CISC452/CMPE452/COGS400/CISC874

Learning in Artificial Neural Networks and Linear Separability

Farhana Zulkernine

Application

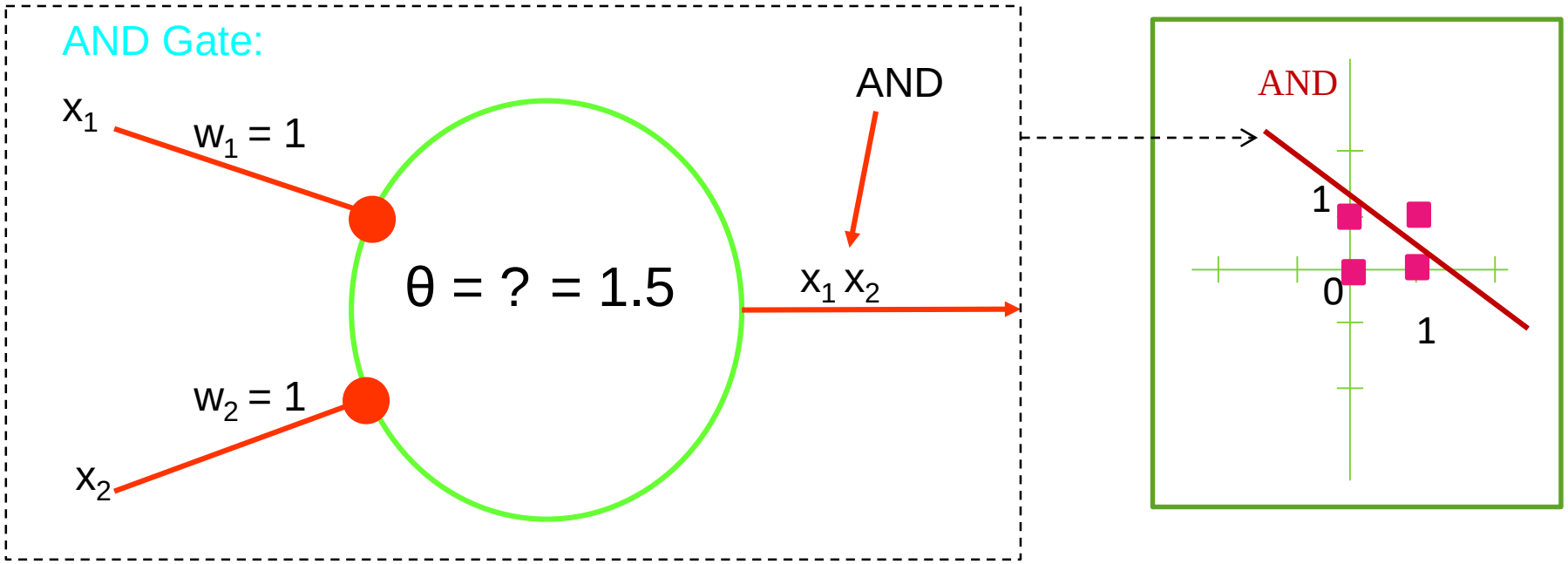
- Classification – Grouping and recognizing the label attached to the group
  - Label is known
  - Identify letters as ‘A’ and ‘B’ which denote class labels
- Clustering – Grouping based on distinctive and common features
  - Label in not known
  - Identify both as letters but different based on shape, name of the letter is not known, define central properties
- So we need logic implementation that if …. Then … category [class label] or group
  - Divide categories and groups using a separator

Threshold Logic Units (TLUs) and Linear Separability – AND Gate

TLUs are similar to the threshold neuron model, except that ***TLUs only accept binary inputs*** (0 or 1).

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

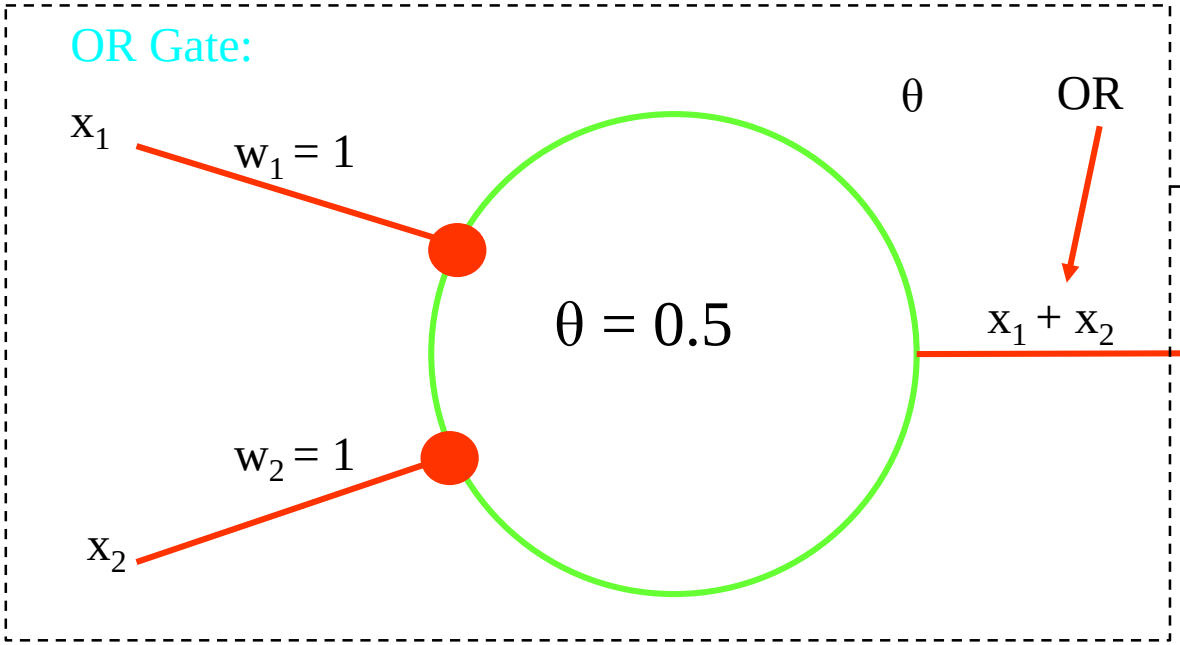
Each point is represented by x_1x_2

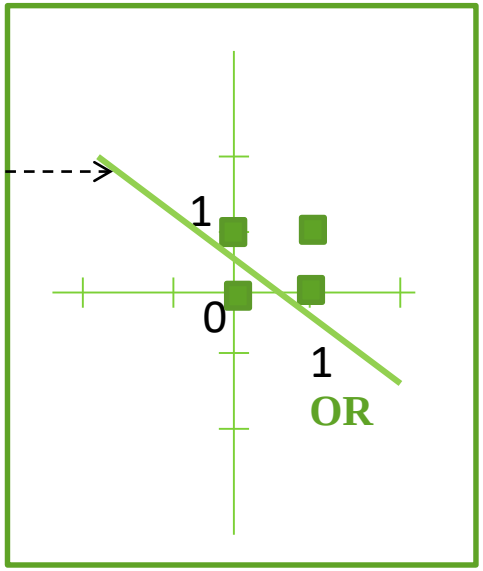


TLUs and Linear Separability – OR

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



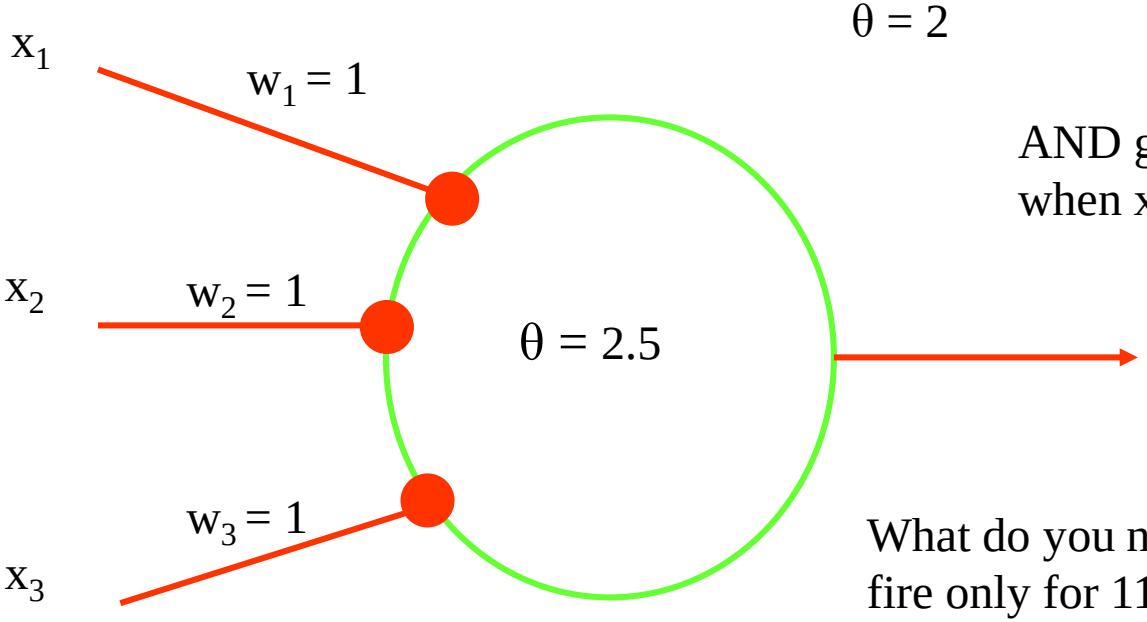


Input space

AND Gate > 2 inputs

TLUs are similar to the threshold neuron model, except that ***TLUs only accept binary inputs*** (0 or 1).

Example: AND Gate with $\theta = 1.5$



What would be θ =? if the AND gate should fire when any two inputs are 1?

$\theta = 2$

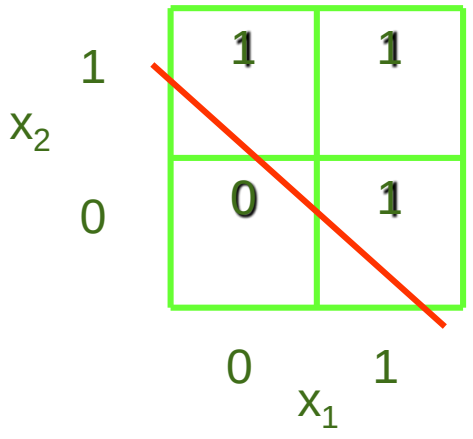
AND gate that fires when $x_1x_2x_3 = 111$

What do you need to change if it should fire only for 110?

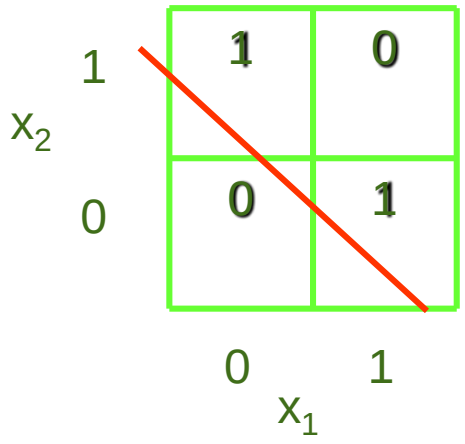
$w_3 = -1, \quad \theta = 1.5$

What if the data are not linearly separable?

- A function $f:\{0, 1\}^n \rightarrow \{0, 1\}$ 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 (two dimensions):



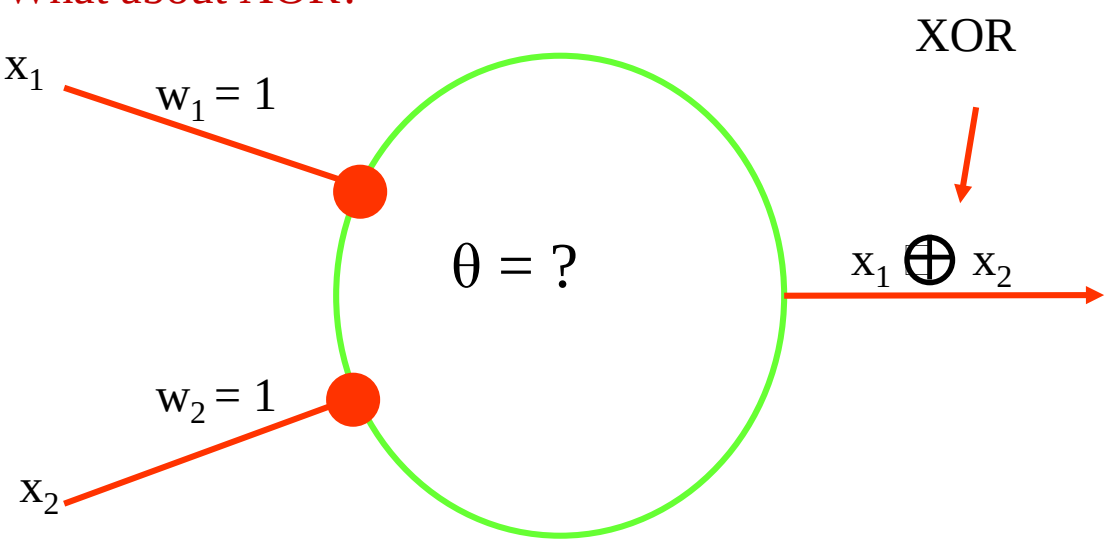
linearly separable (OR)



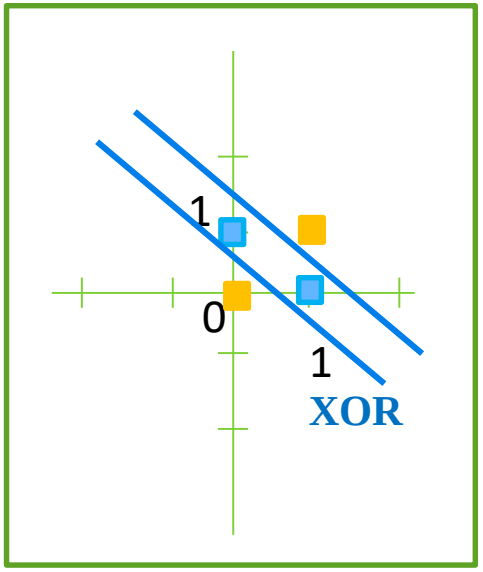
linearly inseparable (XOR)

TLUs cannot implement XOR

What about XOR?



Each point is represented by x_1x_2



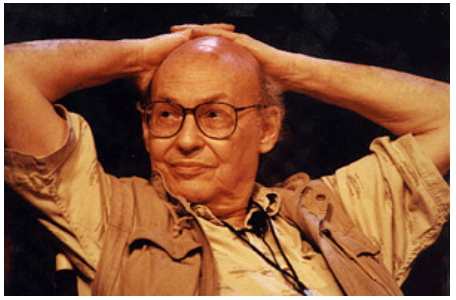
XOR Gate: 00,11=0 and 10,01=1

NOT linearly separable!!!

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.



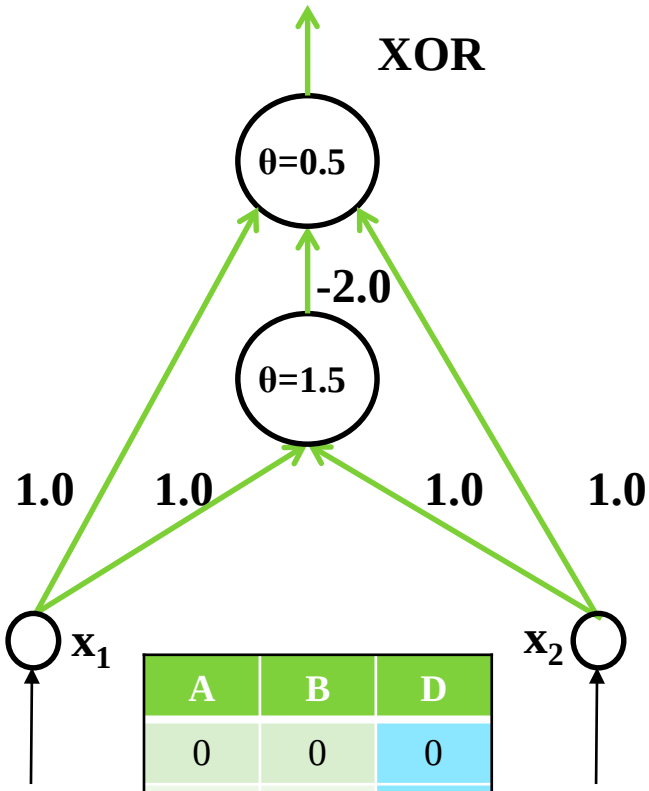
Minsky

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**.

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

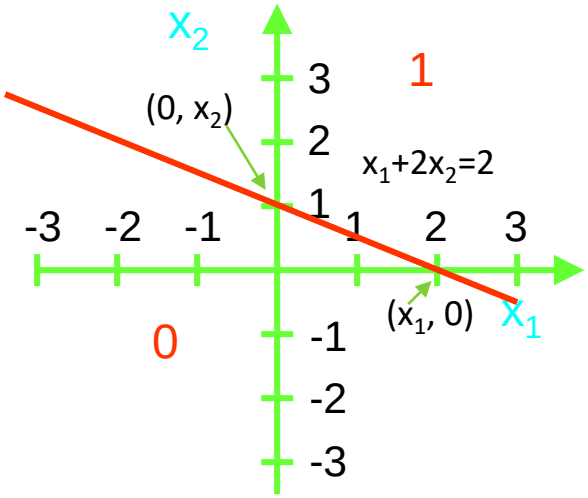
$$= 0, \quad \text{otherwise}$$

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 on a two dimensional space → can be written as $(y = mx + c)$ to plot the line → $x_2 = (-w_1/w_2)x_1 + \theta/w_2$

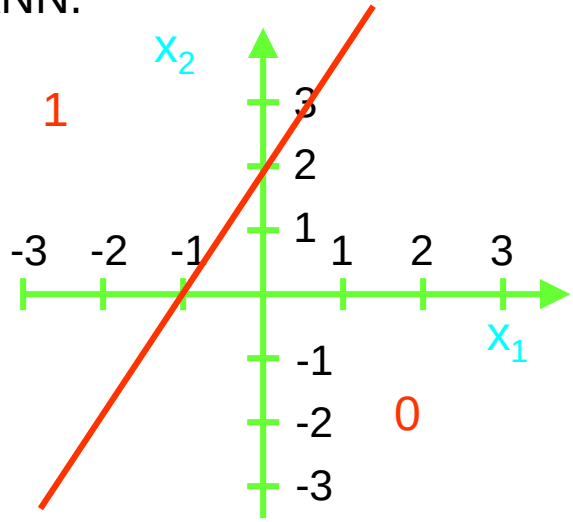
Examples – Linear Separability

Let's say we have a two-dimensional Input space (x1 and x2 and n = 2) and the following lines separate the output categories into 1 and 0.
Model the mapping using ANN.

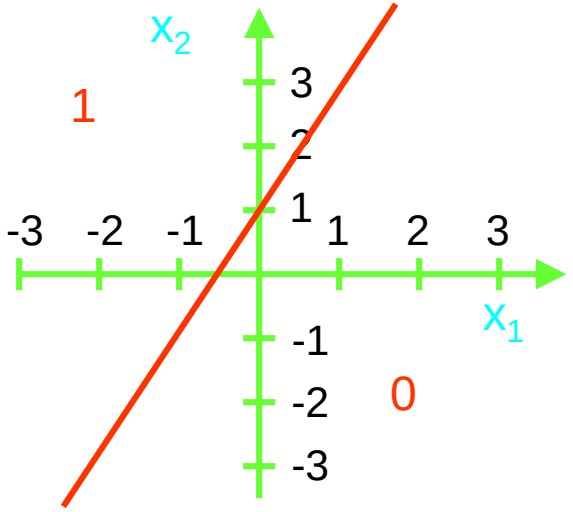
$$f(x_1, x_2, ..., x_n) = 1, \quad \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq \theta$$
$$= 0, \quad \text{otherwise}$$



For **θ = 2**, $w_1 = 1$, $w_2 = 2$



For **θ = 2**, $w_1 = -2$, $w_2 = 1$



For **θ = 1**, $w_1 = -2$, $w_2 = 1$

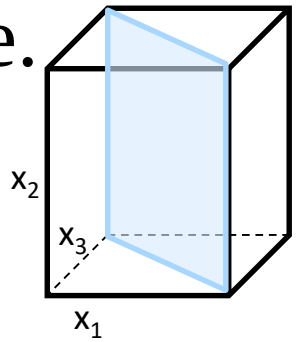
Equation of the straight line $\mathbf{w_1x_1 + w_2x_2 = \theta}$ (consider $\mathbf{x_1=x}$, $\mathbf{x_2=y}$)
To model it using an ANN, we need to find the weights for a given threshold.

Linear (cont…)

- Therefore, it means that by changing θ and the weights, the line can be aligned in any direction on the 2-dimensional surface to separate two categories of input based on the outputs 1 and 0.
- Here θ is called the bias.
- *Training a network means finding the best fitting values of θ and the weights* to categorize a given set of values correctly i.e., find the line that divide 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:

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

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

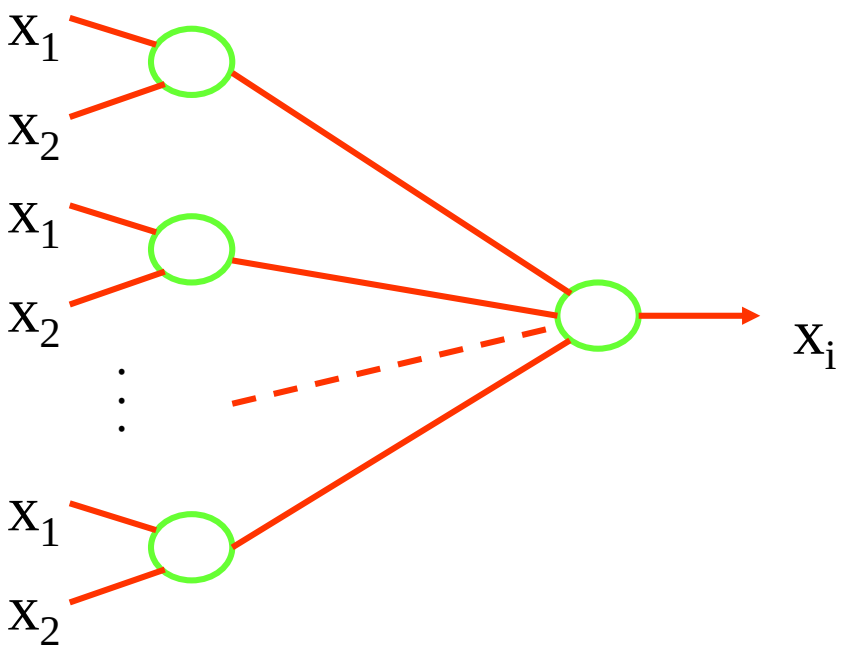
$$\text{net}_i(t) = \| w_i(t) \| \cdot \| x(t) \| \cdot \cos\alpha$$

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

||a|| = sqrt(a.a)

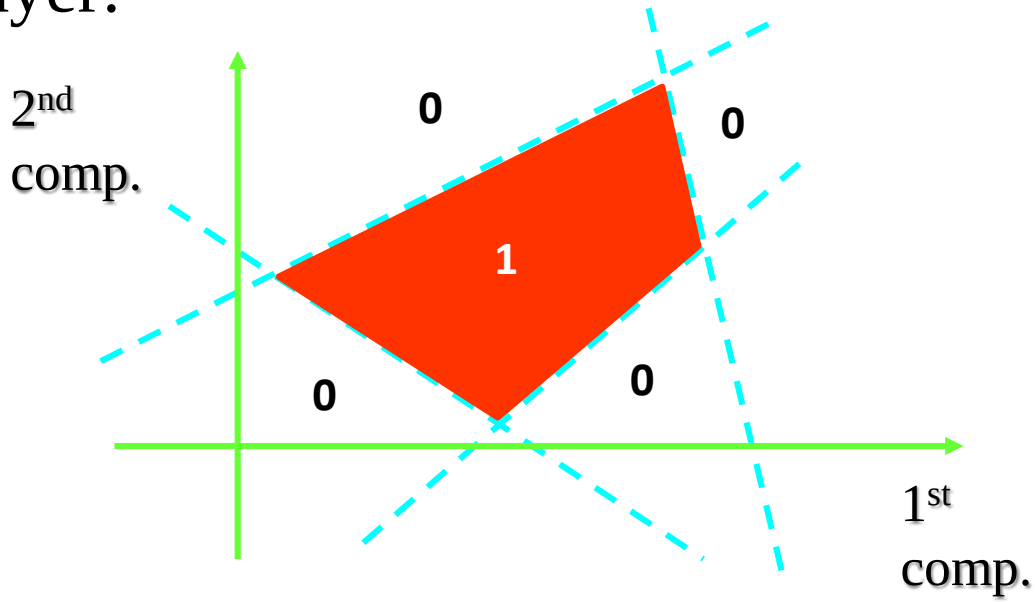
What about complex functions?

- We can combine multiple artificial neurons in multiple layers to form networks with increased capabilities.



Identify a Polygon

- Assume that the dotted lines in the diagram represent the input-dividing lines implemented by the neurons in the first layer:



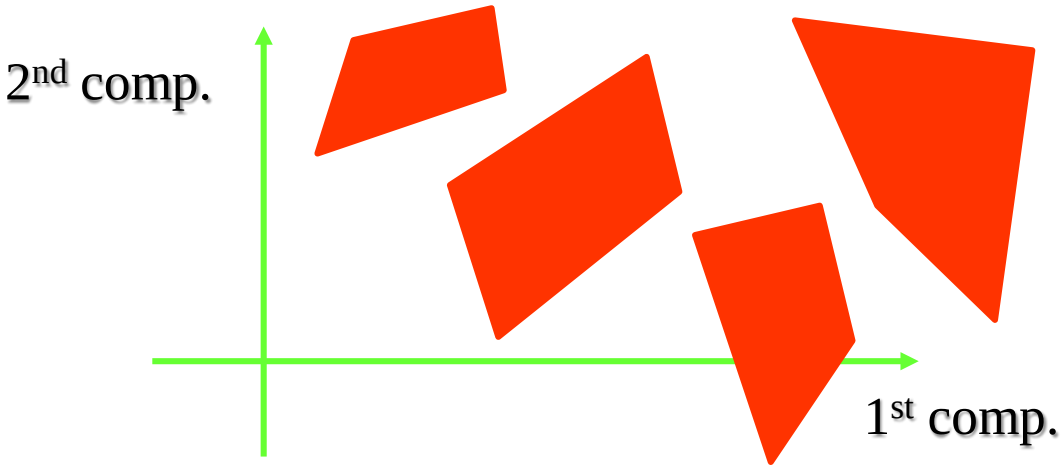
Then, for example, the second-layer neuron could output 1 **if the input is within a polygon, and 0 otherwise.**

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.

Capabilities (cont…)

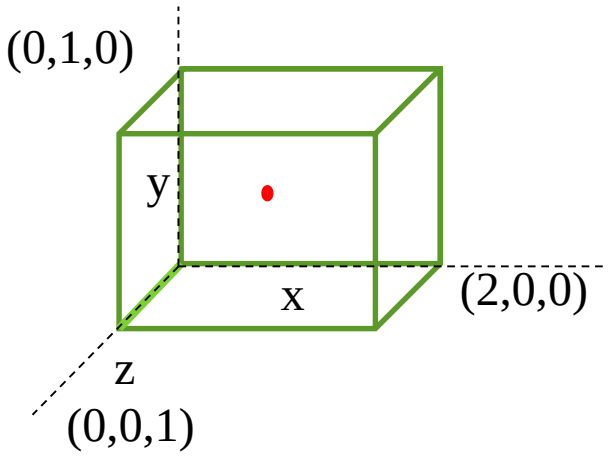
- Assume that the polygons in the diagram indicate the input regions for which each of the **second-layer** neurons yields output 1:



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

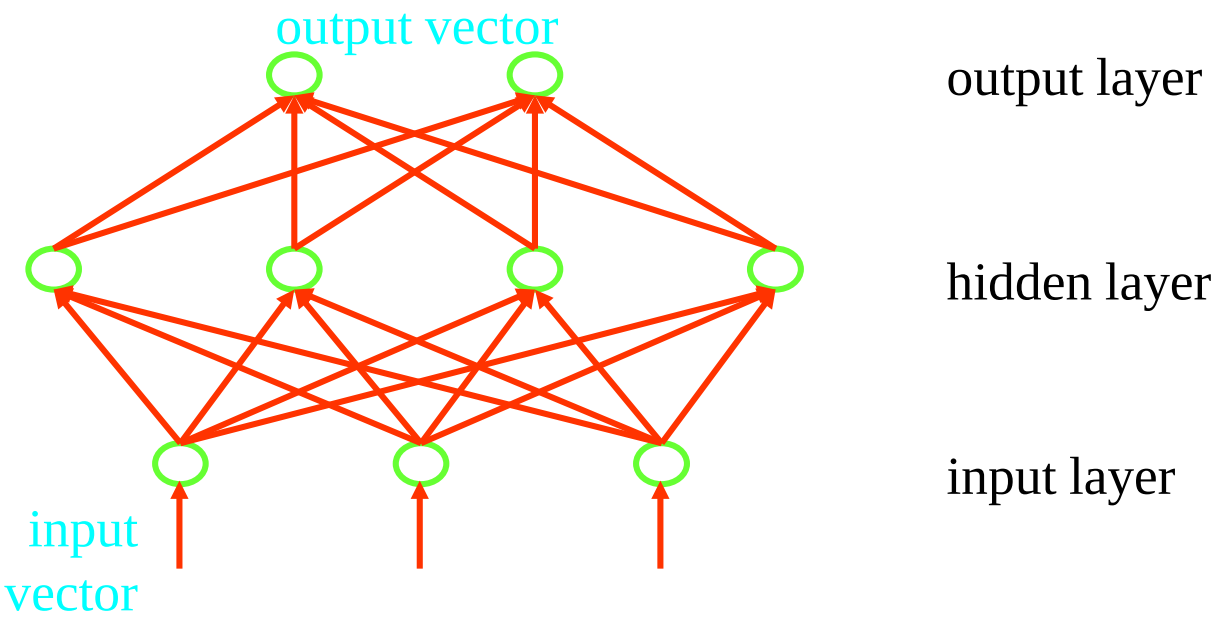
Problem

1. Draw an ANN that can detect whether a point lies on the **surface** of a box. 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.



Terminology

- Usually, we draw neural networks in such a way that the input enters at the bottom and the output is generated at the top.
- Arrows indicate the direction of data flow.
- 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.