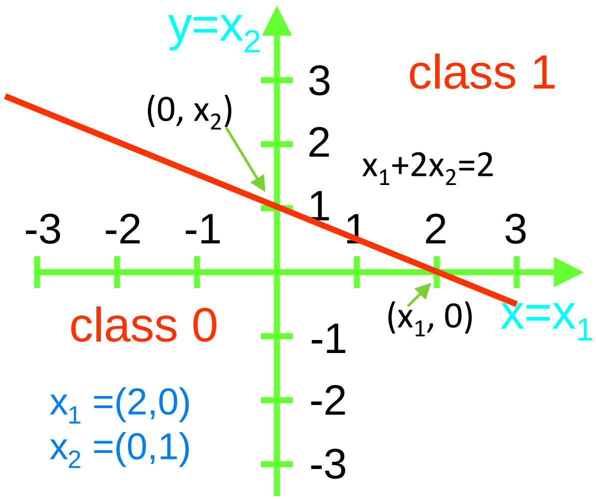
- Here the output function is generating values 0 or 1.
- $(w_1x_1 + w_2x_2 = \theta)$ represents the separator line in a 2-D space. Writing as $(y = mx + c)$ to plot the line

$\rightarrow x_2 = (-w_1/w_2)x_1 + \theta/w_2$

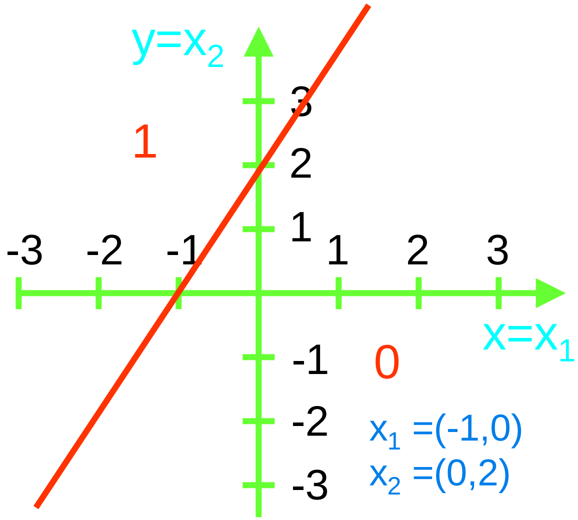
Examples – Linear Separability

Given x values and θ, we can determine the weights (w_1, w_2) to model the mapping using ANN using $x_2 = (-w_1/w_2)x_1 + \theta/w_2$

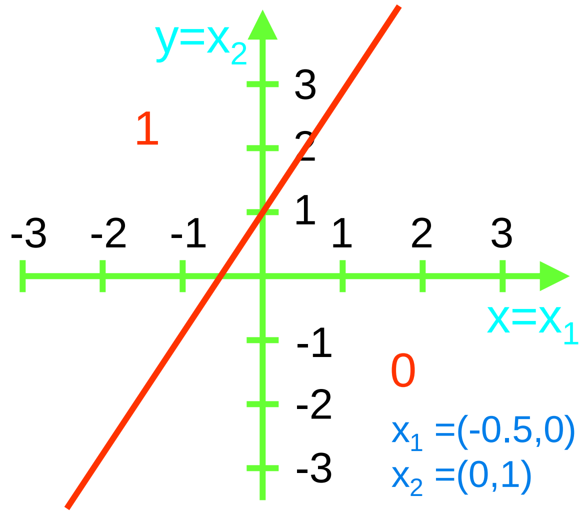
$$f(x_1, x_2, \ldots, x_n) = 1, \quad \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq \theta$$
$$= 0, \quad \text{otherwise}$$



$w_1 = 1$, $w_2 = 2$, $\boldsymbol{\theta = 2}$



$w_1 = -2$, $w_2 = 1$, $\boldsymbol{\theta = 2}$



$w_1 = -2$, $w_2 = 1$, $\boldsymbol{\theta = 2}$

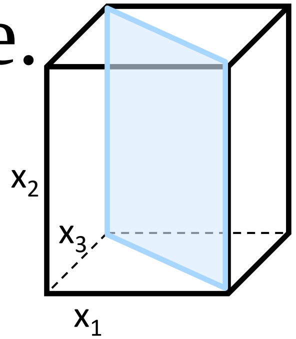
By changing w and θ, we can adjust the line to separate two classes in a 2-D space.

Learning/Training in ANN

- *Training a network means finding the best fitting values of θ and the weights* to categorize a given set of values correctly
  - Find the line that divides a labelled set of values into known categories.

Linear Separability (cont…)

- As we have seen, a two-dimensional input space can be divided by any straight line.
- A three-dimensional input space can be divided by any two-dimensional plane.



- In general, an *n-dimensional input space can be divided by an (n-1)-dimensional plane* or hyperplane.
- Of course, for n > 3 this is hard to visualize.

Activation as a Vector Product

The net input signal is the sum of all inputs where *x* values represent all the features in a data point:

$$\mathrm{net}_i(t) = \sum_{j=1}^{n} w_{i,j}(t) x_j(t)$$

$$\mathrm{net}_i(t) = |\boldsymbol{w_i(t)}| \cdot |\boldsymbol{x(t)}| \cdot \boldsymbol{\cos \alpha} \quad = \mathbf{w}^{\boldsymbol{T}} \boldsymbol{x} = [w_1 w_2 w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

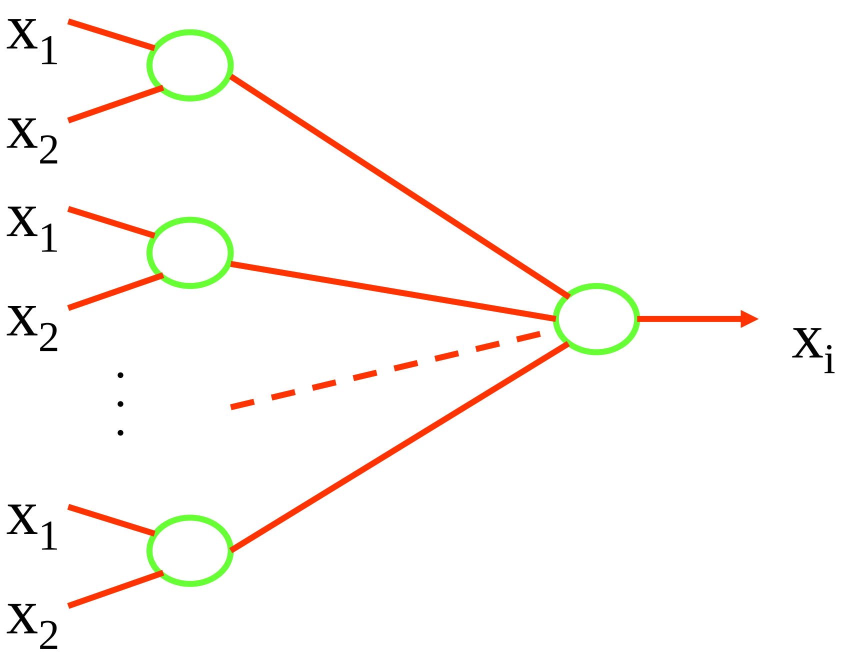
This can be viewed as computing the **inner product** of the **vectors** $\mathbf{w_i}$ and **x**: https://en.wikipedia.org/wiki/Dot_product

where α is the **angle** between the two vectors and

$|a| = \mathrm{sqrt}(a.a)$ and *a* is a given vector

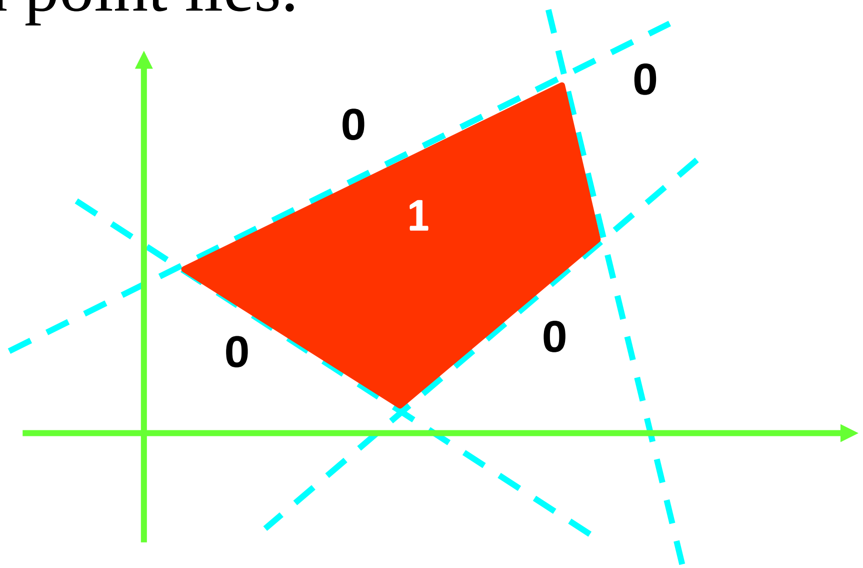
What about complex functions?

- So what can threshold neurons do for us?
- Let us consider a neuron with two inputs. We can combine multiple artificial neurons in multiple layers to form networks with increased capabilities.



Identify a Polygon

- The dotted lines can represent the first layer neurons which output 0 or 1 depending on which side of the line the input data point lies.



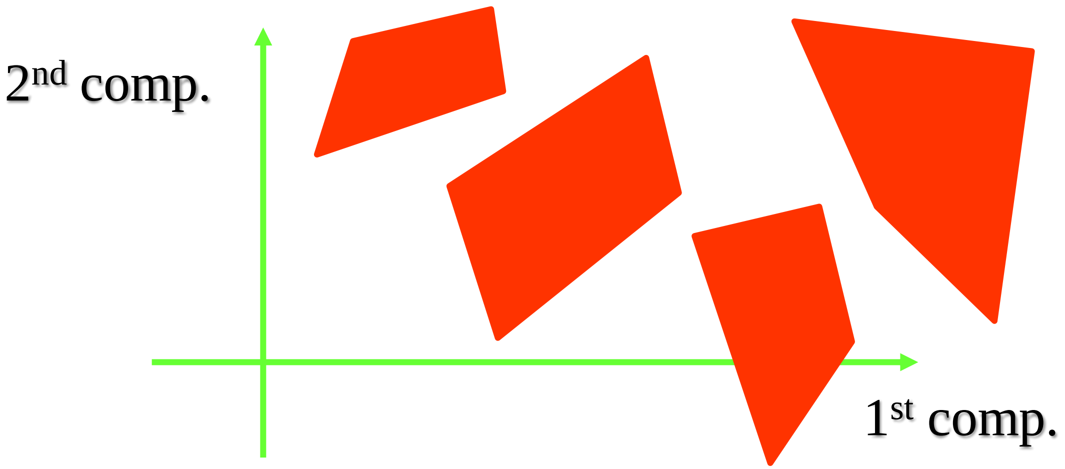
- Then, the second-layer neuron could output 1 **if all the outputs of the first layer neurons is 1 to identify a polygon, and 0 otherwise.**

Identify Multiple Polygons

- The more neurons there are in the first layer, the more vertices can the polygons have.
- With a sufficient number of first-layer neurons, the polygons can approximate any given shape.
- The more neurons there are in the second layer, the more of these polygons can be combined to form the output function of the network.
- With multi-layered ANN, you can identify many different shapes.

Capabilities (cont…)

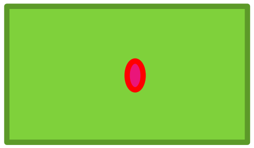
- Assume that the polygons in the diagram indicate the input regions for which each of the **second-layer** neurons yields output 1:

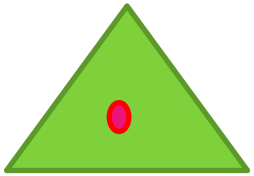


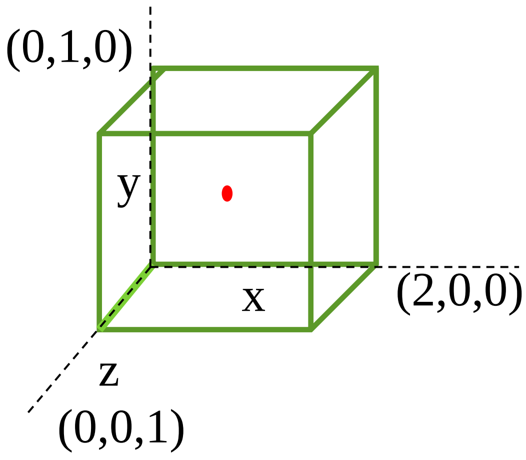
Then, for example, **the third-layer** neuron could output 1 if the input is within **any of the polygons**, and 0 otherwise. Example application ???

Problem

1. Draw an ANN that can detect whether a point lies on the **surface** of a square plane or a triangle. Explain how it would work.
2. Draw an ANN to find if a point lies **inside** a hollow box. Explain how it would work.

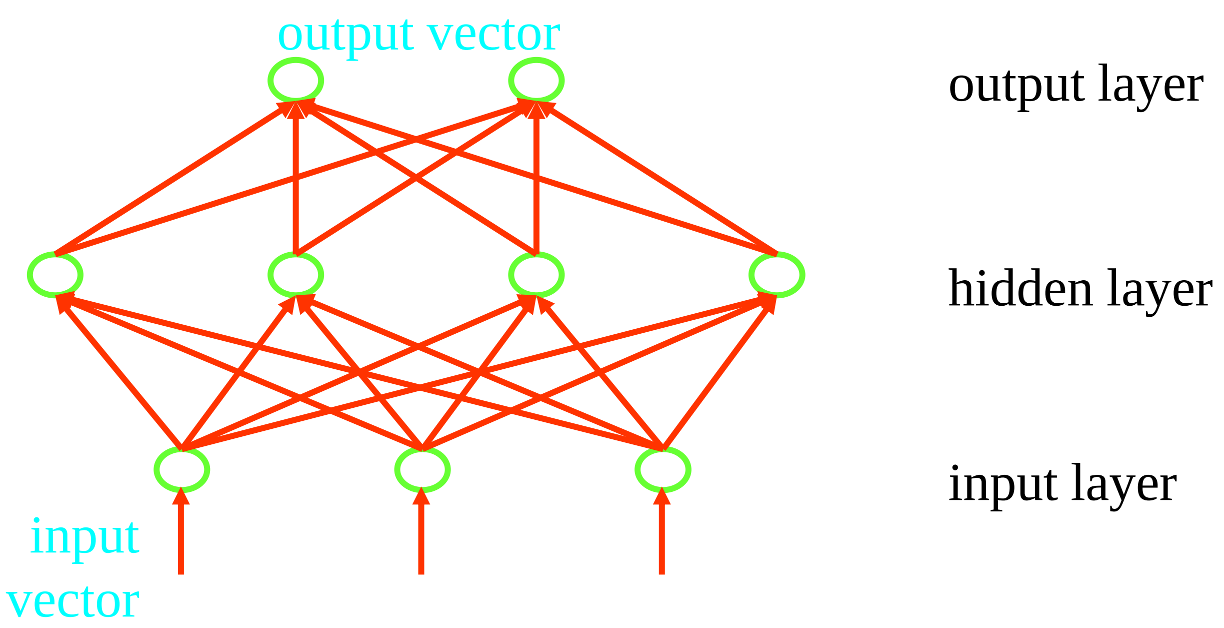






Terminology

- Example: Network function f: $R^3 \rightarrow \{0, 1\}^2$ means that the network takes 3-dimensional real values as input and generates two-dimensional binary output values.



Summary

- Complex ANNs can be created by combining neurons at multiple layers.
- Weights and bias are the variables that need to be assigned correct values to classify data points.
- The *input layer* just contains the input vector and *does not perform any computations other than distributing inputs to the next layers (used optionally)*.
- The intermediate layers, termed *hidden layers*, receives input from the input layer or previous hidden layers and *sends output to the final output layer*.
- After applying their *activation function*, the neurons in the final *output layer* generate the output vector.