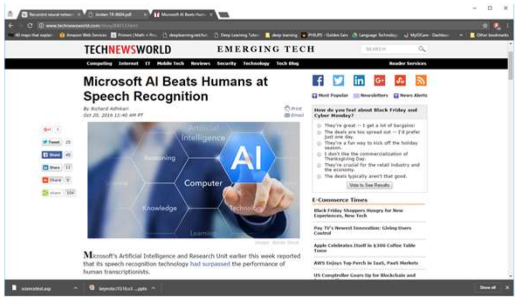
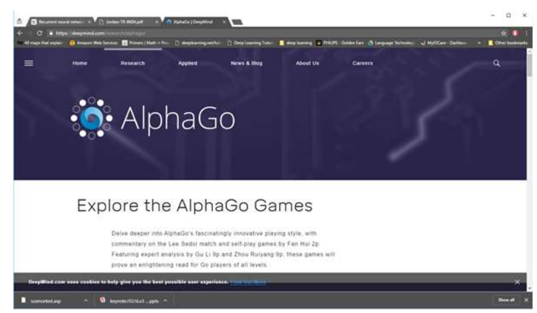
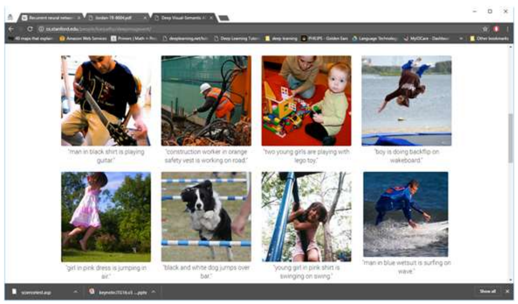
Neural Networks:
What can a network represent

Deep Learning, Spring 2021

Recap : Neural networks have taken over AI



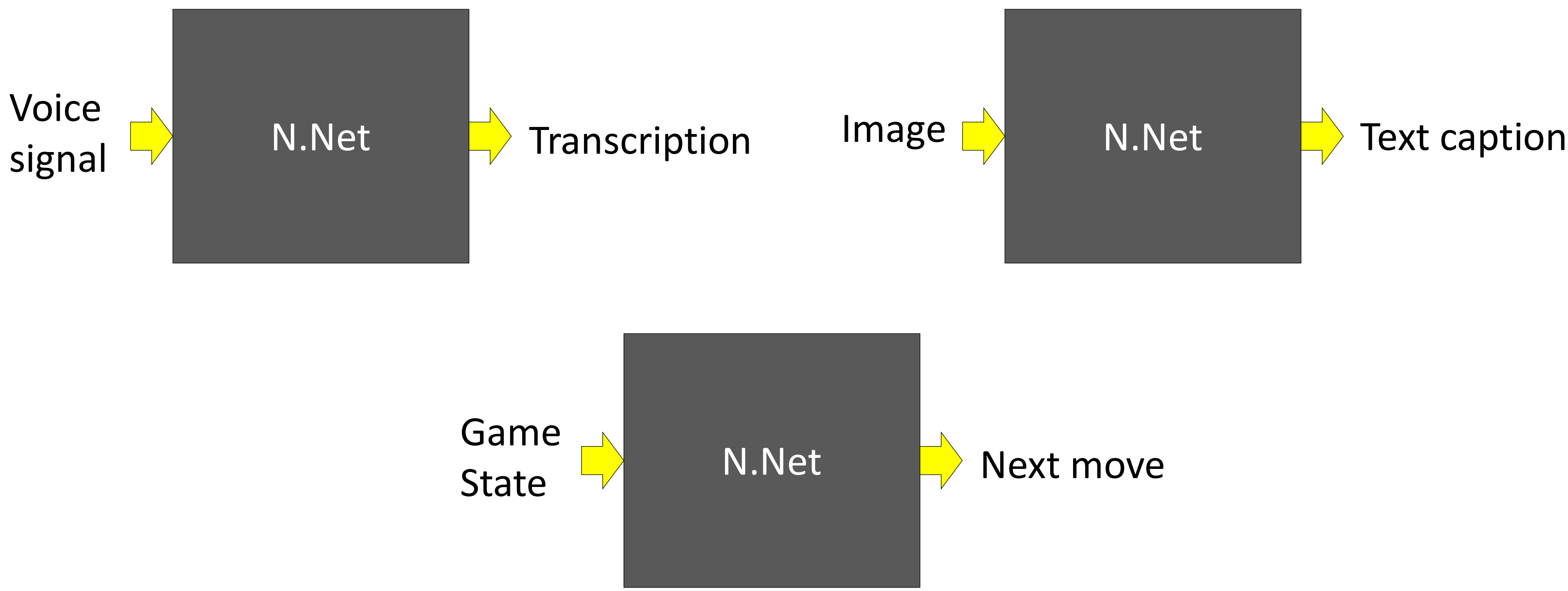




Reference Our Result

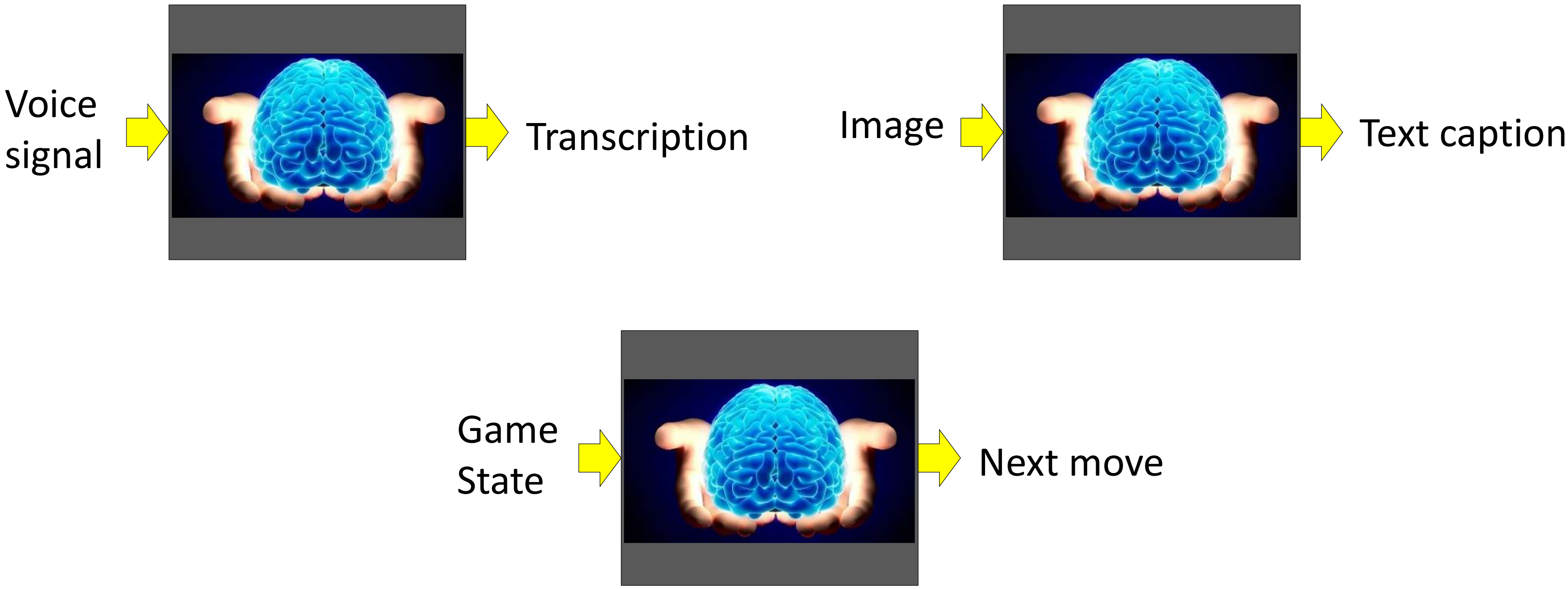
- Tasks that are made possible by NNs, aka deep learning
  - Tasks that were once assumed to be purely in the human domain of expertise

So what are neural networks??



- What are these boxes?
  - Functions that take an input and produce an output
  - What's in these functions?

The human perspective

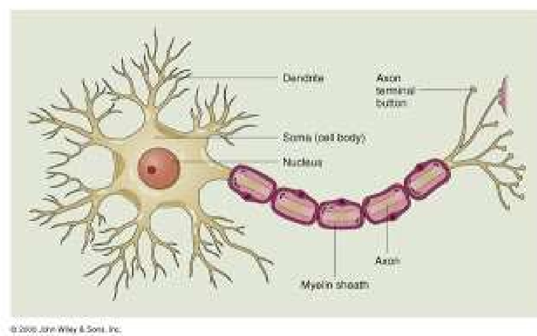


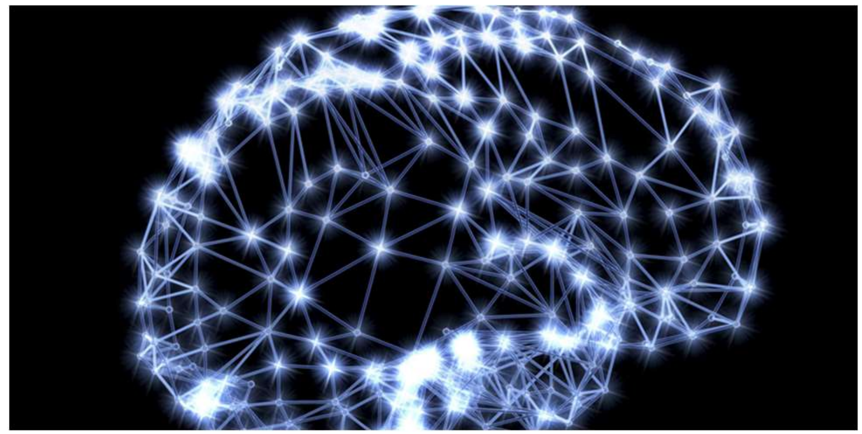
- In a human, those functions are computed by the brain…

Recap : NNets and the brain

- In their basic form, NNets mimic the networked structure in the brain

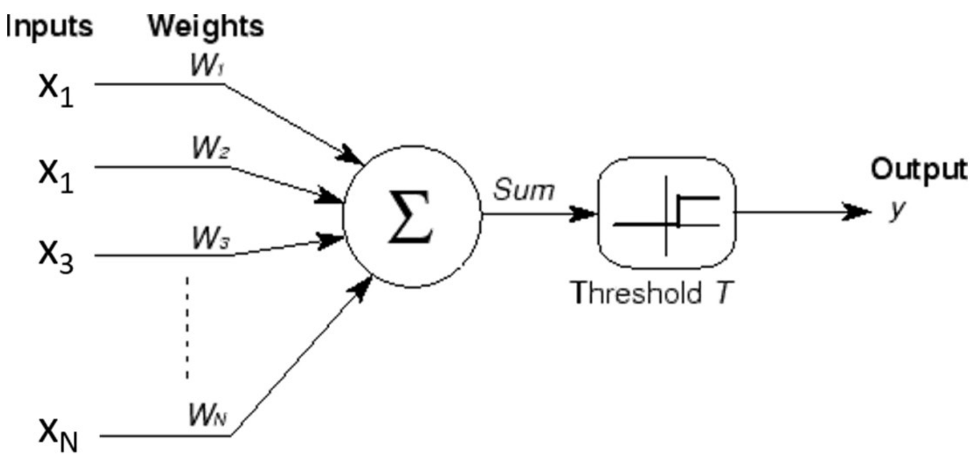
Recap : The brain

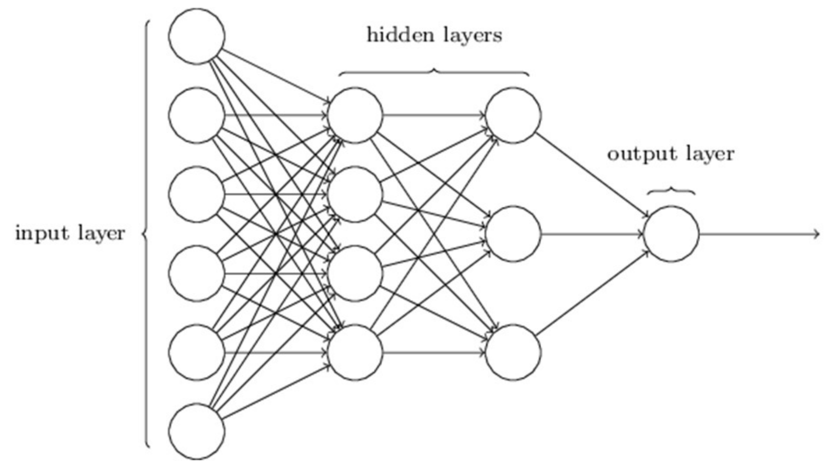


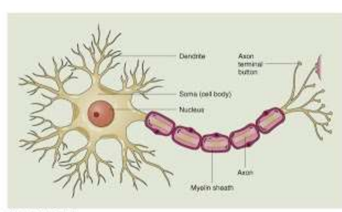


- The Brain is composed of networks of neurons

Recap : Nnets and the brain

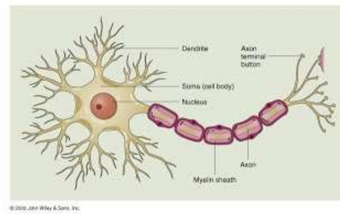


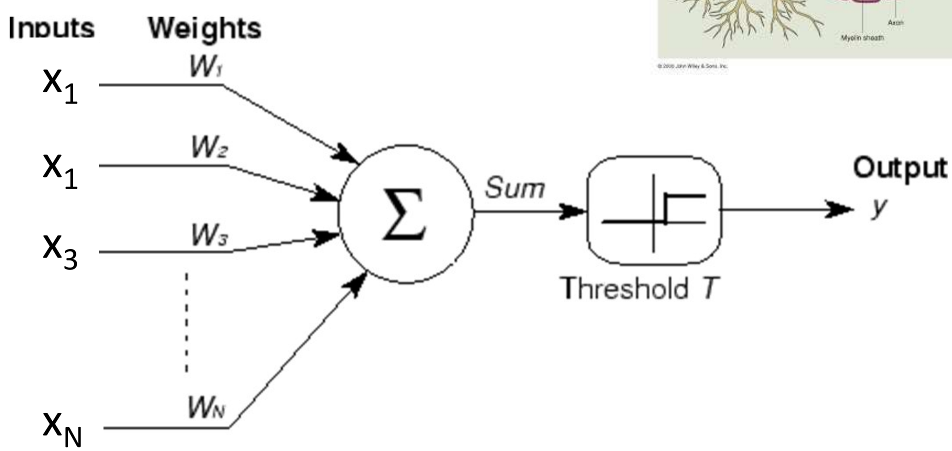




- Neural nets are composed of networks of computational models of neurons called perceptrons

Recap: the perceptron

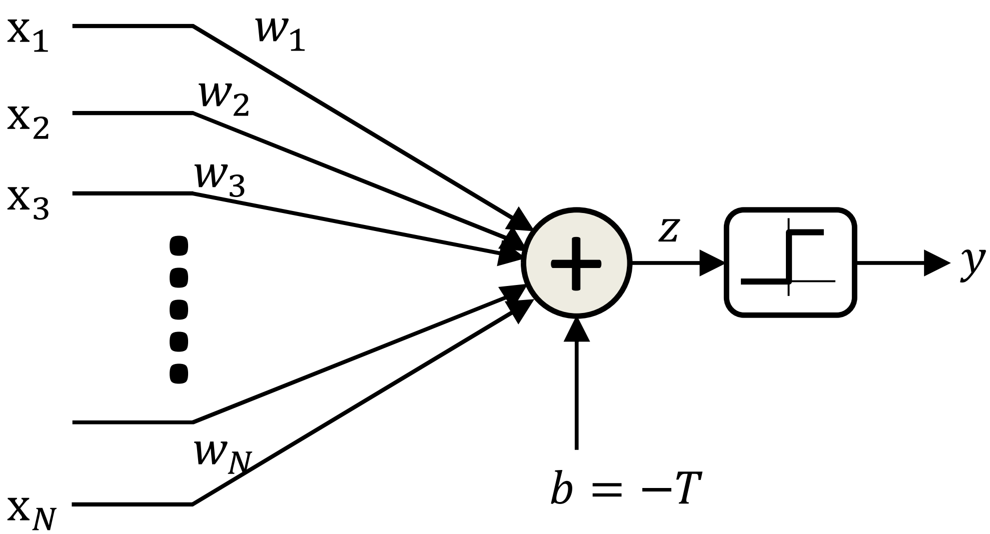




$$y = \begin{cases} 1 \ if \ \sum_i w_i x_i \geq T \\ 0 \ else \end{cases}$$

- A threshold unit
  - "Fires" if the weighted sum of inputs exceeds a threshold
  - Electrical engineers will call this a ***threshold gate***
    - A basic unit of Boolean circuits

A better figure

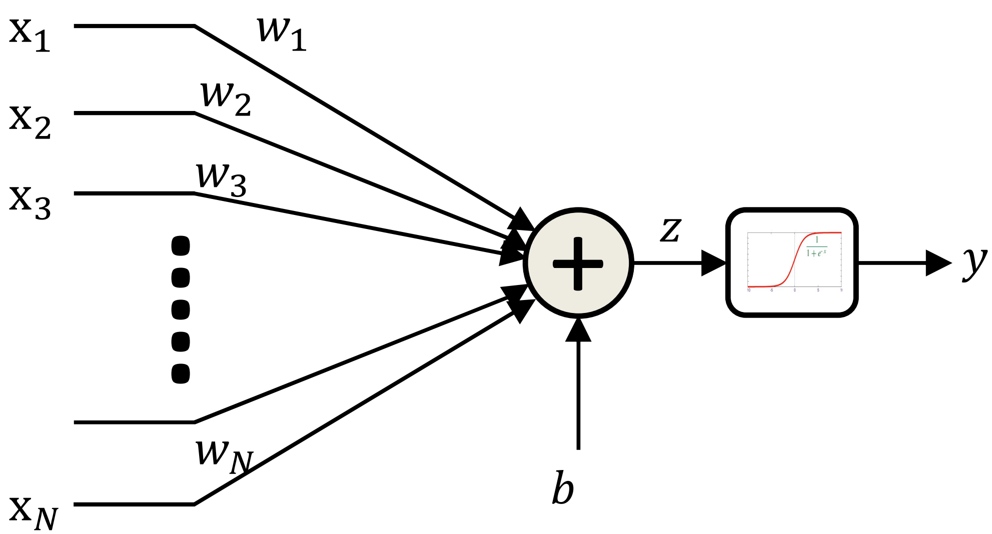


$$z = \sum_i w_i x_i + b$$

$$y = \begin{cases} 1 \ if \ z \geq 0 \\ 0 \ else \end{cases}$$

- A threshold unit
  - "Fires" if the affine function of inputs is positive
    - The bias is the negative of the threshold T in the previous slide

The "soft" perceptron (logistic)

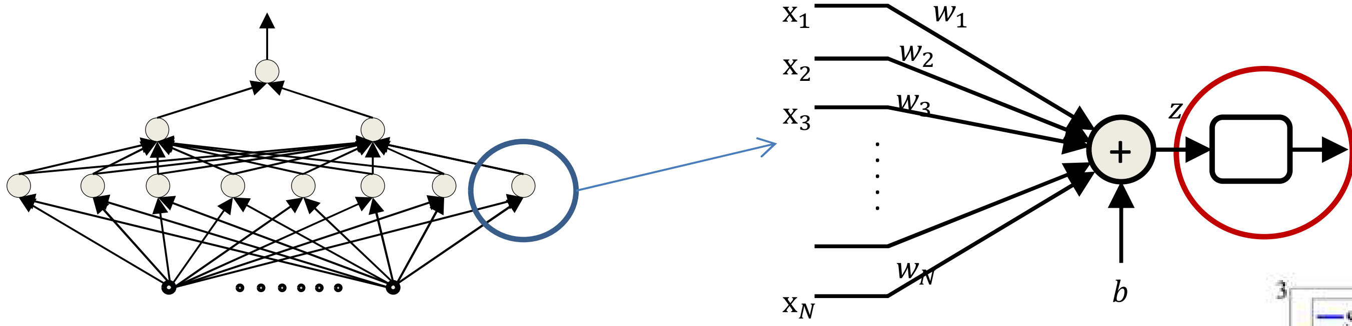


$$z = \sum_i w_i x_i + b$$

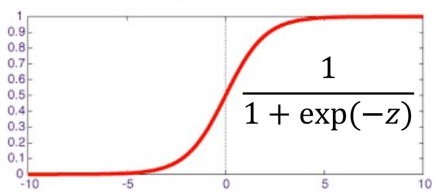
$$y = \frac{1}{1 + exp(-z)}$$

- A "squashing" function instead of a threshold at the output
  - The sigmoid "activation" replaces the threshold
    - **Activation:** The function that acts on the weighted combination of inputs (and threshold)

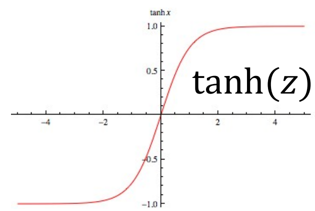
Other “activations”

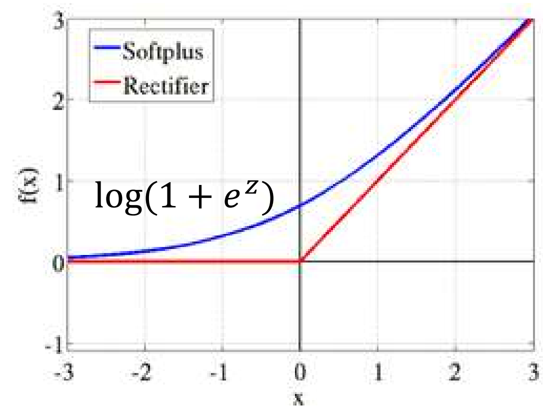


sigmoid



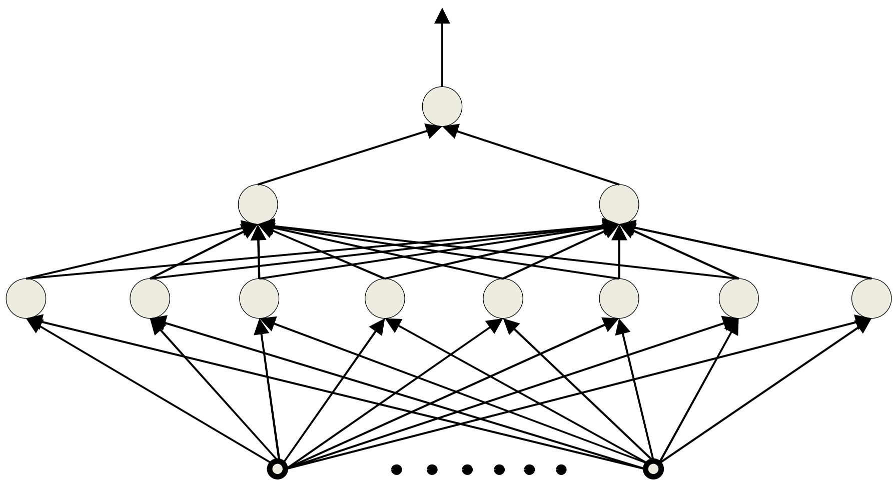
tanh

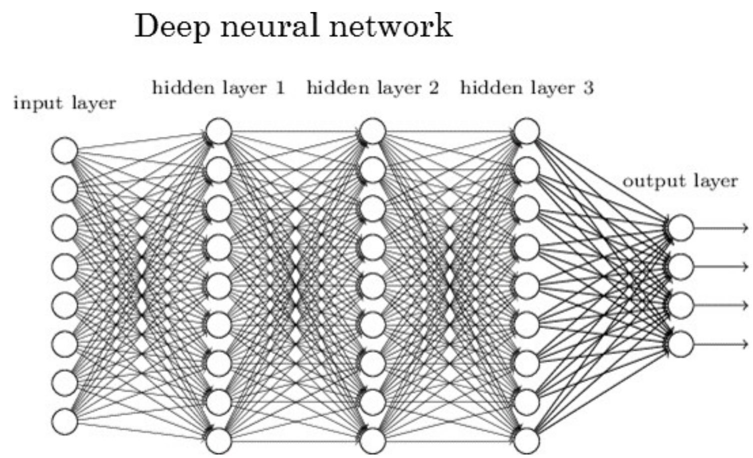




- Does not always have to be a squashing function
  - We will hear more about activations later
- We will continue to assume a “threshold” activation in this lecture

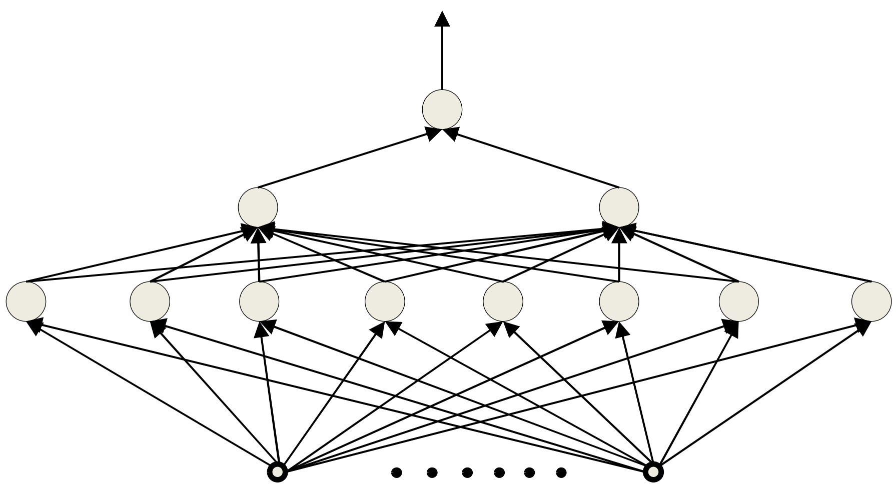
The *multi-layer* perceptron

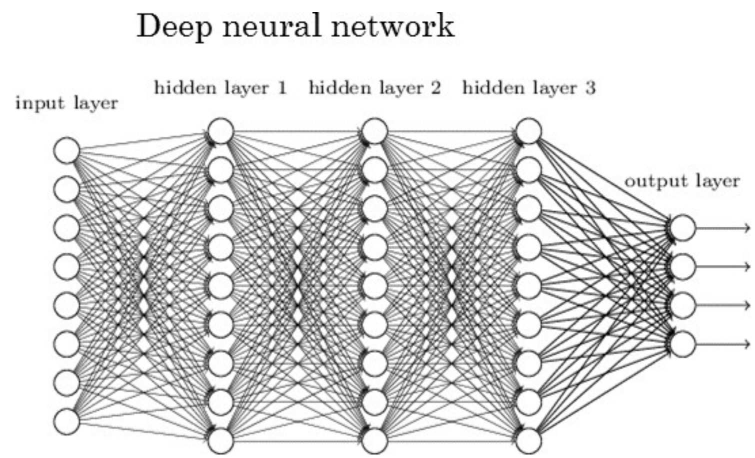




- A network of perceptrons
  - Perceptrons "feed" other perceptrons
  - We give you the "formal" definition of a layer later

Defining "depth"

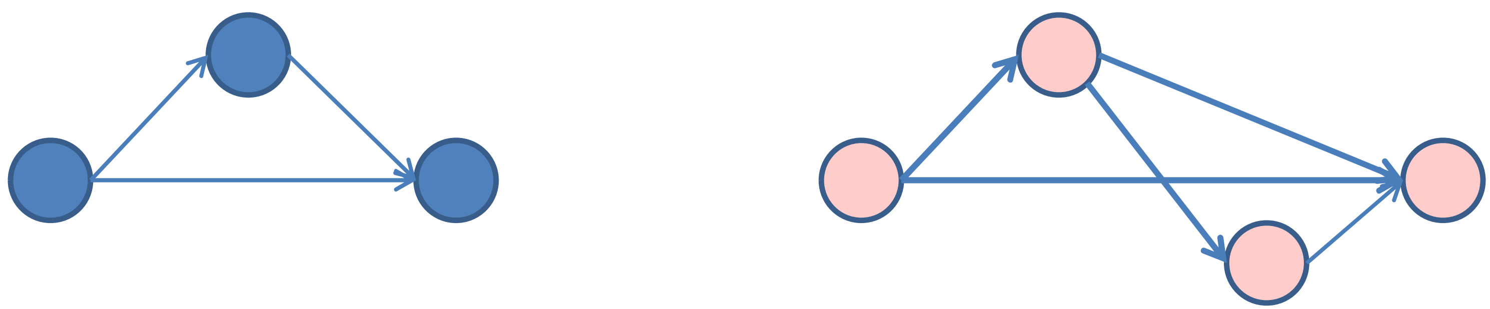




- What is a "deep" network

Deep Structures

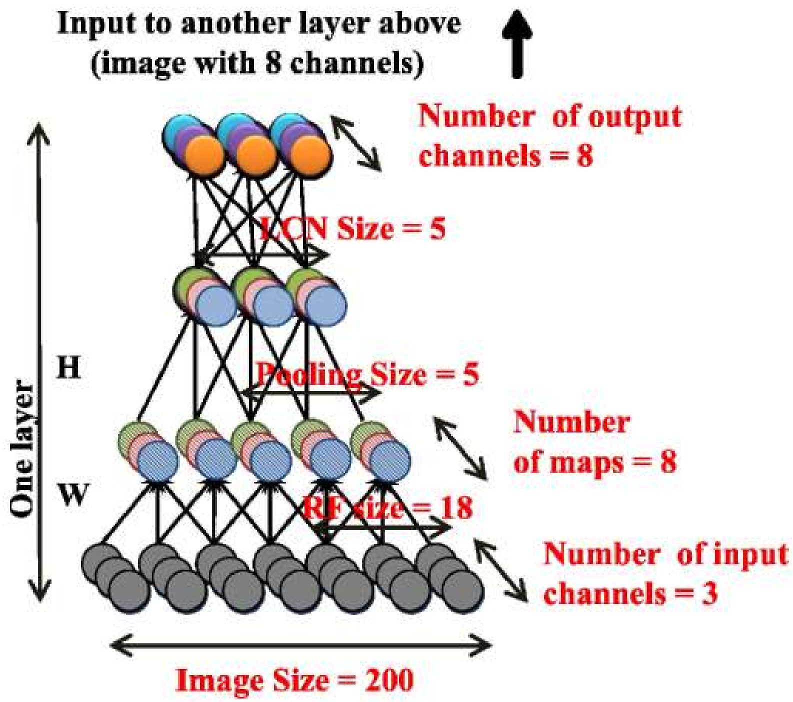
- In any directed network of computational elements with input source nodes and output sink nodes, "depth" is the length of the longest path from a source to a sink
  - A "source" node in a directed graph is a node that has only outgoing edges
  - A "sink" node is a node that has only incoming edges

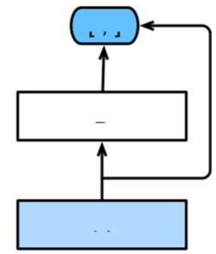


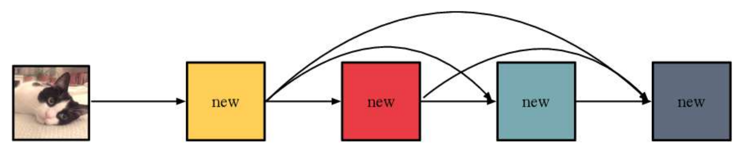
- Left: Depth = 2.   Right: Depth = 3

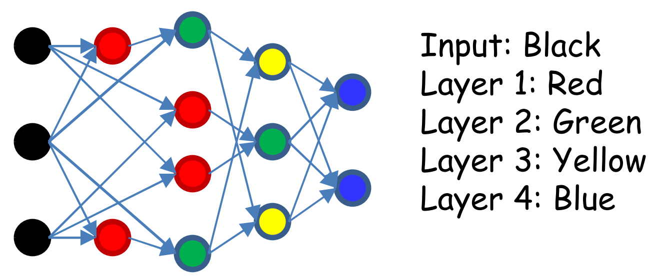
Deep Structures

- *Layered* deep structure
  - *The input is the "source",*
  - *The output nodes are "sinks"*



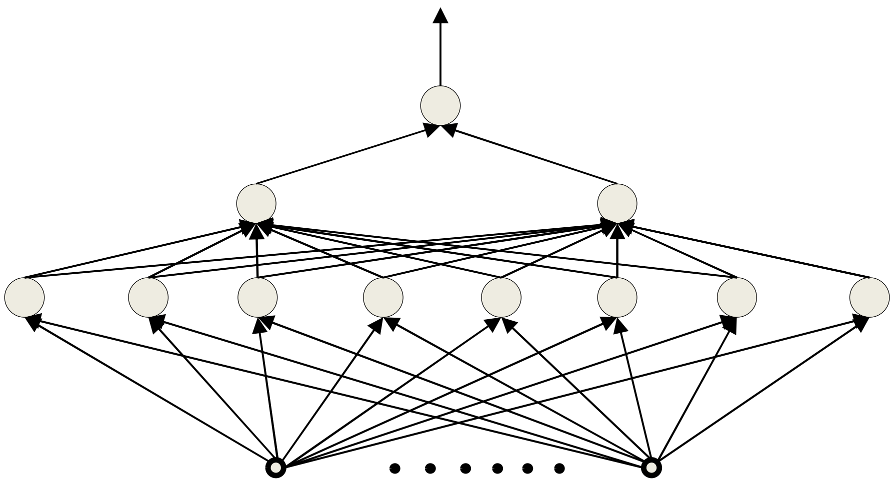


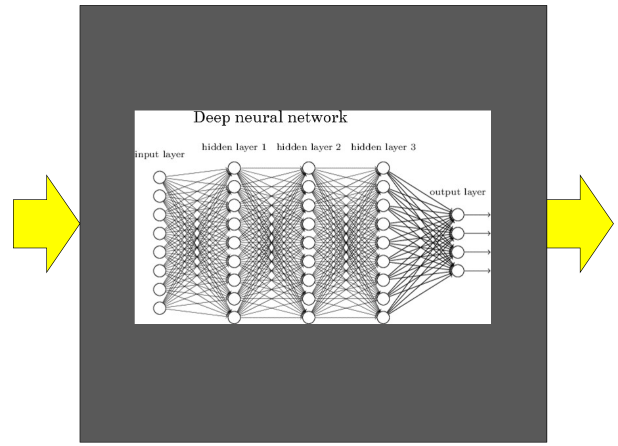




- "Deep" → Depth greater than 2
- "Depth" of a layer – the depth of the neurons in the layer w.r.t. input

The multi-layer perceptron

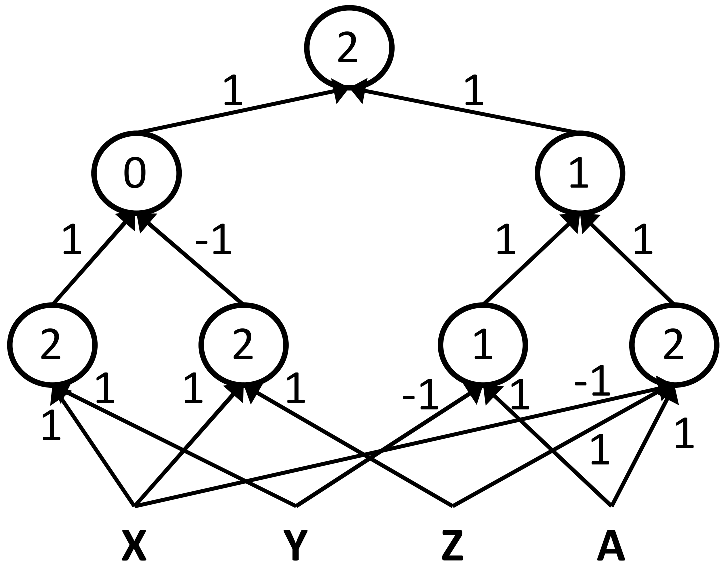


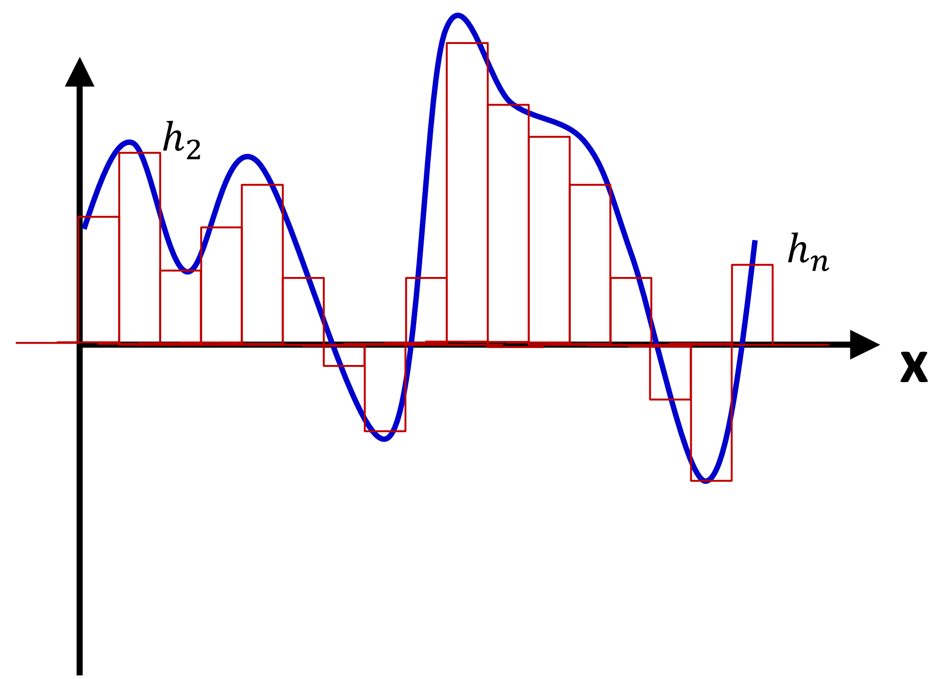


- Inputs are real or Boolean stimuli
- Outputs are real or Boolean values
  - Can have multiple outputs for a single input
- **What can this network compute?**
  - **What kinds of input/output relationships can it model?**

MLPs approximate functions

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





- MLPs can compose Boolean functions
- MLPs can compose real-valued functions
- What are the limitations?

Today

- Multi-layer Perceptrons as universal Boolean functions
  - The need for depth
- MLPs as universal classifiers
  - The need for depth
- MLPs as universal approximators
- A discussion of optimal depth and width
- Brief segue: RBF networks

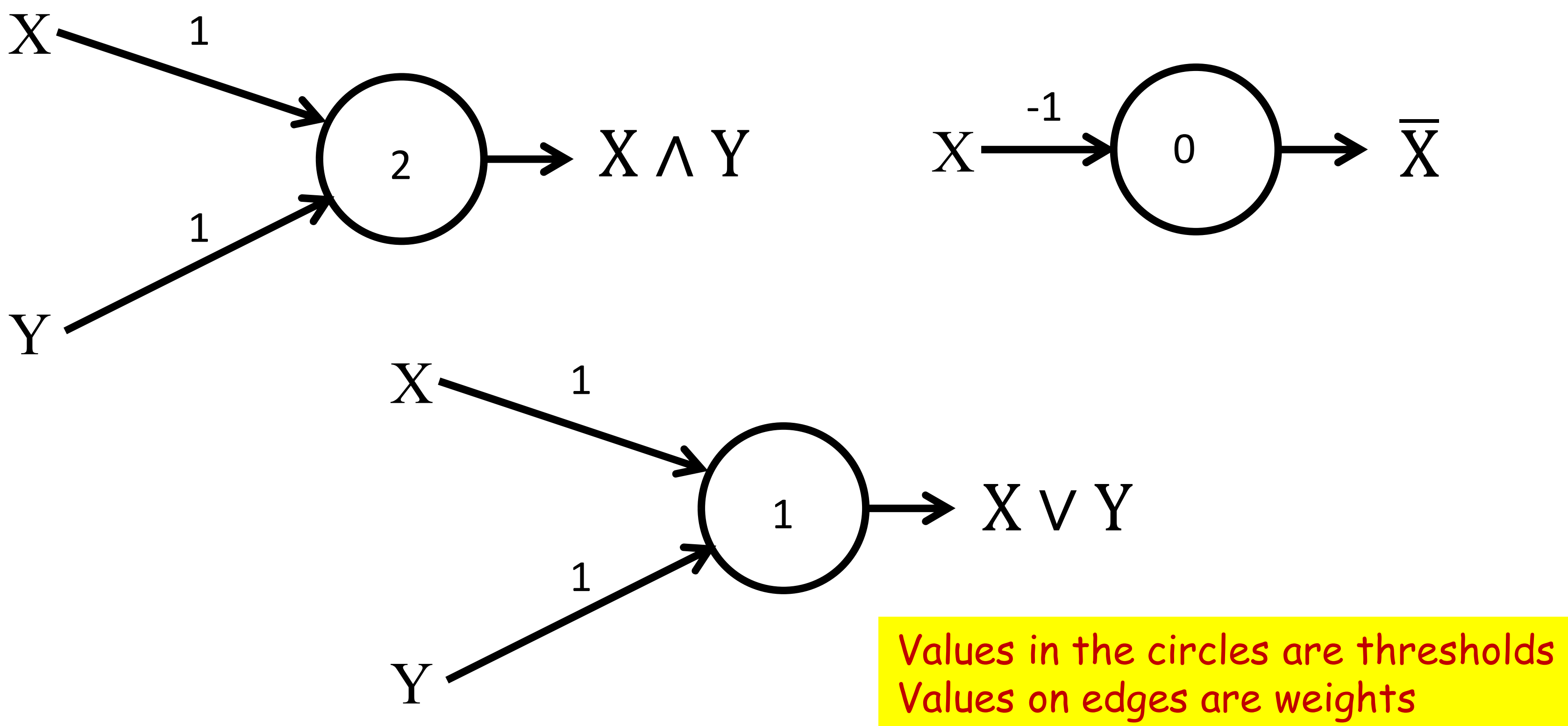
Today

- Multi-layer Perceptrons as universal Boolean functions
  - The need for depth
- MLPs as universal classifiers
  - The need for depth
- MLPs as universal approximators
- A discussion of optimal depth and width
- Brief segue: RBF networks

The MLP as a Boolean function

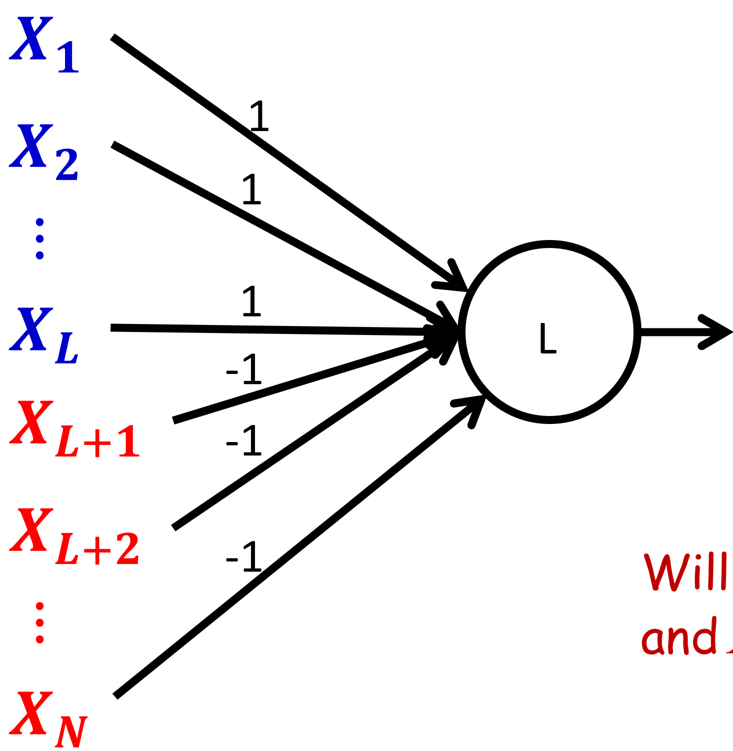
- How well do MLPs model Boolean functions?

The perceptron as a Boolean gate



- A perceptron can model any simple binary Boolean gate

Perceptron as a Boolean gate

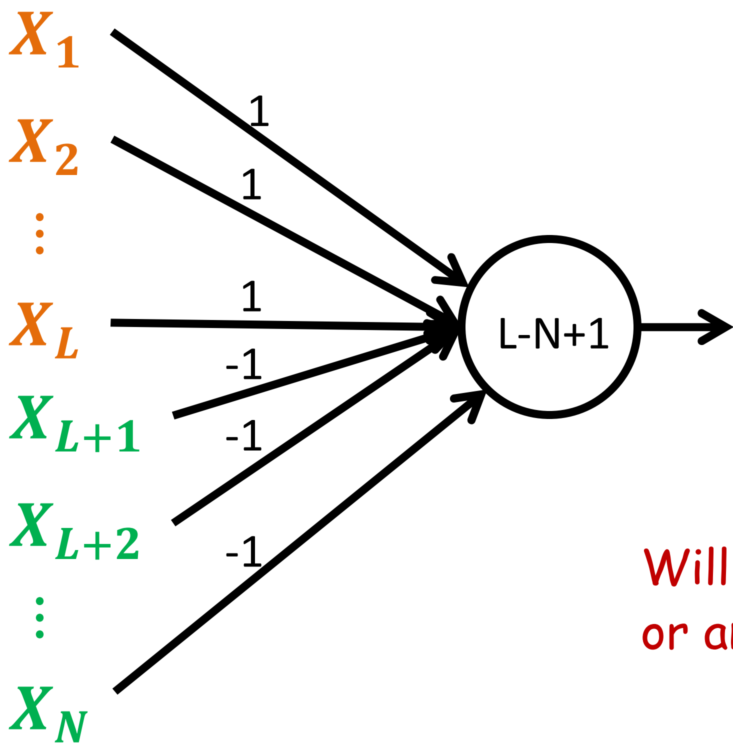


$$\left(\bigwedge_{i=1}^{L} X_i\right) \wedge \left(\bigwedge_{i=L+1}^{N} \bar{X}_i\right)$$

Will fire only if $X_1 .. X_L$ are all 1 and $X_{L+1} .. X_N$ are all 0

- The universal AND gate
  - AND any number of inputs
    - Any subset of who may be negated

Perceptron as a Boolean gate

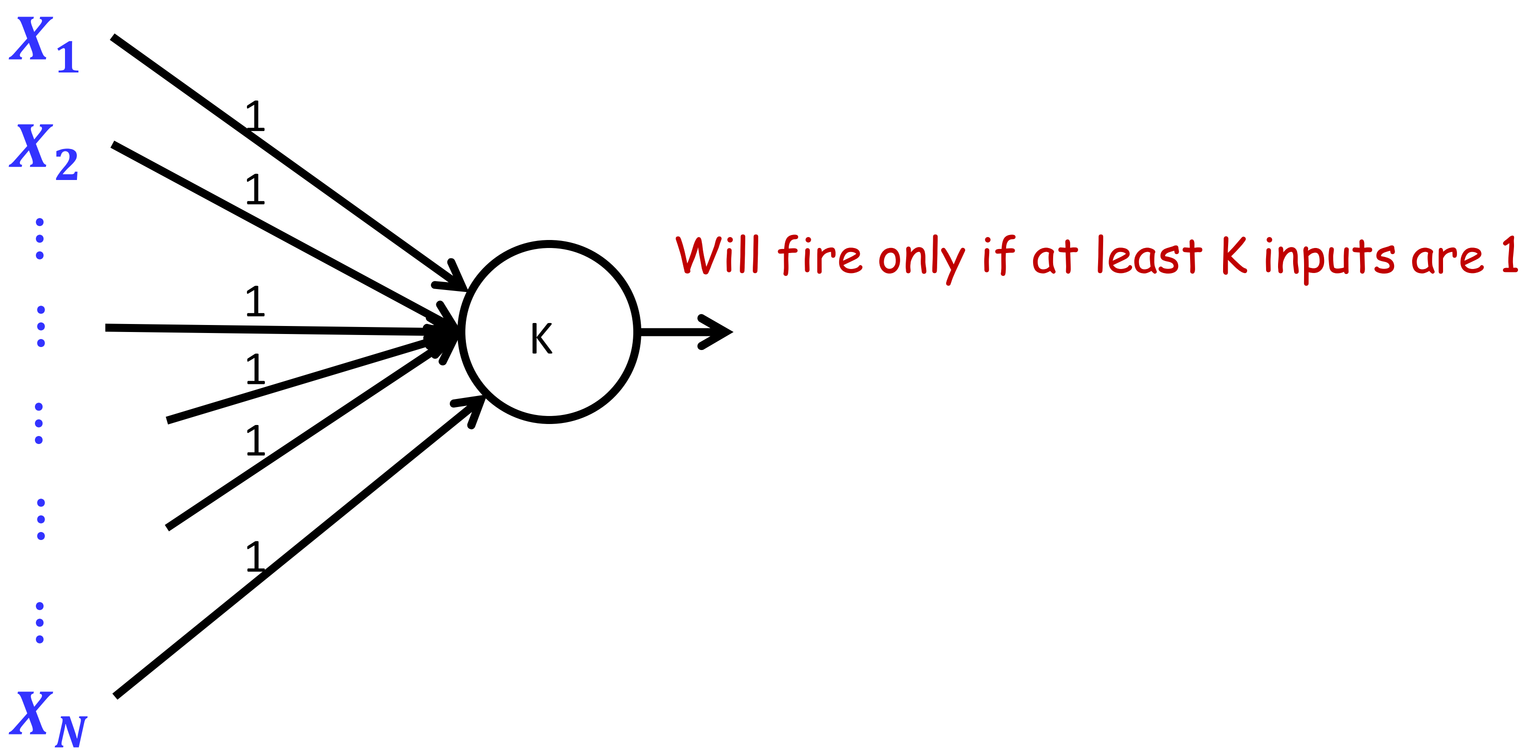


$$\left(\bigvee_{i=1}^{L} X_i\right) \vee \left(\bigvee_{i=L+1}^{N} \bar{X}_i\right)$$

Will fire only if any of X_1 .. X_L are 1 or any of X_{L+1} .. X_N are 0

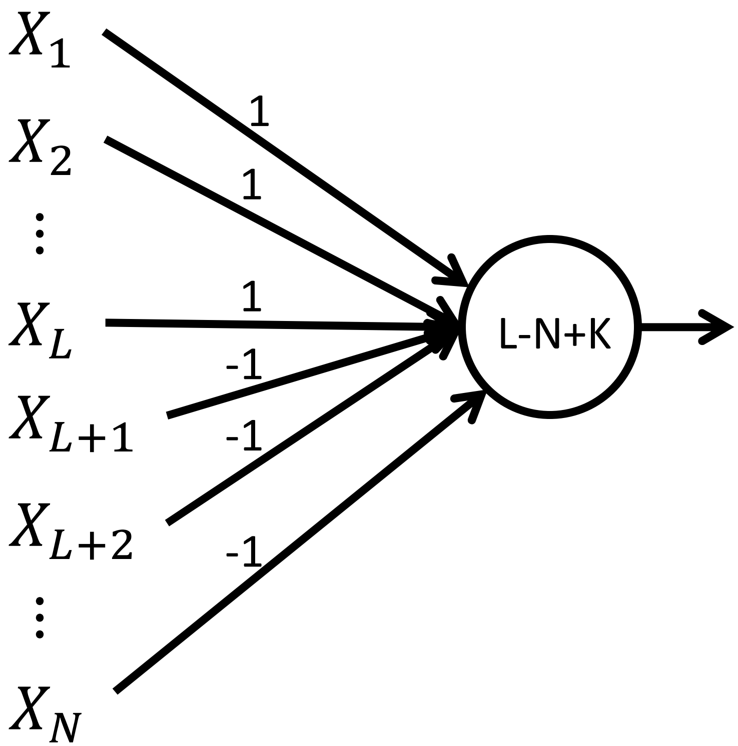
- The universal OR gate
  - OR any number of inputs
    - Any subset of who may be negated

Perceptron as a Boolean Gate



- Generalized *majority* gate
  - Fire if at least *K* inputs are of the desired polarity

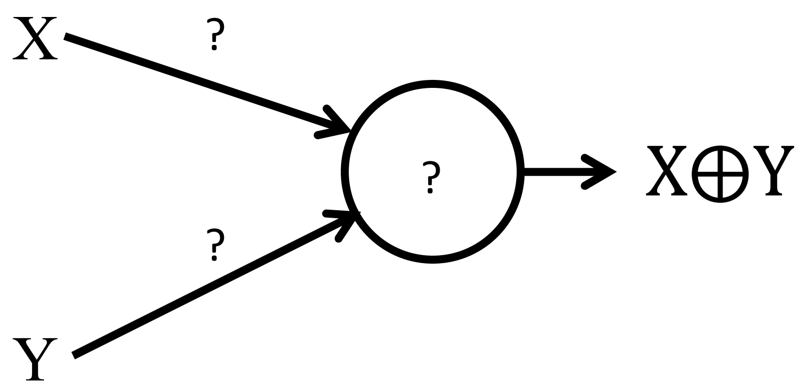
Perceptron as a Boolean Gate



Will fire only if the total number of of X_1 .. X_L that are 1 and X_{L+1} .. X_N that are 0 is at least K

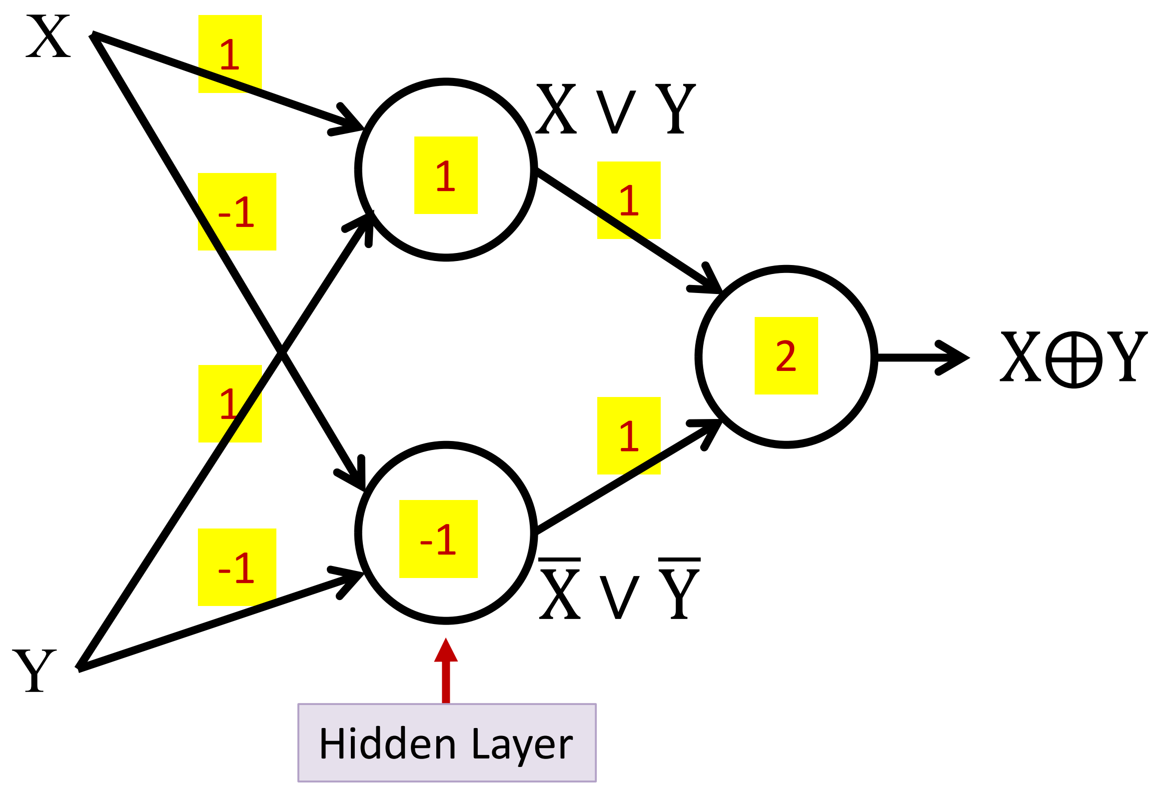
- Generalized *majority* gate
  - Fire if at least *K* inputs are of the desired polarity

The perceptron is not enough



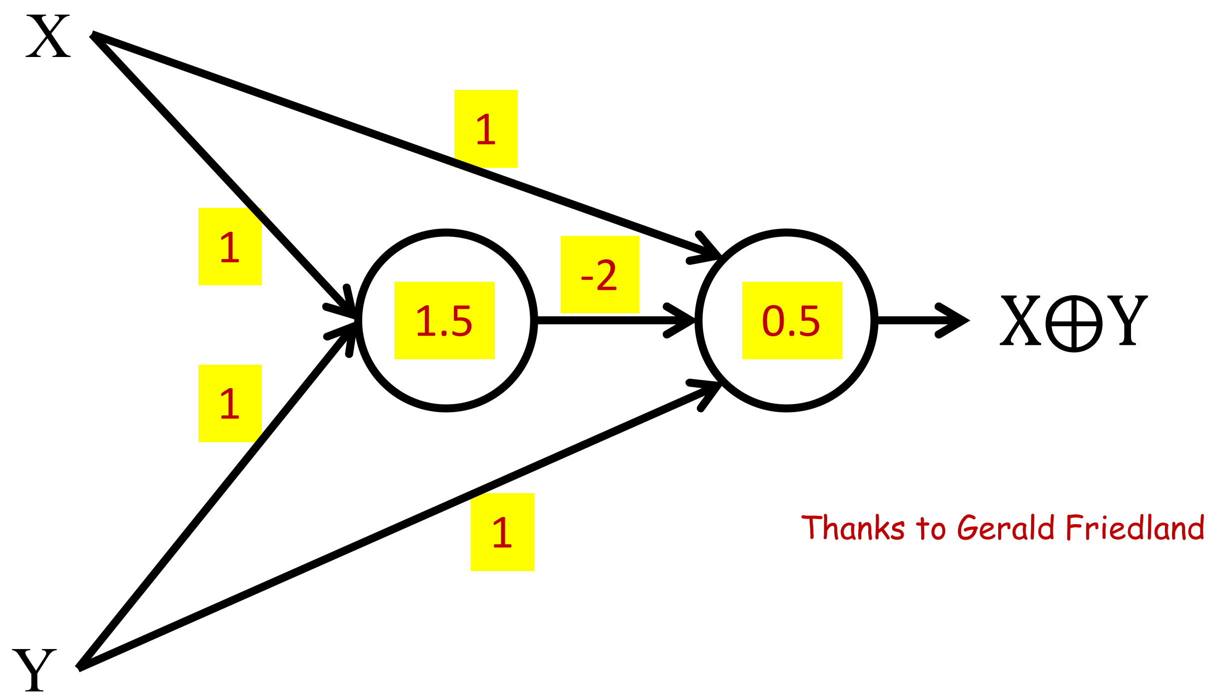
- Cannot compute an XOR

Multi-layer perceptron



- MLPs can compute the XOR

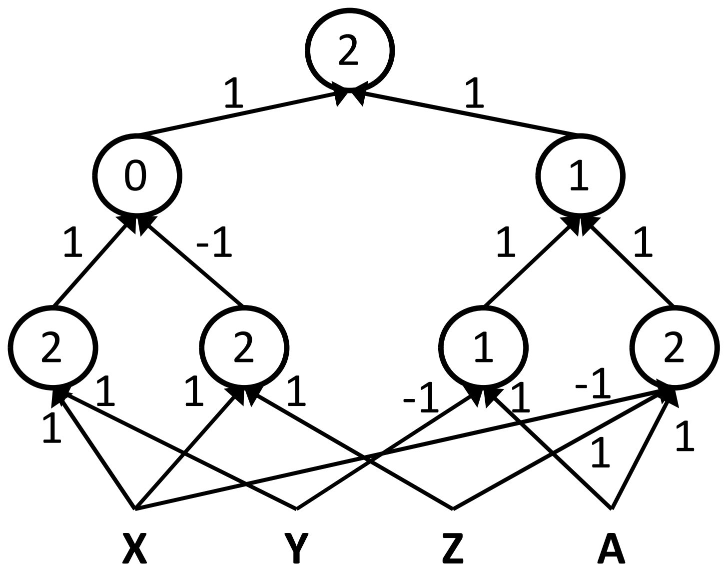
Multi-layer perceptron XOR



- With 2 neurons
  - 5 weights and two thresholds

Multi-layer perceptron

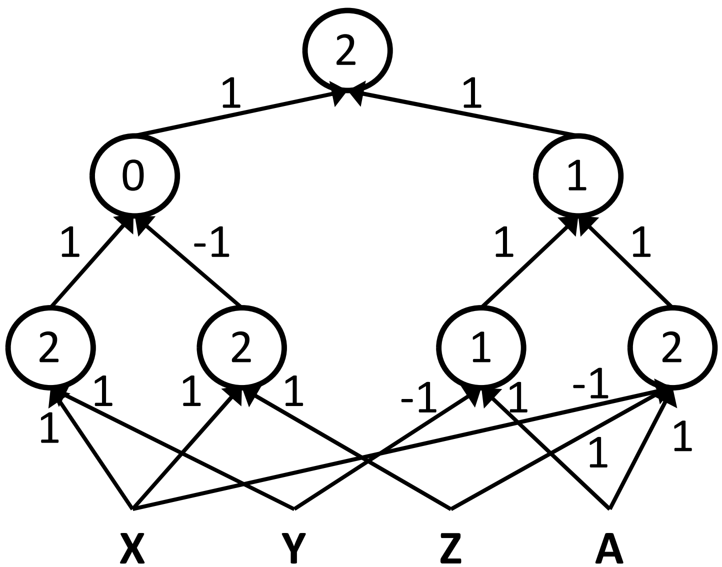
$$((A\&\bar{X}\&Z)|(A\&\bar{Y}))\&((X\ \&\ Y)|\overline{(X\&Z)})$$

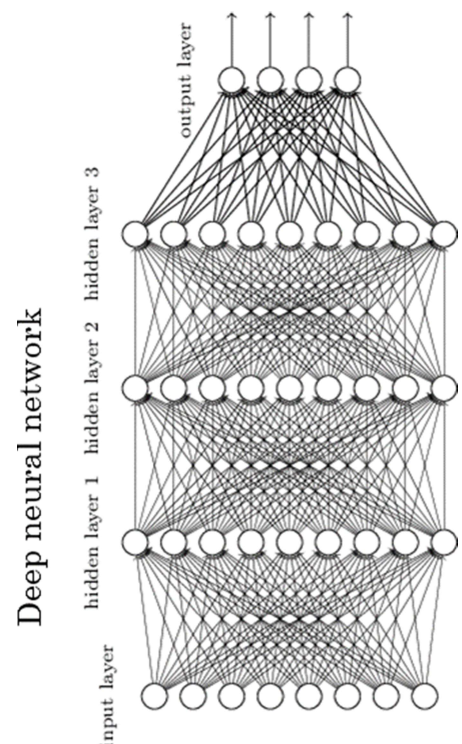


- MLPs can compute more complex Boolean functions
- MLPs can compute *any* Boolean function
  – Since they can emulate individual gates
- **MLPs are *universal Boolean functions***

MLP as Boolean Functions

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





- MLPs are universal Boolean functions
  - Any function over any number of inputs and any number of outputs
- But how many "layers" will they need?

How many layers for a Boolean MLP?

Truth table shows *all* input combinations for which output is 1

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

- ***A Boolean function is just a truth table***

How many layers for a Boolean MLP?

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Truth table shows *all* input combinations for which output is 1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$

- Expressed in disjunctive normal form

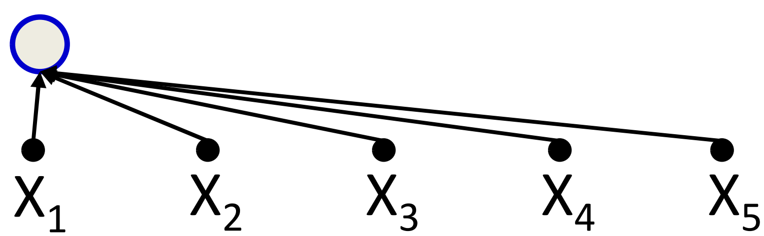
How many layers for a Boolean MLP?

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Truth table shows *all* input combinations for which output is 1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



- Expressed in disjunctive normal form

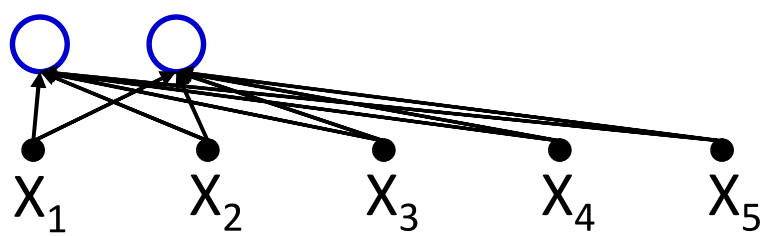
How many layers for a Boolean MLP?

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Truth table shows *all* input combinations for which output is 1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



- Expressed in disjunctive normal form

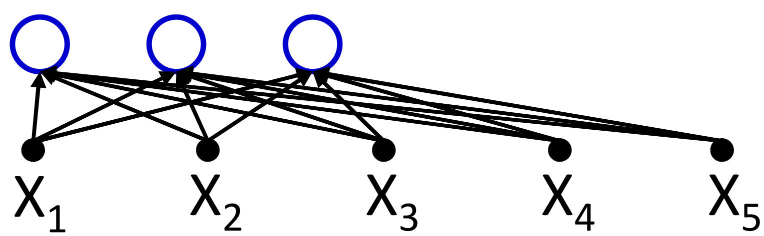
How many layers for a Boolean MLP?

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Truth table shows *all* input combinations for which output is 1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



- Expressed in disjunctive normal form

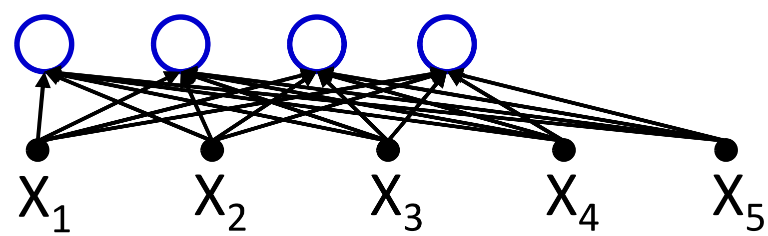
How many layers for a Boolean MLP?

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Truth table shows *all* input combinations for which output is 1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



- Expressed in disjunctive normal form

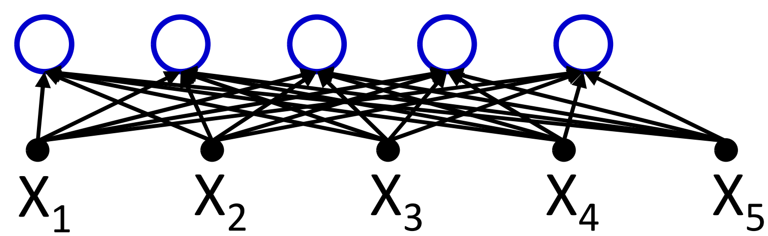
How many layers for a Boolean MLP?

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Truth table shows *all* input combinations for which output is 1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



- Expressed in disjunctive normal form

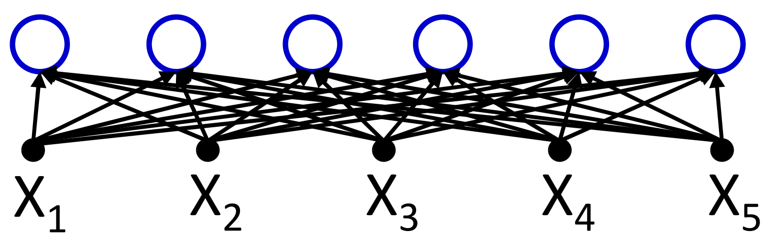
How many layers for a Boolean MLP?

Truth table shows *all* input combinations for which output is 1

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



- Expressed in disjunctive normal form

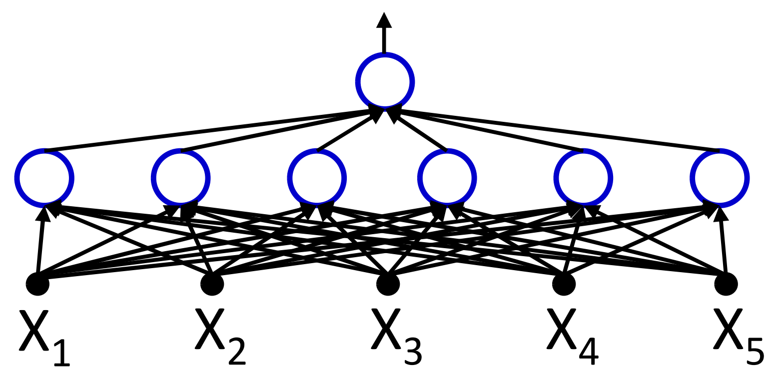
How many layers for a Boolean MLP?

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Truth table shows *all* input combinations for which output is 1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



- Expressed in disjunctive normal form

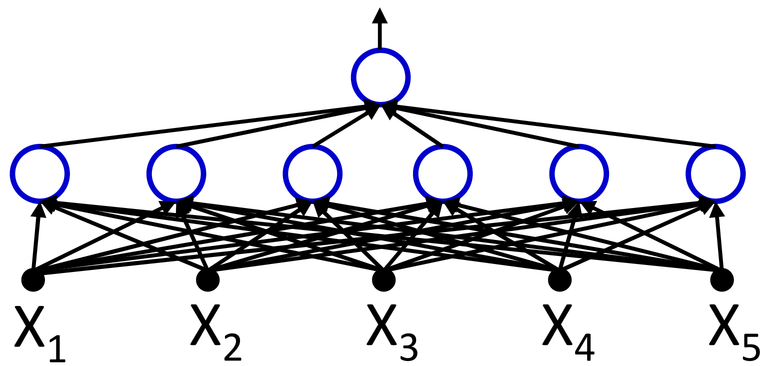
How many layers for a Boolean MLP?

Truth table shows *all* input combinations for which output is 1

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



- *Any truth table can be expressed in this manner!*
- **A one-hidden-layer MLP is a Universal Boolean Function**

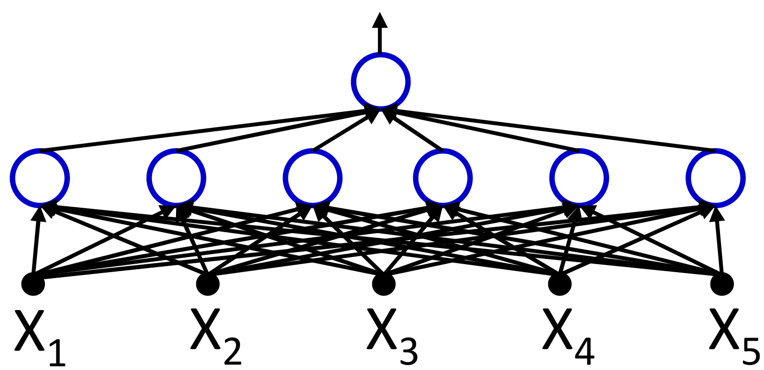
How many layers for a Boolean MLP?

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Truth table shows *all* input combinations for which output is 1

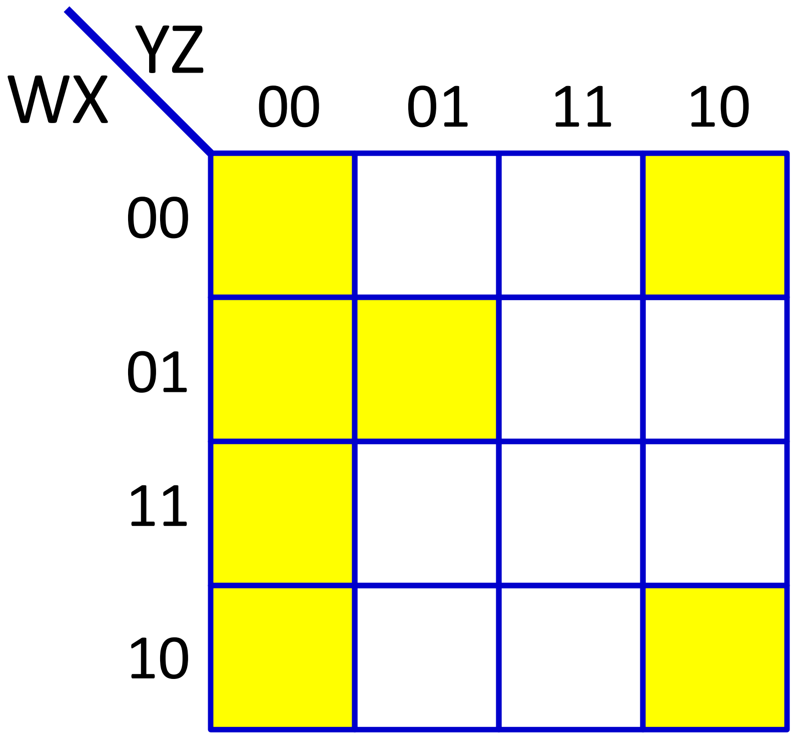
$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



- *Any truth table can be expressed in this manner!*
- **A one-hidden-layer MLP is a Universal Boolean Function**

But what is the largest number of perceptrons required in the single hidden layer for an N-input-variable function?

Reducing a Boolean Function



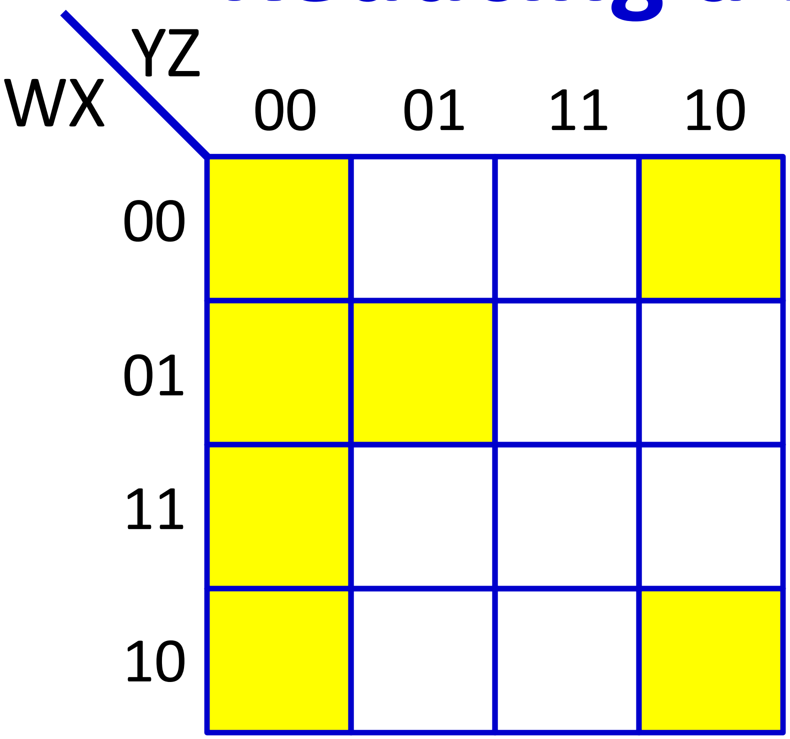
This is a "Karnaugh Map"

It represents a truth table as a grid
Filled boxes represent input combinations for which output is 1; blank boxes have output 0

Adjacent boxes can be "grouped" to reduce the complexity of the DNF formula for the table

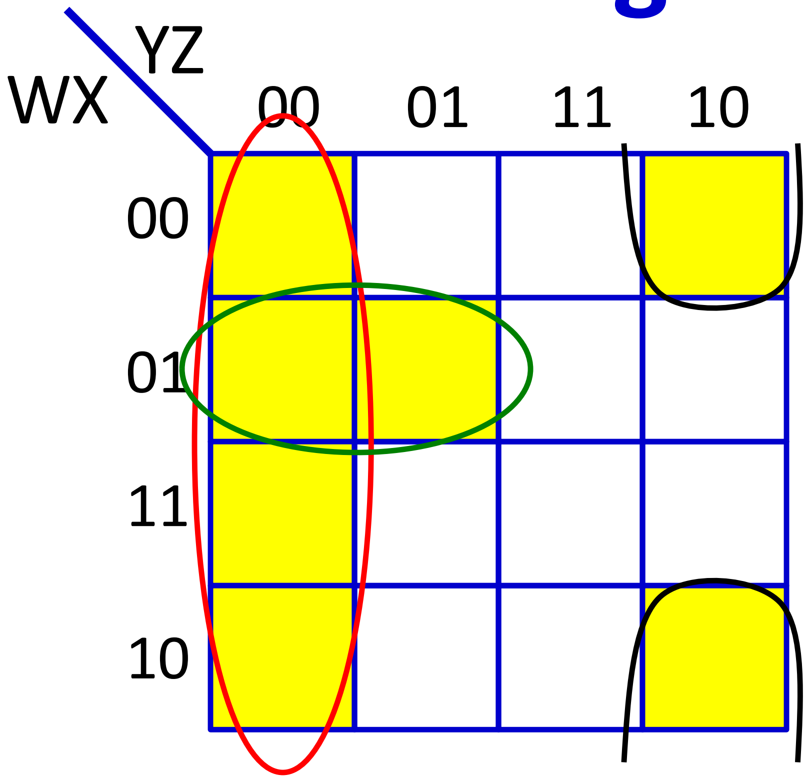
- DNF form:
  - Find groups
  - Express as reduced DNF

Reducing a Boolean Function



Basic DNF formula will require 7 terms

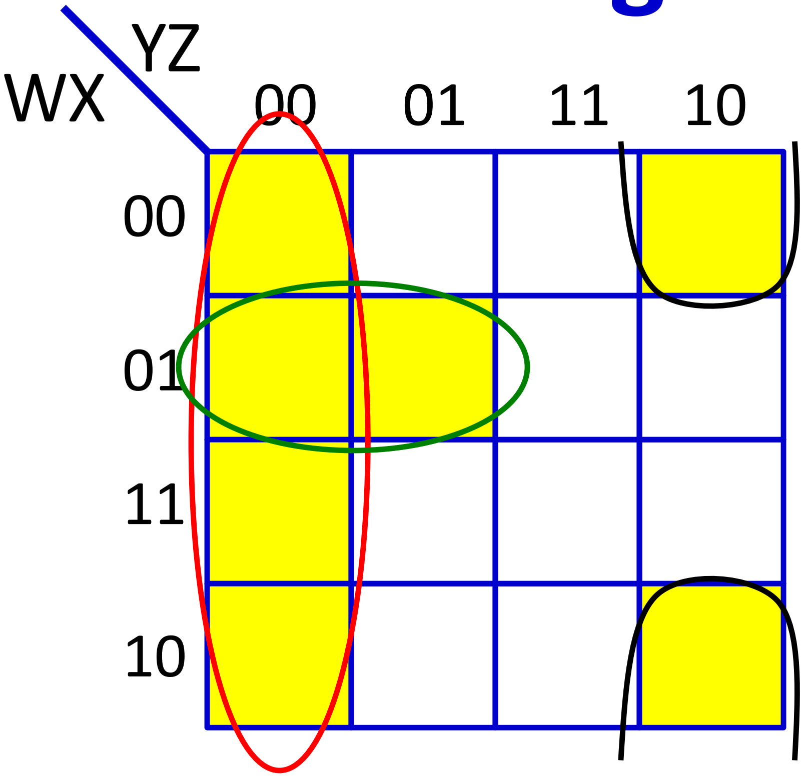
Reducing a Boolean Function



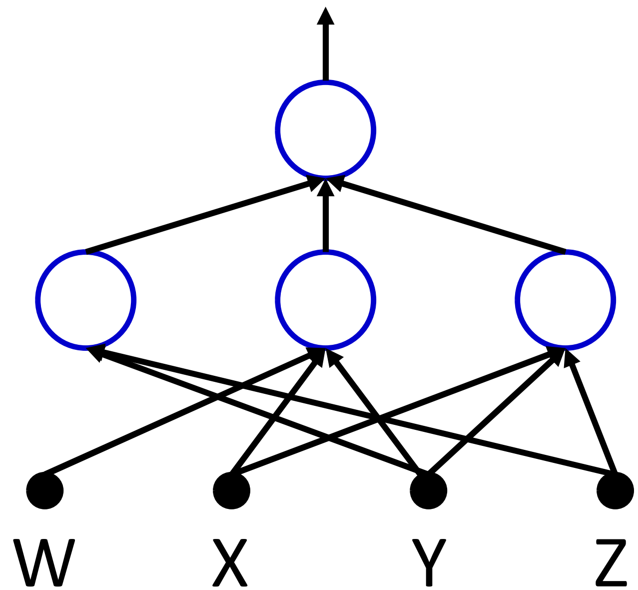
$$O = \bar{Y}\bar{Z} + \bar{W}X\bar{Y} + \bar{X}Y\bar{Z}$$

- *Reduced* DNF form:
  - Find groups
  - Express as reduced DNF

Reducing a Boolean Function



$$O = \bar{Y}\bar{Z} + \bar{W}X\bar{Y} + \bar{X}Y\bar{Z}$$



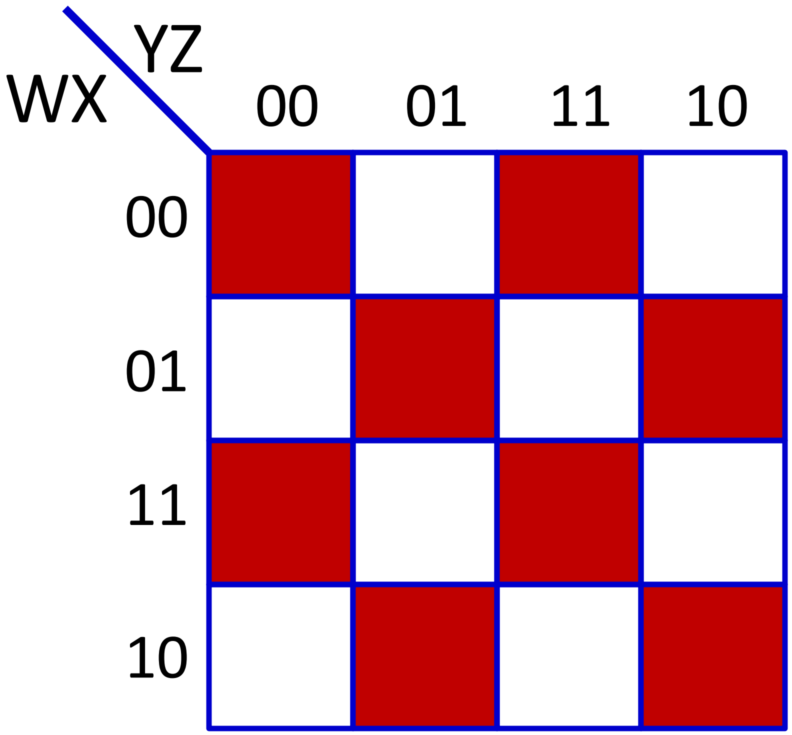
- *Reduced* DNF form:
  - Find groups
  - Express as *reduced* DNF
  - Boolean network for this function needs only 3 hidden units
    - Reduction of the DNF reduces the size of the one-hidden-layer network

Largest irreducible DNF?

WX \ YZ	00	01	11	10
00				
01				
11				
10				

- What arrangement of ones and zeros simply cannot be reduced further?

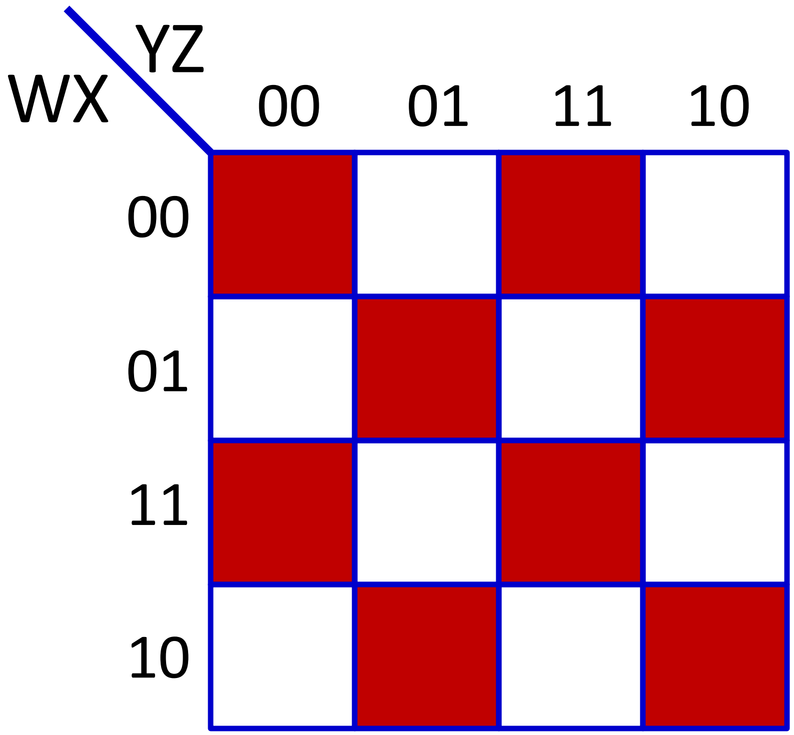
Largest irreducible DNF?



Red=0, white=1

- What arrangement of ones and zeros simply cannot be reduced further?

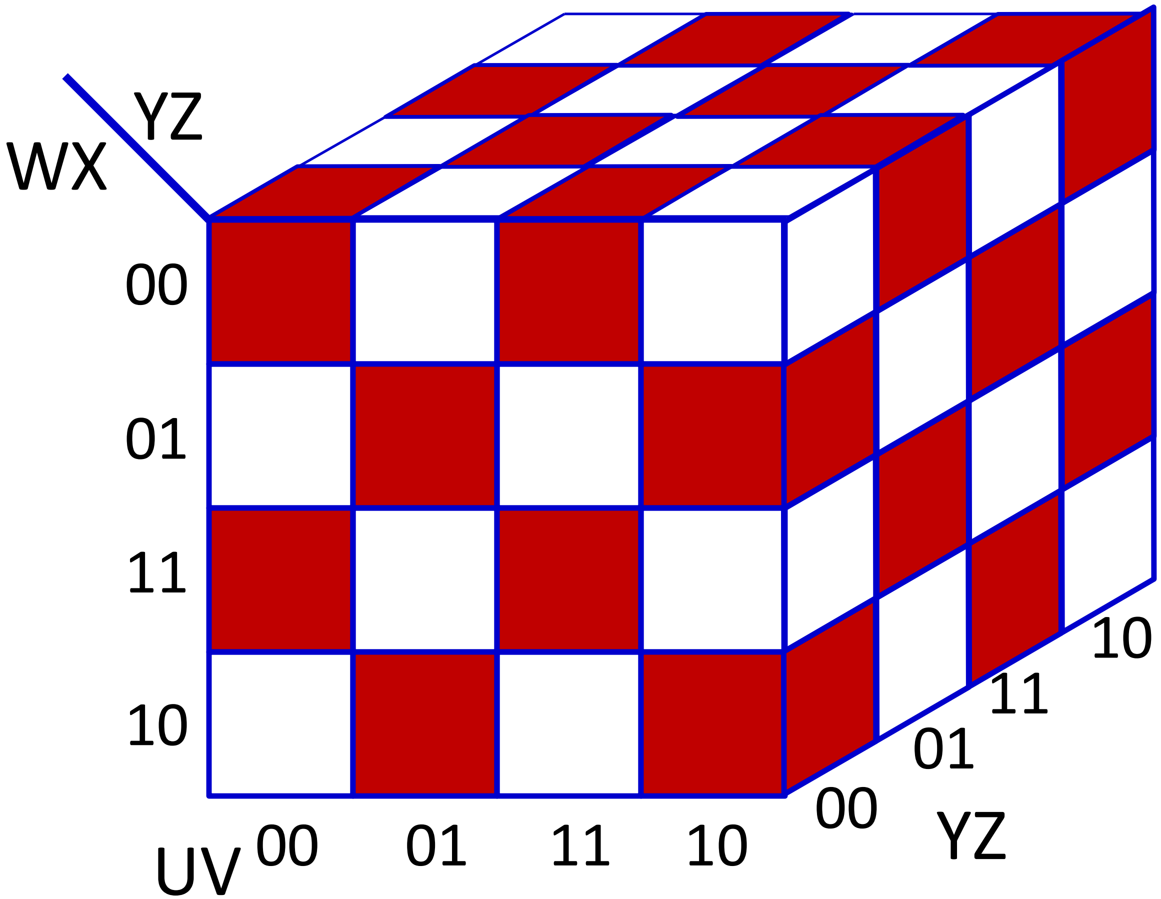
Largest irreducible DNF?



How many neurons in a DNF (one-hidden-layer) MLP for this Boolean function?

- What arrangement of ones and zeros simply cannot be reduced further?

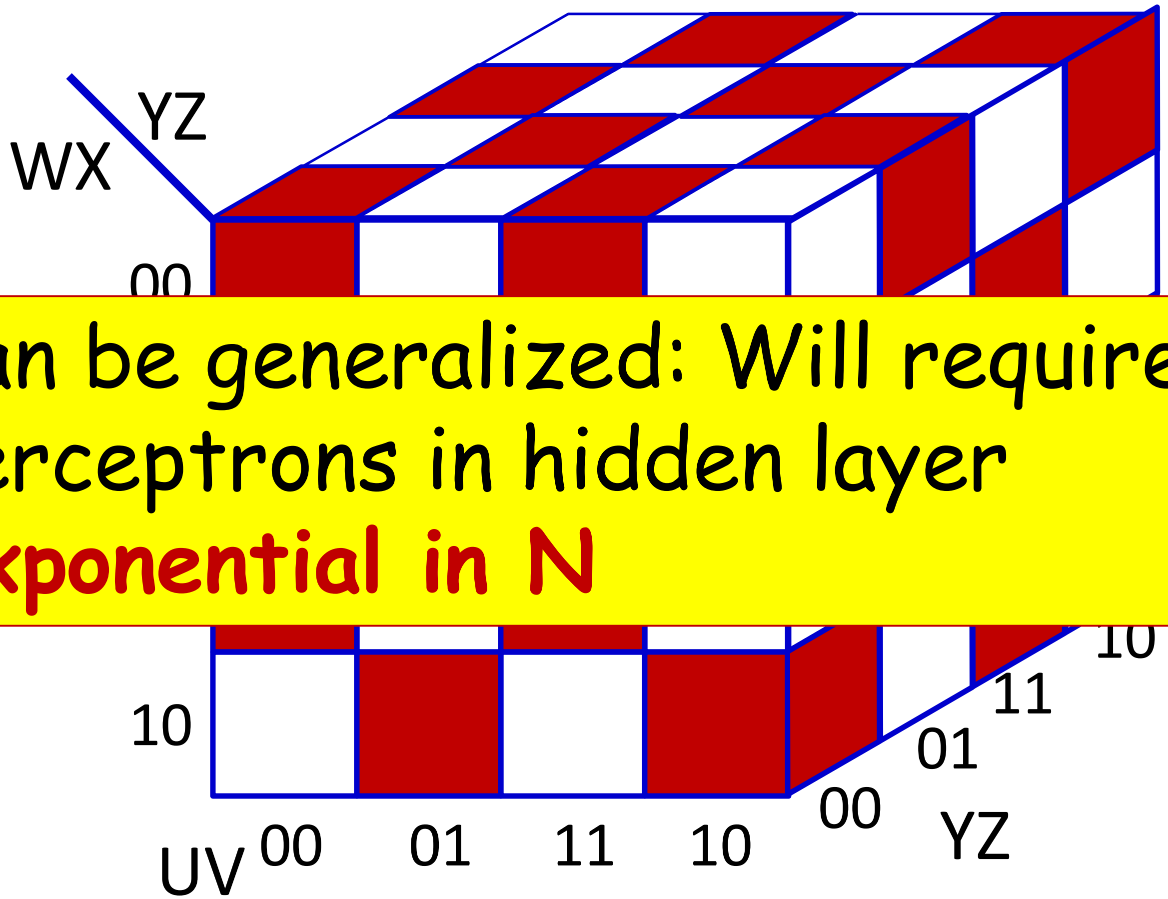
Width of a one-hidden-layer Boolean MLP



Red=0, white=1

- How many neurons in a DNF (one-hidden-layer) MLP for this Boolean function of 6 variables?

Width of a one-hidden-layer Boolean MLP

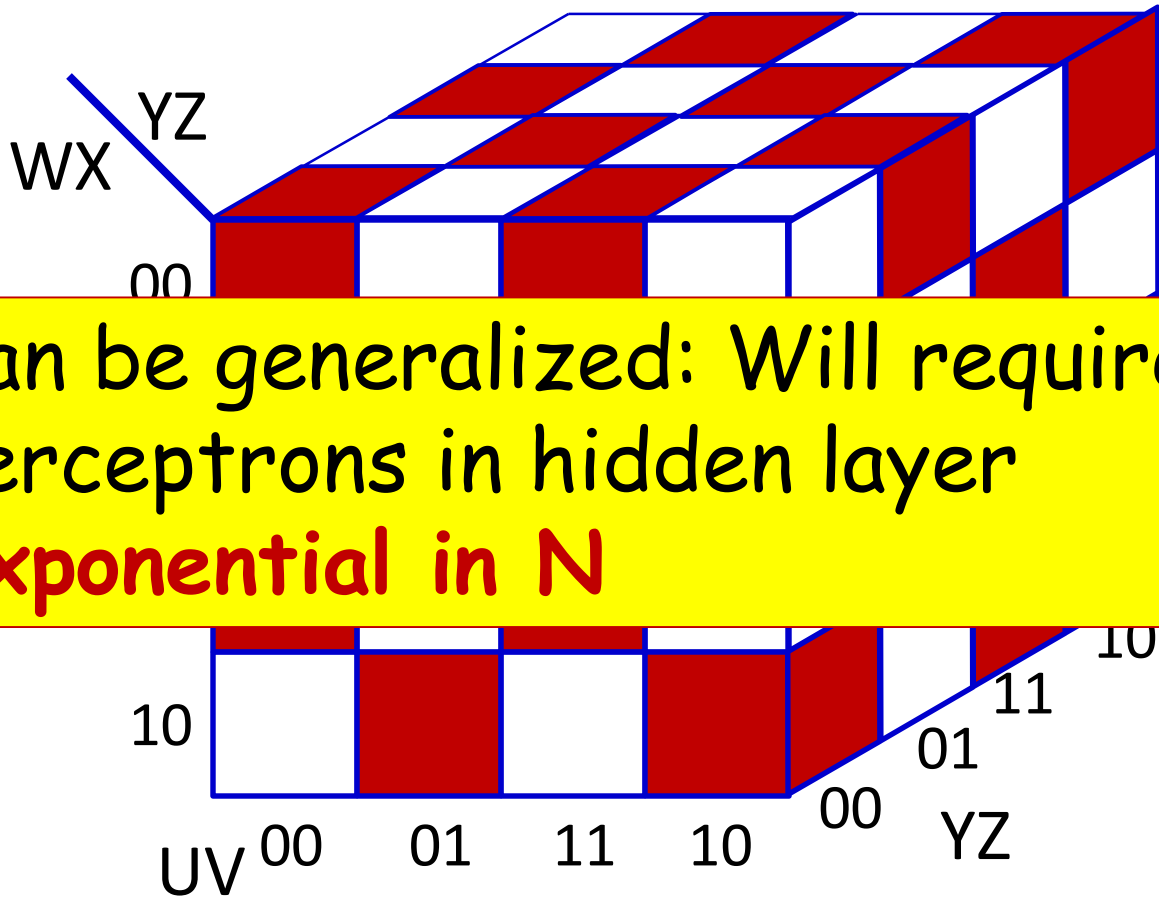


Can be generalized: Will require 2^{N-1} perceptrons in hidden layer

Exponential in N

- How many neurons in a DNF (one-hidden-layer) MLP for this Boolean function

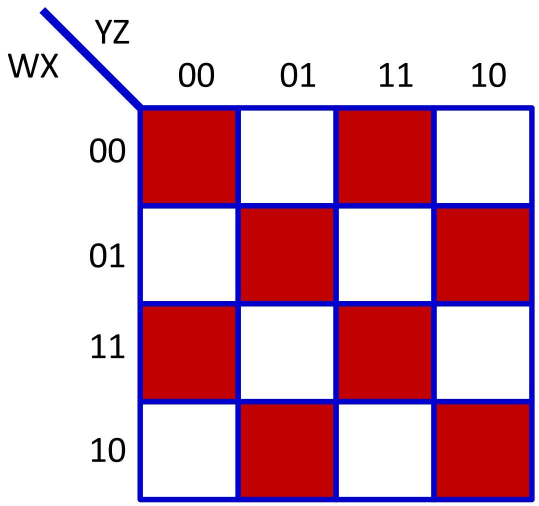
Width of a one-hidden-layer Boolean MLP

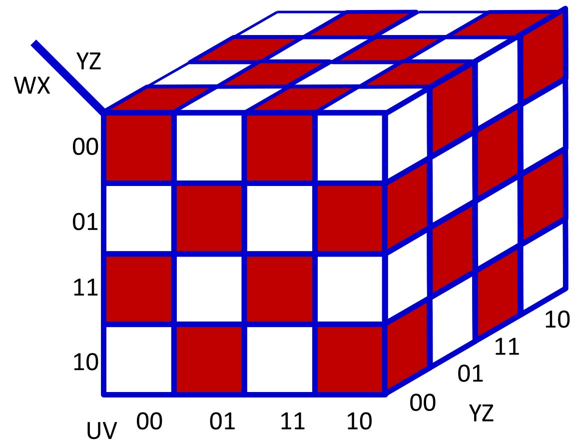


Can be generalized: Will require 2^{N-1} perceptrons in hidden layer
Exponential in N

How many units if we use *multiple hidden layers*?

Size of a deep MLP

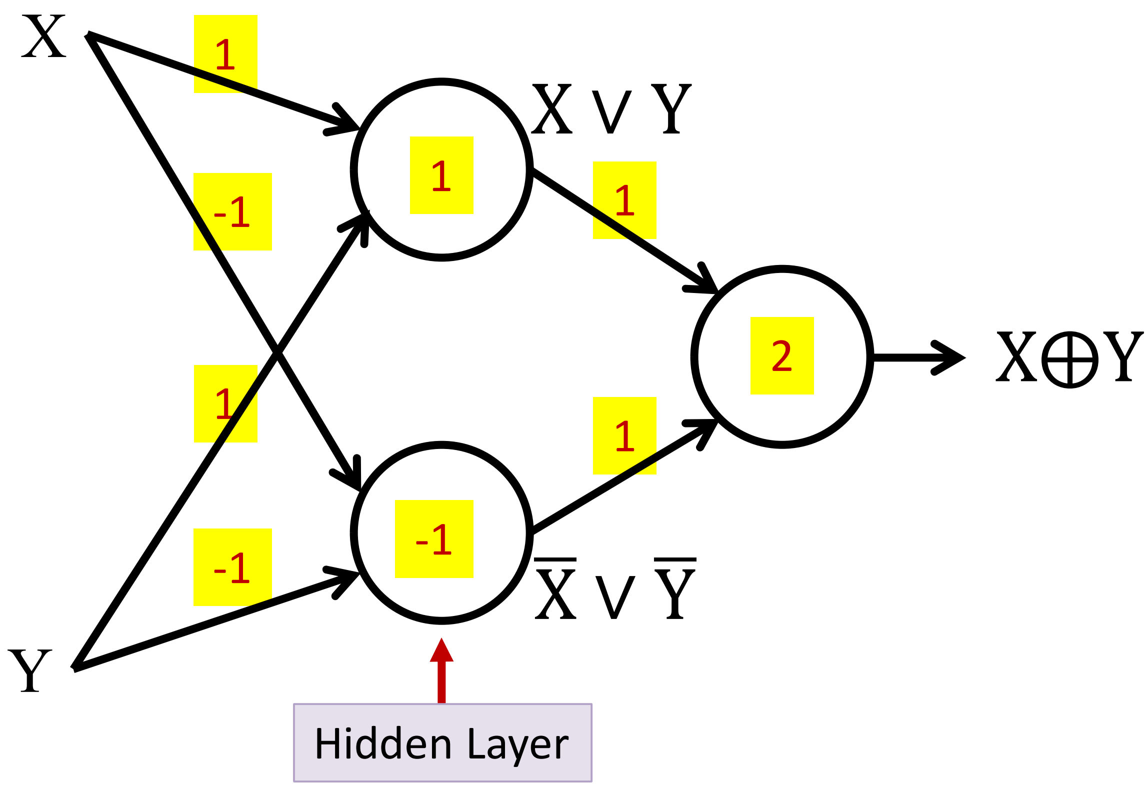




$$O = W \oplus X \oplus Y \oplus Z$$

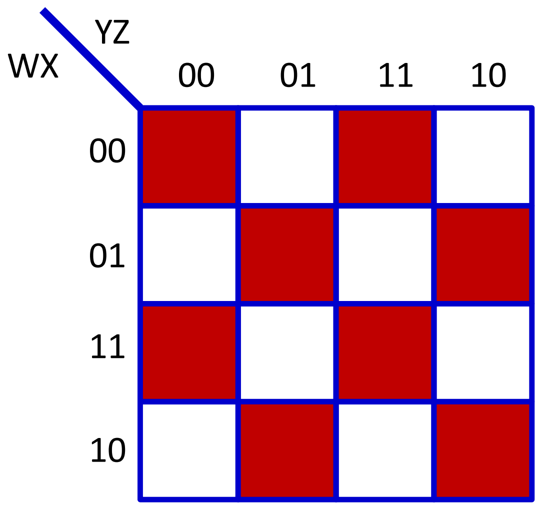
$$O = U \oplus V \oplus W \oplus X \oplus Y \oplus Z$$

Multi-layer perceptron XOR

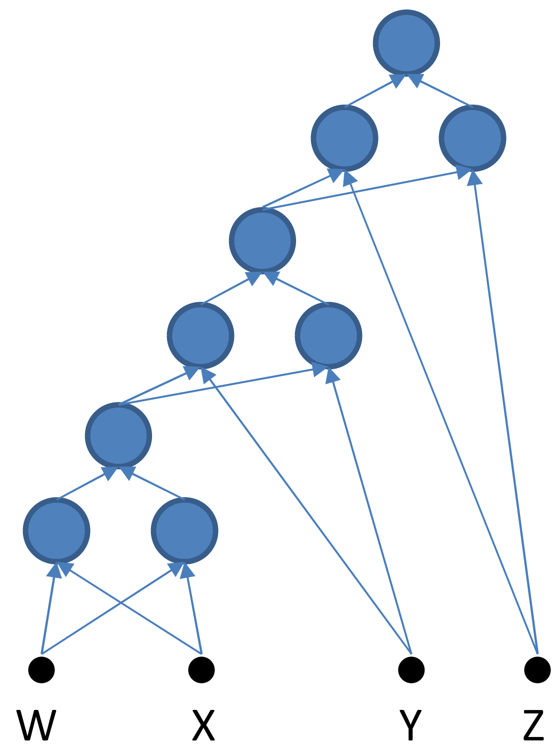


- An XOR takes three perceptrons

Size of a deep MLP



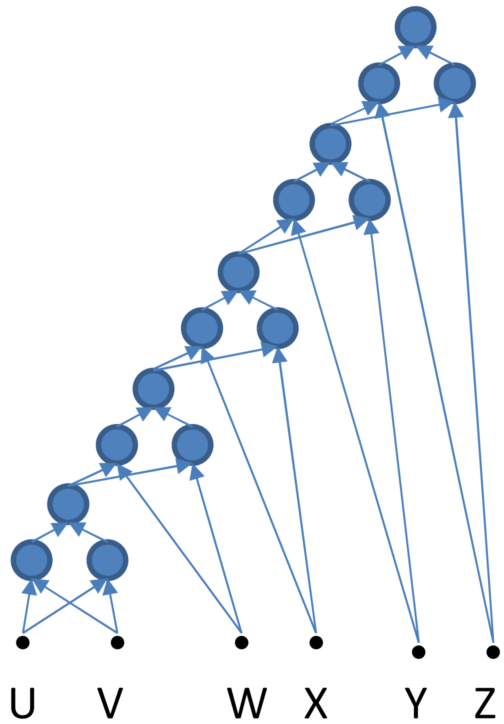
$$O = W \oplus X \oplus Y \oplus Z$$

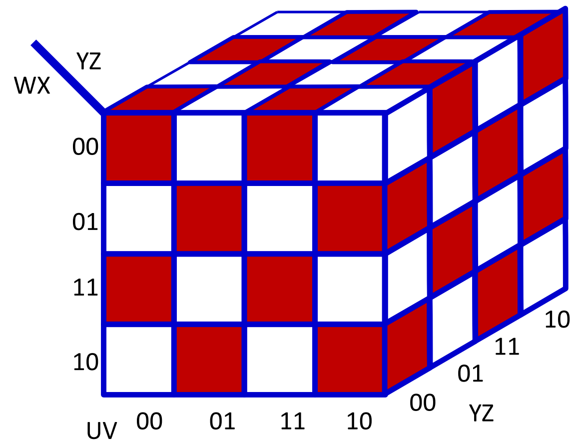


9 perceptrons

- An XOR needs 3 perceptrons
- This network will require 3x3 = 9 perceptrons

Size of a deep MLP



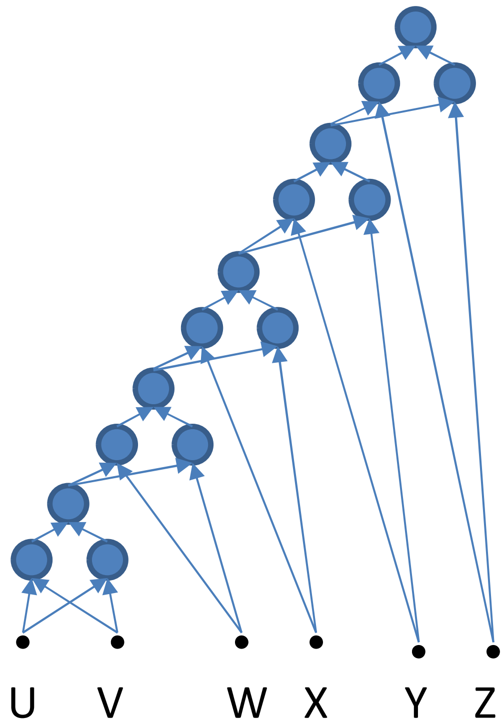


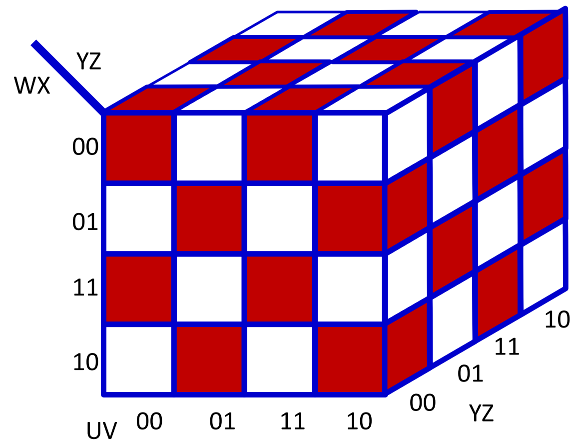
$$O = U \oplus V \oplus W \oplus X \oplus Y \oplus Z$$

15 perceptrons

- An XOR needs 3 perceptrons
- This network will require 3x5 = 15 perceptrons

Size of a deep MLP



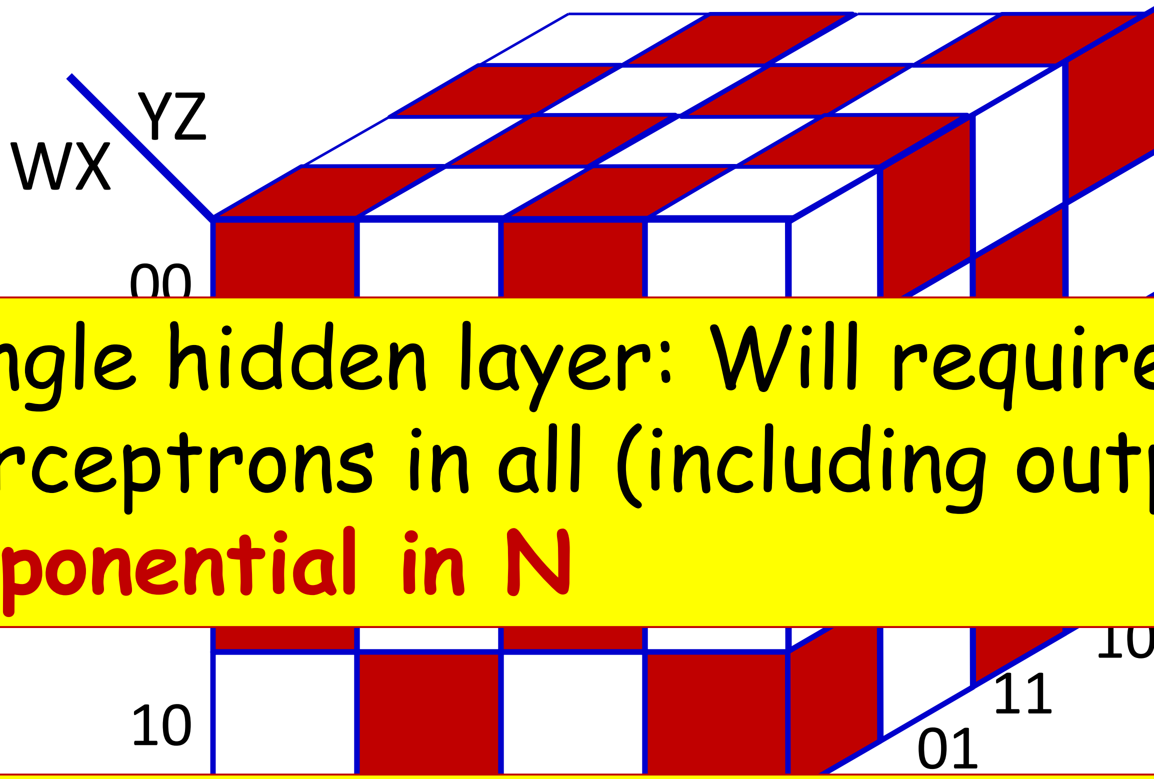


$$O = U \oplus V \oplus W \oplus X \oplus Y \oplus Z$$

More generally, the XOR of N variables will require 3(N-1) perceptrons!!

- An XOR needs 3 perceptrons
- This network will require 3x5 = 15 perceptrons

One-hidden layer vs deep Boolean MLP



Single hidden layer: Will require $2^{N-1}+1$ perceptrons in all (including output unit)

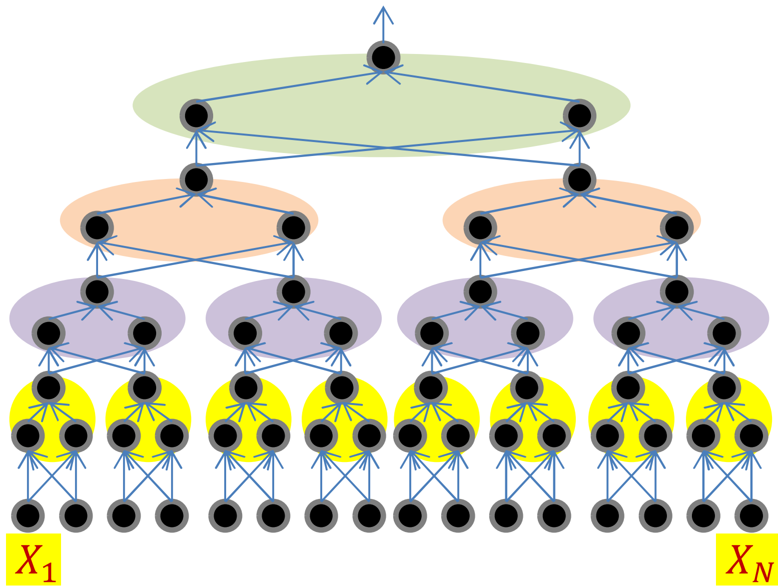
Exponential in N

Will require 3(N-1) perceptrons in a deep network

Linear in N!!!

Can be arranged in only $2\log_2(N)$ layers

A better representation

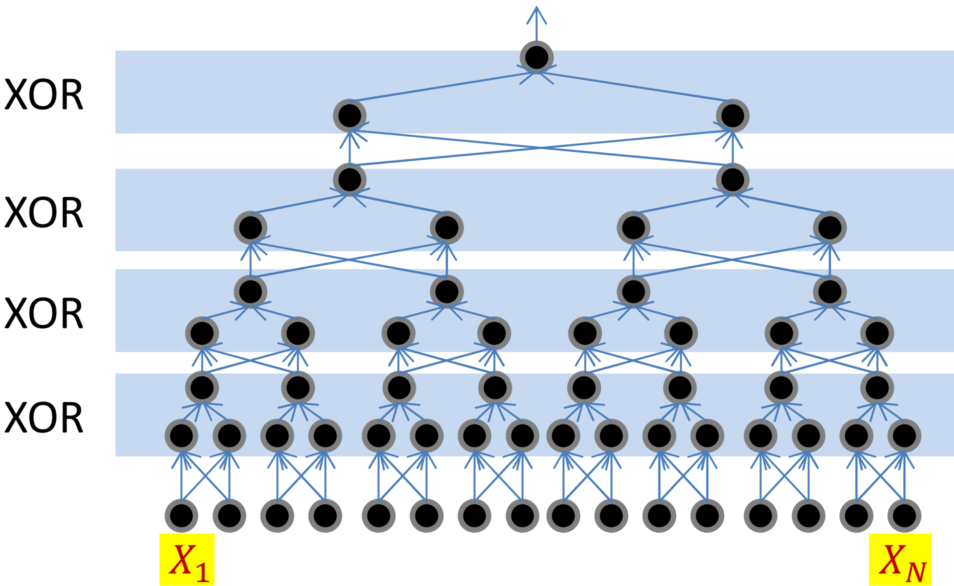


$$O = X_1 \oplus X_2 \oplus \cdots \oplus X_N$$

- Only $2\log_2 N$ layers
  - By pairing terms
  - 2 layers per XOR

$$O = (((((X_1 \oplus X_2) \oplus (X_3 \oplus X_4)) \oplus ((X_5 \oplus X_6) \oplus (X_7 \oplus X_8))) \oplus (((\ldots$$

A better representation

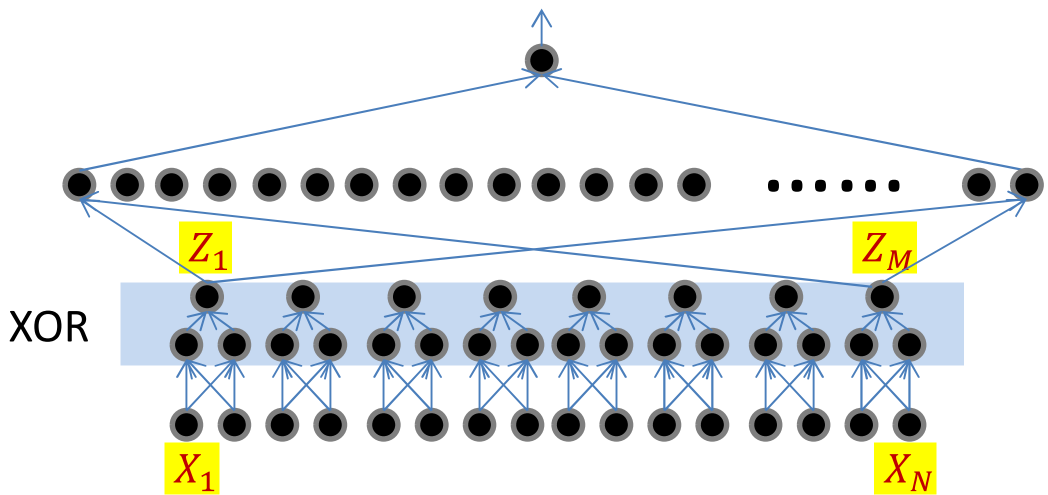


$$O = X_1 \oplus X_2 \oplus \cdots \oplus X_N$$

- Only $2 \log_2 N$ layers
  - By pairing terms
  - 2 layers per XOR

$$O = (((((X_1 \oplus X_2) \oplus (X_3 \oplus X_4)) \oplus ((X_5 \oplus X_6) \oplus (X_7 \oplus X_8))) \oplus (((\ldots$$

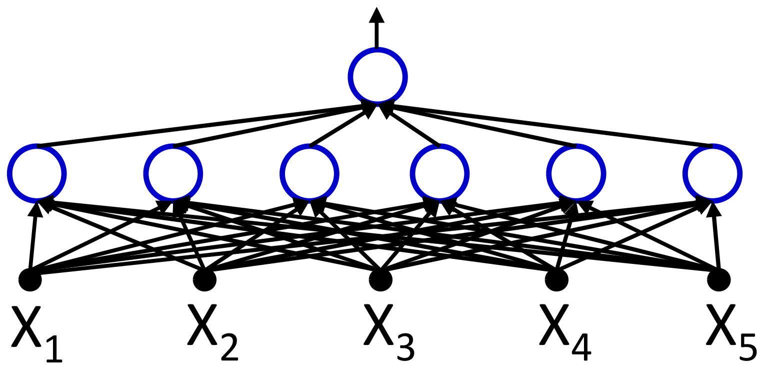
The challenge of depth



$$O = X_1 \oplus X_2 \oplus \cdots \oplus X_N$$
$$= Z_1 \oplus Z_2 \oplus \cdots \oplus Z_M$$

- Using only K hidden layers will require O(2^{CN}) neurons in the Kth layer, where $C = 2^{-(K-1)/2}$
  - Because the output can be shown to be the XOR of all the outputs of the K-1th hidden layer
  - **I.e. reducing the number of layers below the minimum will result in an exponentially sized network to express the function fully**
  - **A network with fewer than the minimum required number of neurons *cannot* model the function**

The actual number of parameters in a network



- The actual number of parameters in a network is the number of *connections*
  – In this example there are 30

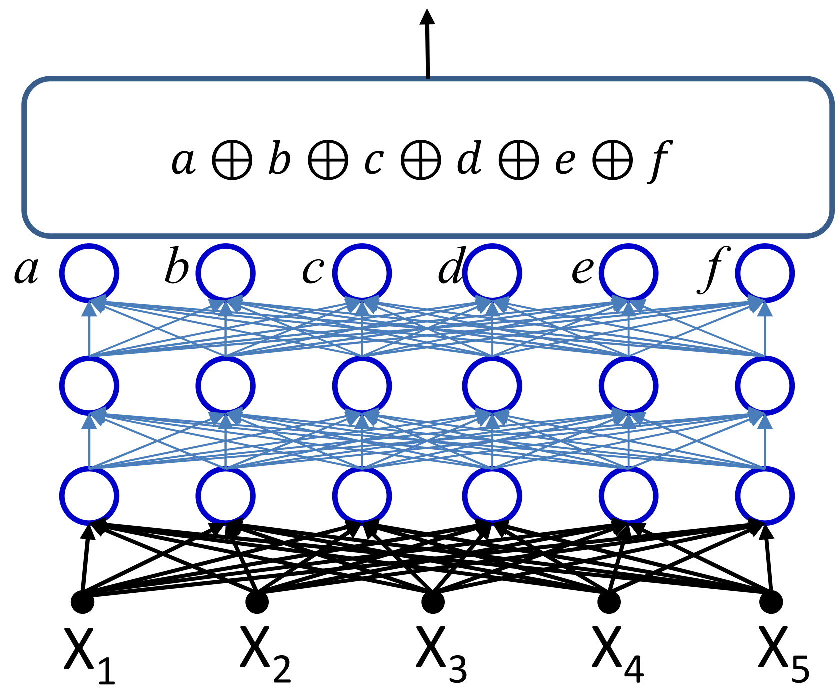
- This is the number that really matters in software or hardware implementations
- Networks that require an exponential number of neurons will require an exponential number of weights..

Recap: The need for depth

- *Deep* Boolean MLPs that scale *linearly* with the number of inputs …
- … can become exponentially large if recast using only one hidden layer

The need for depth

The XORs could occur anywhere!



- An MLP for any function that can eventually be expressed as the XOR of a number of intermediate variables will require depth.
  - The XOR structure could occur in any layer
  - If you have a fixed depth from that point on, the network can grow exponentially in size.
- Having a few extra layers can greatly reduce network size

Depth vs Size in Boolean Circuits

- The XOR is really a parity problem

- Any *Boolean* parity circuit of depth d using AND,OR and NOT gates with unbounded fan-in must have size $2^{n^{1/d}}$
  - Parity, Circuits, and the Polynomial-Time Hierarchy, M. Furst, J. B. Saxe, and M. Sipser, Mathematical Systems Theory 1984
  - Alternately stated: $parity \notin AC^0$
    - Set of constant-depth polynomial size circuits of unbounded fan-in elements

Caveat 1: Not all Boolean functions..

- Not all Boolean circuits have such clear depth-vs-size tradeoff

- Shannon's theorem: For $n > 2$, there is a Boolean function of n variables that requires at least $2^n/n$ Boolean gates
  - More correctly, for large n, almost all n-input Boolean functions need more than $2^n/n$ Boolean gates
    - Regardless of depth

- Note: If all Boolean functions over n inputs could be computed using a circuit of size that is polynomial in n, P = NP!

Network size: summary

- An MLP is a universal Boolean function
- But can represent a given function only if
  - It is sufficiently wide
  - It is sufficiently deep
  - Depth can be traded off for (sometimes) exponential growth of the width of the network
- Optimal width and depth depend on the number of variables and the complexity of the Boolean function
  - Complexity: *minimal* number of terms in DNF formula to represent it

Story so far

- Multi-layer perceptrons are *Universal Boolean Machines*
- Even a network with a *single* hidden layer is a universal Boolean machine
  - But a single-layer network may require an exponentially large number of perceptrons
- Deeper networks may require far fewer neurons than shallower networks to express the same function
  - Could be *exponentially* smaller

Caveat 2

- Used a simple "Boolean circuit" analogy for explanation
- We actually have *threshold circuit* (TC) not, just a Boolean circuit (AC)
  - Specifically composed of threshold gates
    - More versatile than Boolean gates (can compute majority function)
      - E.g. "at least K inputs are 1" is a single TC gate, but an exponential size AC
      - For fixed depth, $Boolean\ circuits \subset threshold\ circuits$ (strict subset)
  - A depth-2 TC parity circuit can be composed with $\mathcal{O}(n^2)$ weights
    - But a network of depth $\log(n)$ requires only $\mathcal{O}(n)$ weights
  - But more generally, for large n, for most Boolean functions, a threshold circuit that is polynomial in n at optimal depth d may become exponentially large at $d - 1$

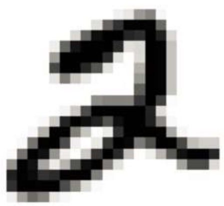
- Other formal analyses typically view neural networks as *arithmetic circuits*
  - Circuits which compute polynomials over any field

- So let's consider functions over the field of reals

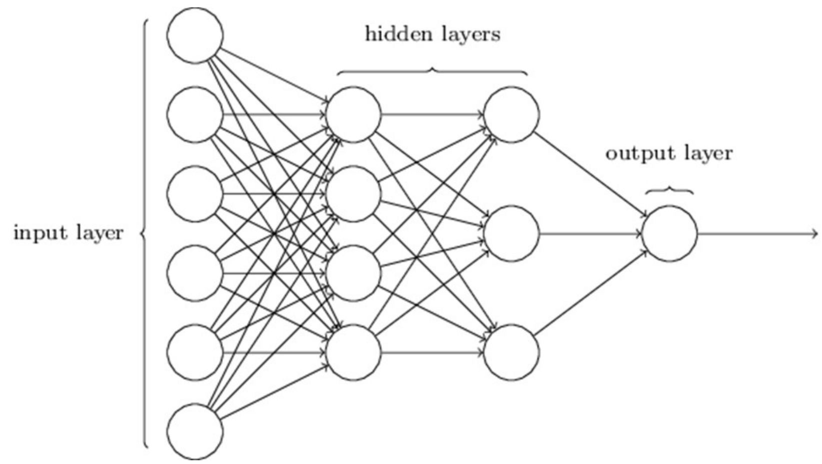
Today

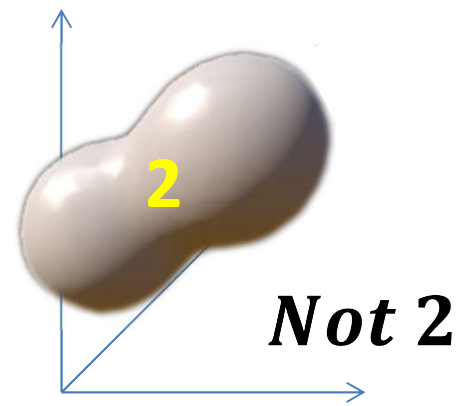
- Multi-layer Perceptrons as universal Boolean functions
  - The need for depth
- MLPs as universal classifiers
  - The need for depth
- MLPs as universal approximators
- A discussion of optimal depth and width
- Brief segue: RBF networks

Recap: The MLP as a classifier



784 dimensions
(MNIST)

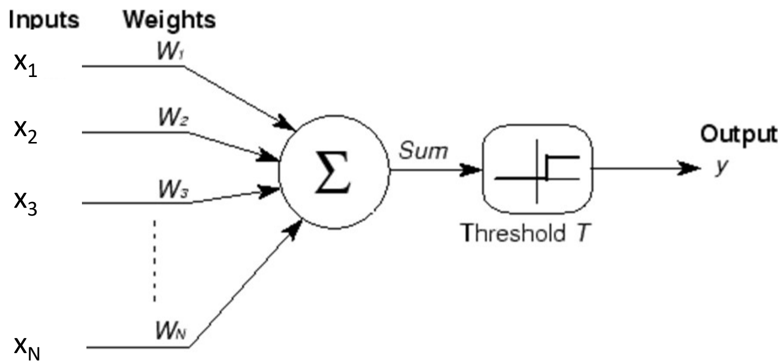




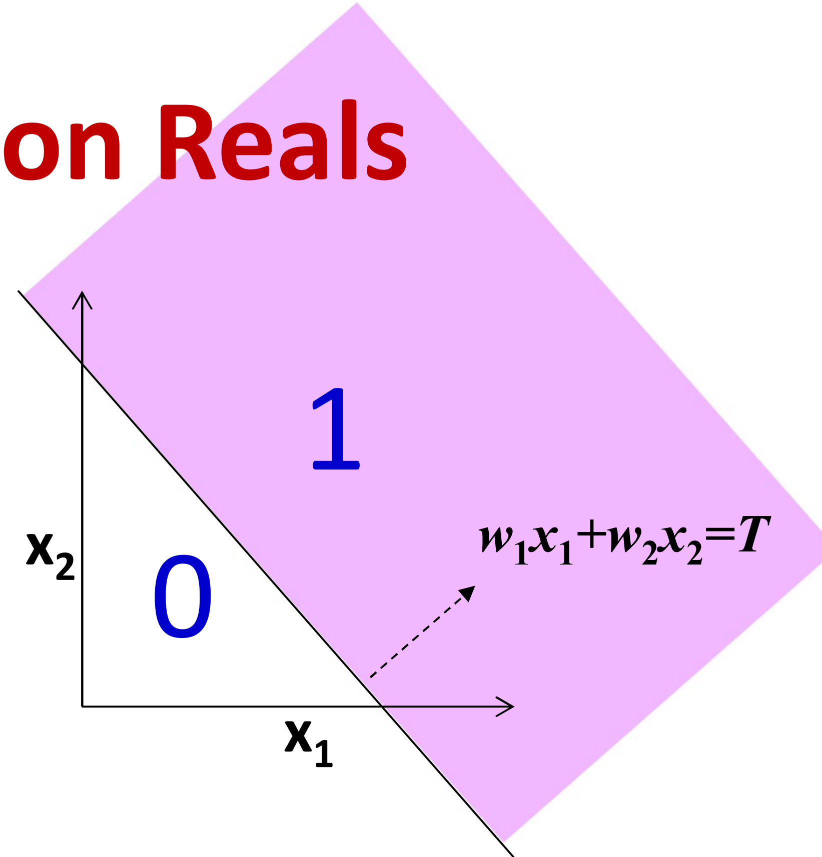
784 dimensions

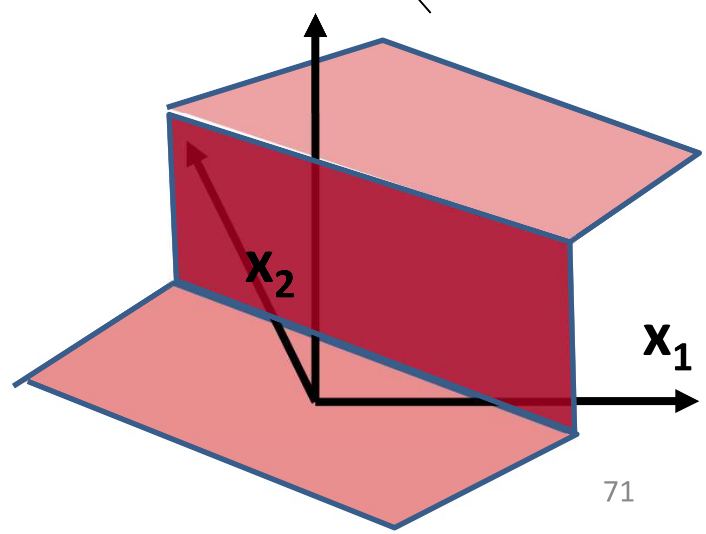
- MLP as a function over real inputs
- MLP as a function that finds a complex "decision boundary" over a space of *reals*

A Perceptron on Reals



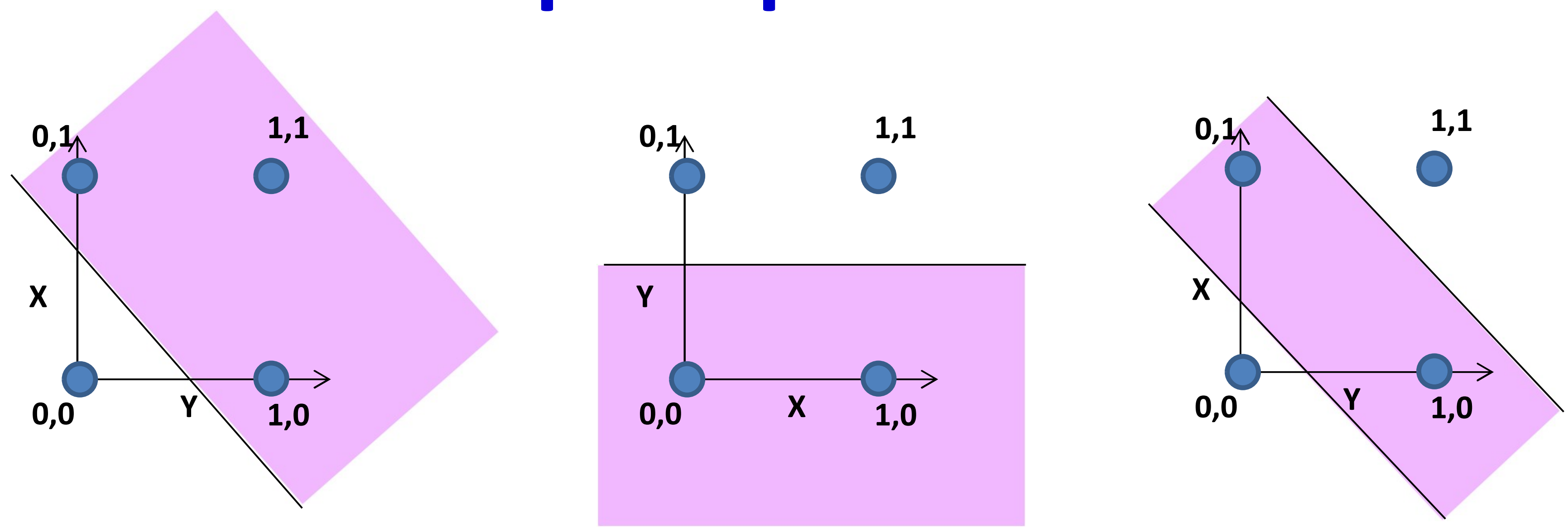
$$y = \begin{cases} 1 \; if \; \sum_i w_i x_i \geq T \\ 0 \; else \end{cases}$$





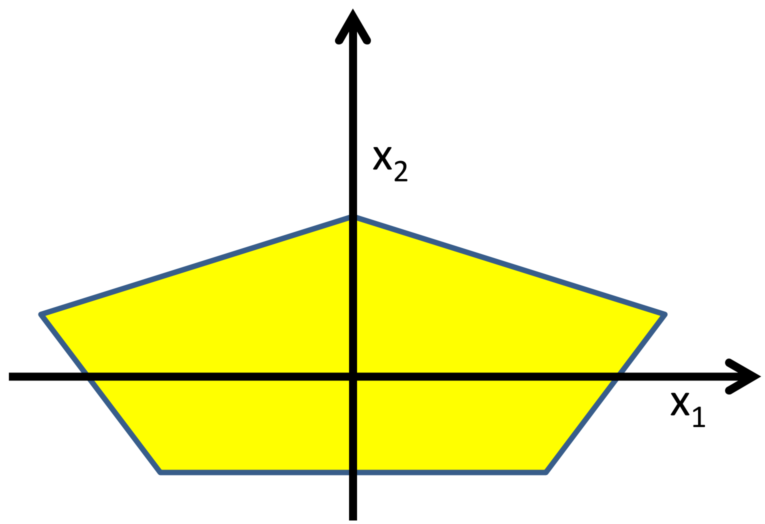
- A perceptron operates on *real*-valued vectors
  - **This is a *linear classifier***

Boolean functions with a real perceptron



- Boolean perceptrons are also linear classifiers
  - Purple regions are 1

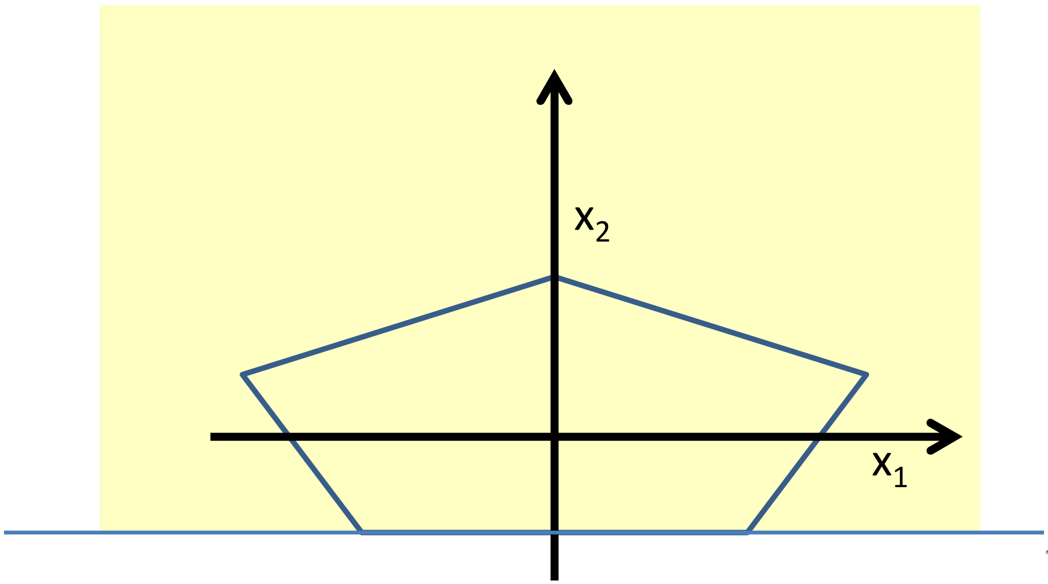
Composing complicated "decision" boundaries

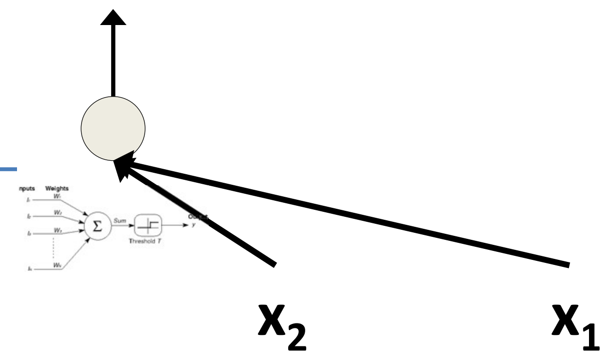


Can now be composed into "networks" to compute arbitrary classification "boundaries"

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

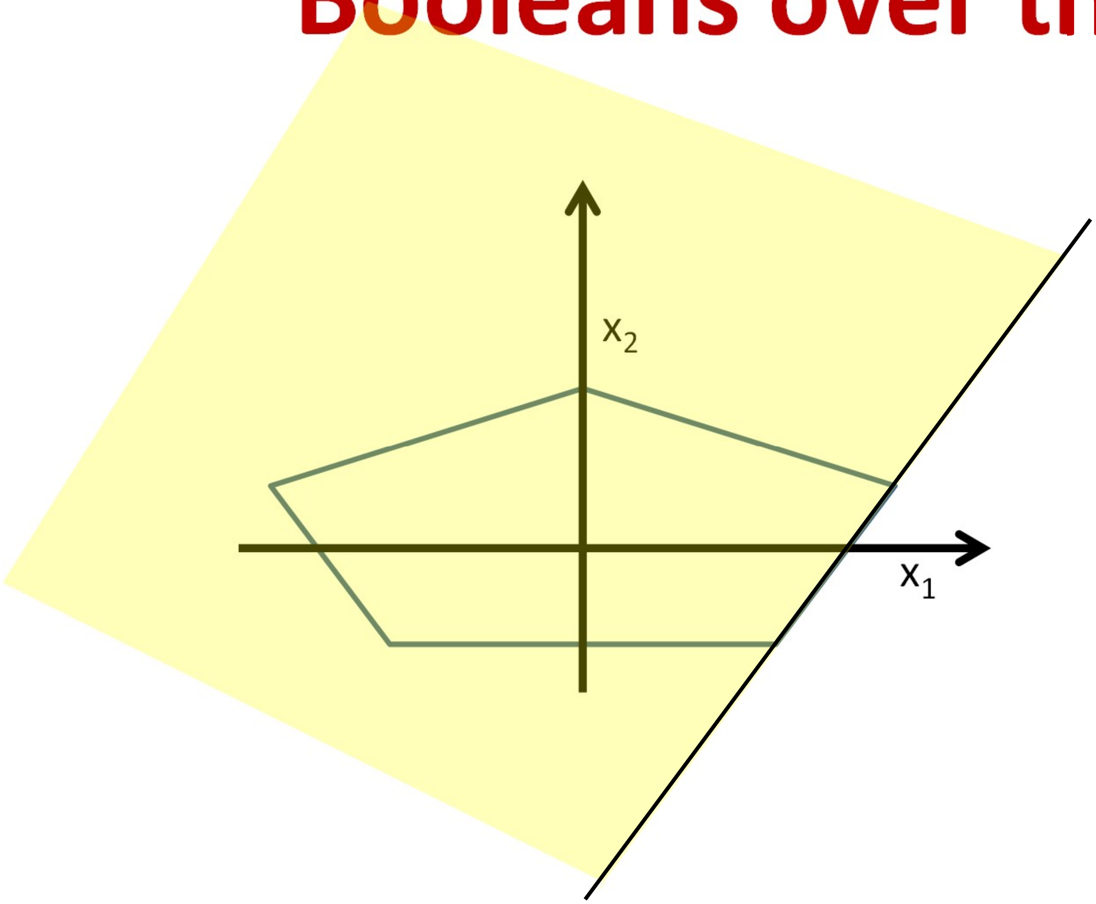
Booleans over the reals

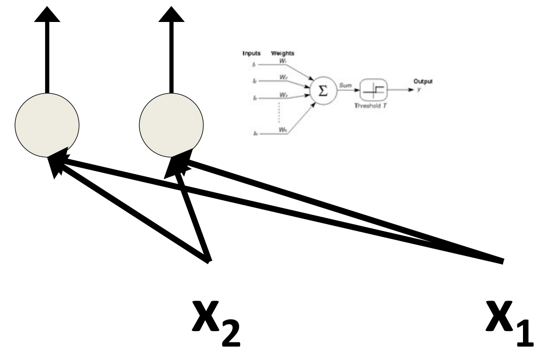




- The network must fire if the input is in the coloured area

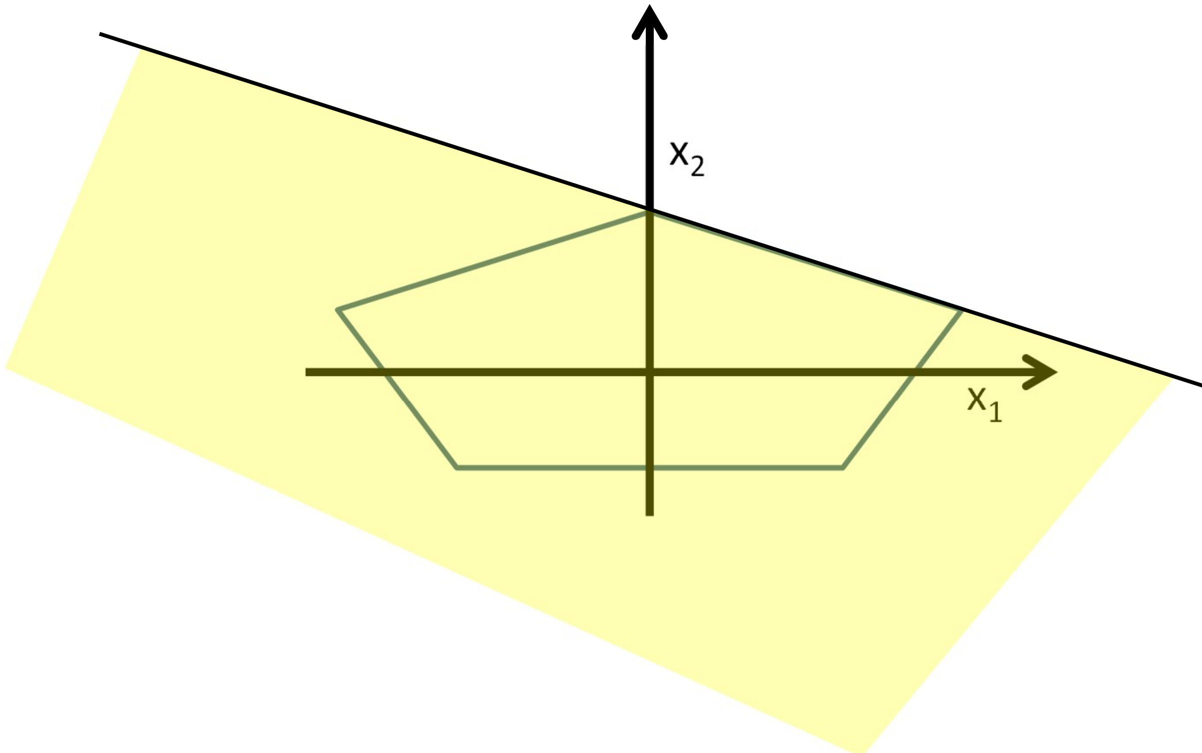
Booleans over the reals

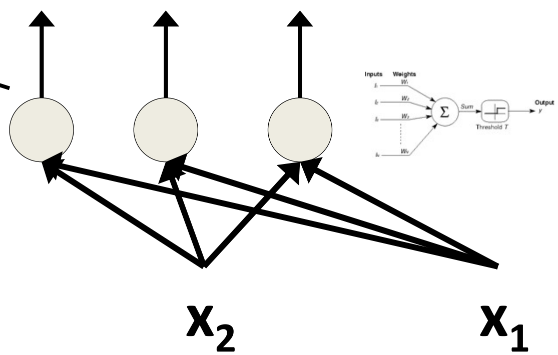




- The network must fire if the input is in the coloured area

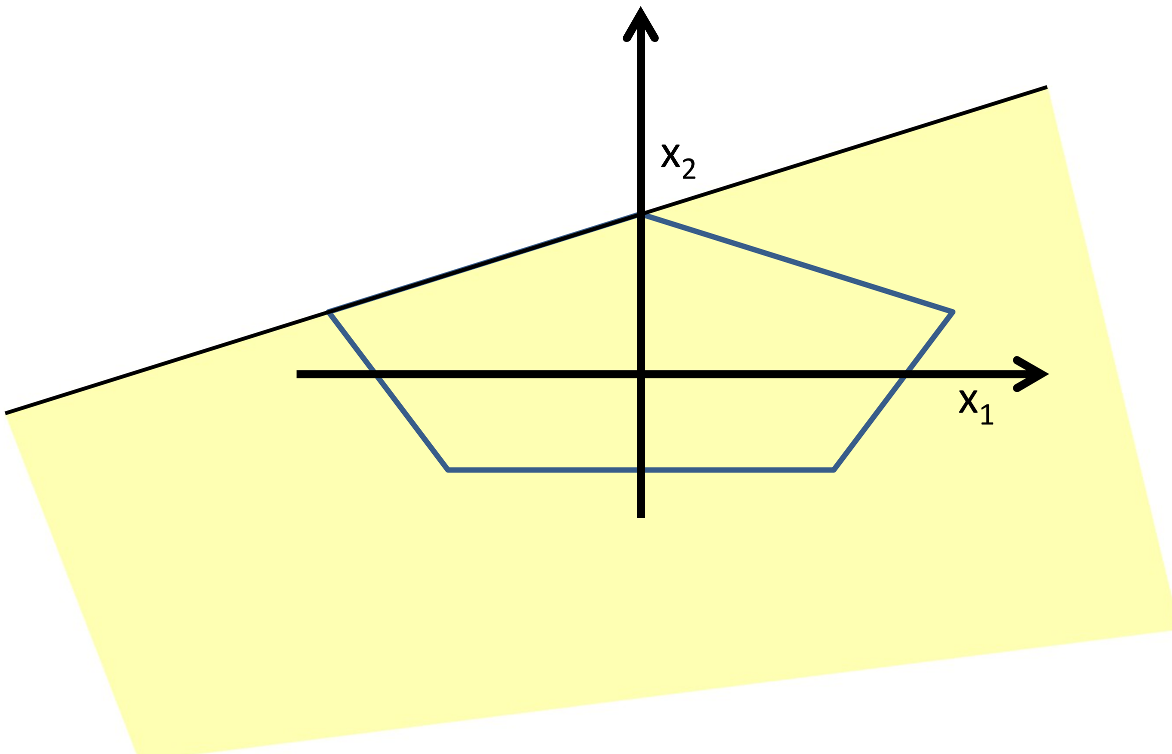
Booleans over the reals

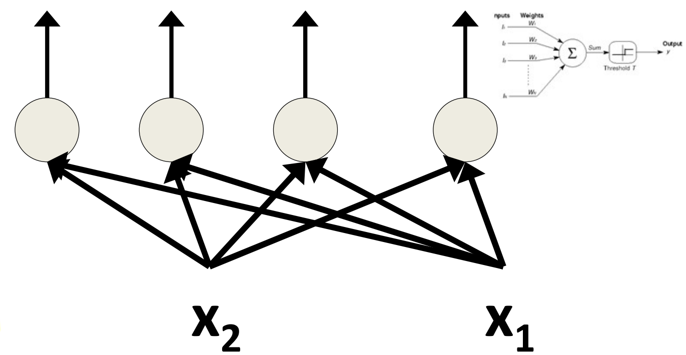




- The network must fire if the input is in the coloured area

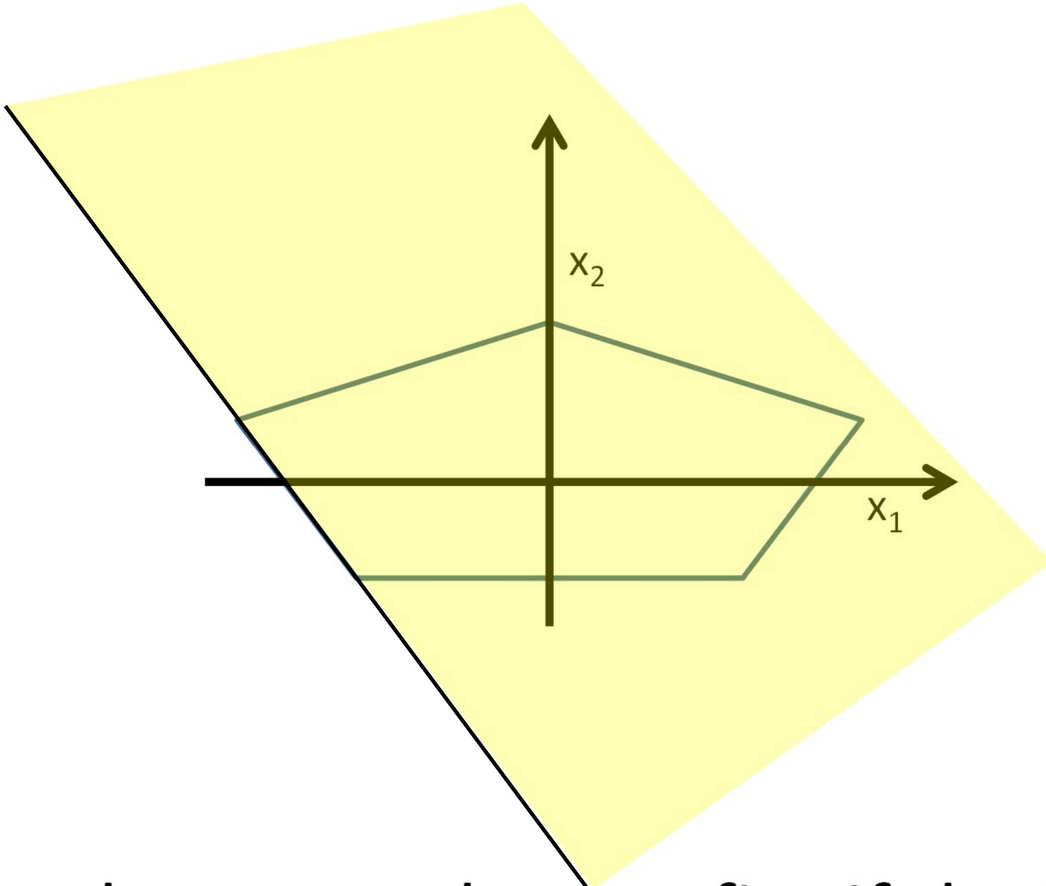
Booleans over the reals

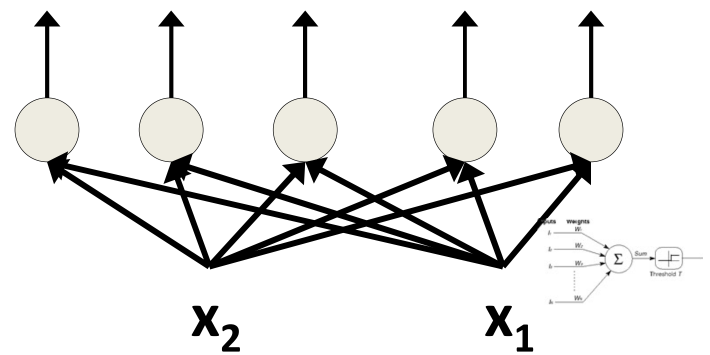




- The network must fire if the input is in the coloured area

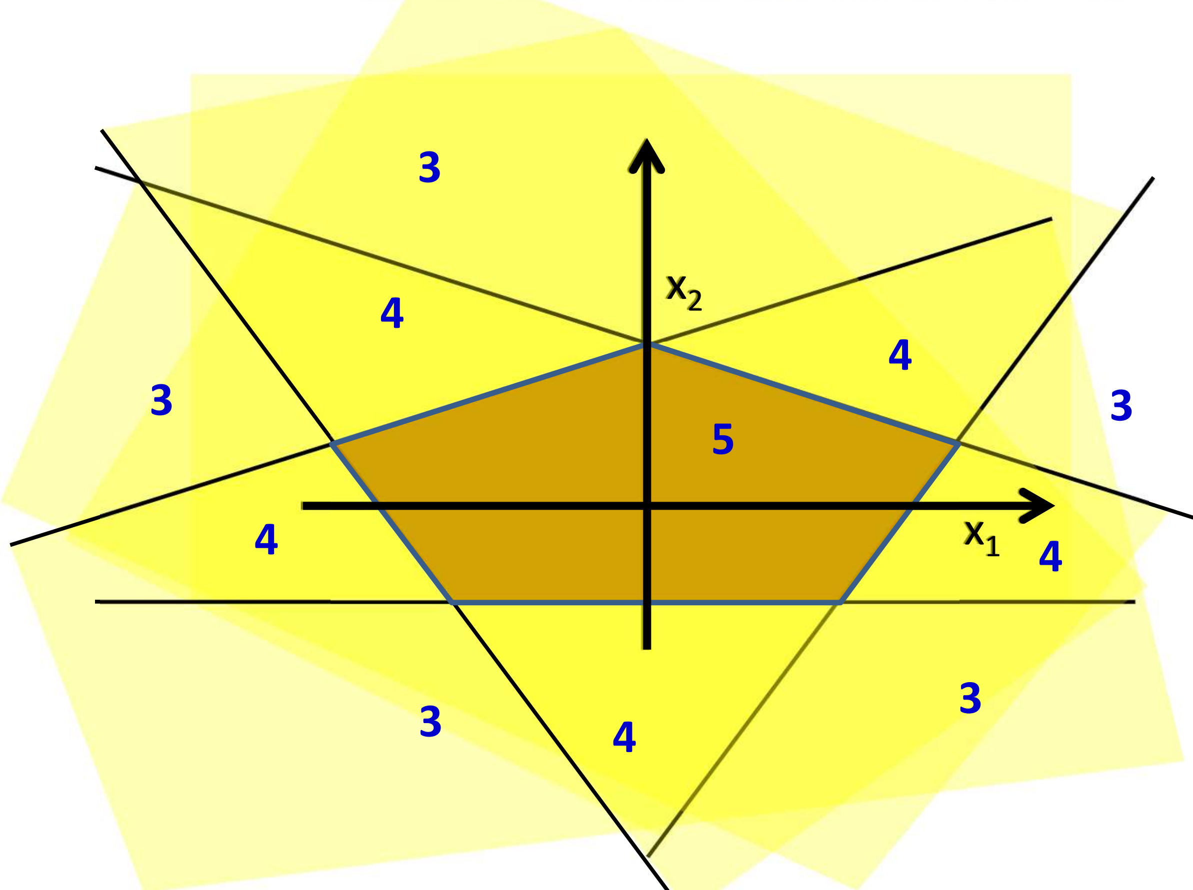
Booleans over the reals

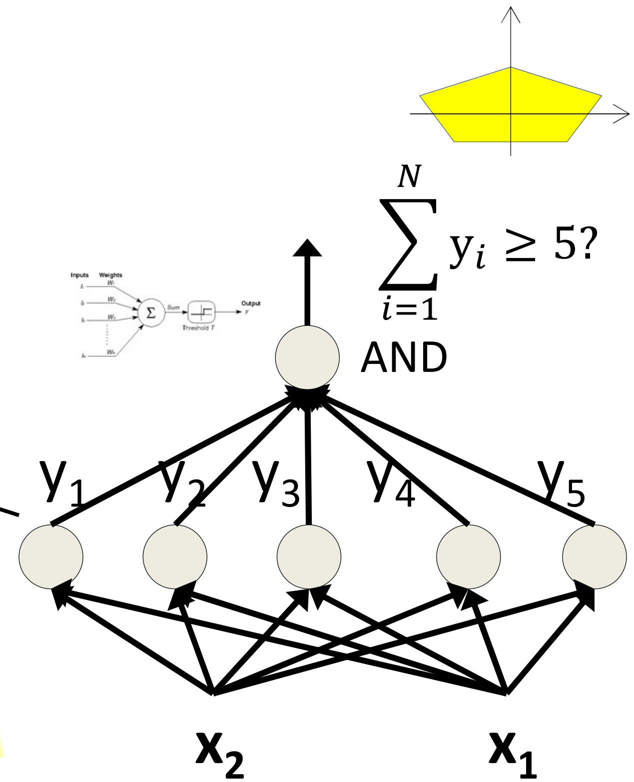




- The network must fire if the input is in the coloured area

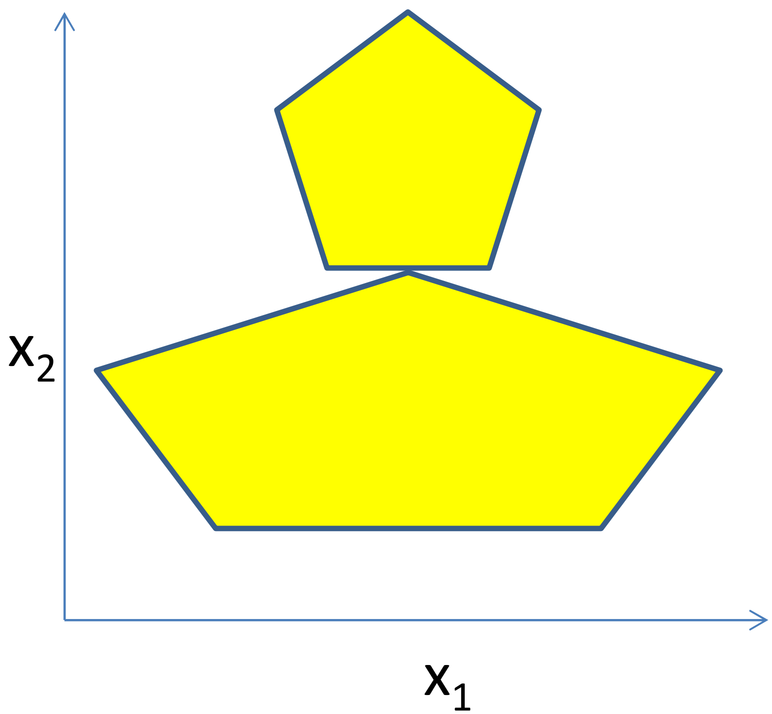
Booleans over the reals

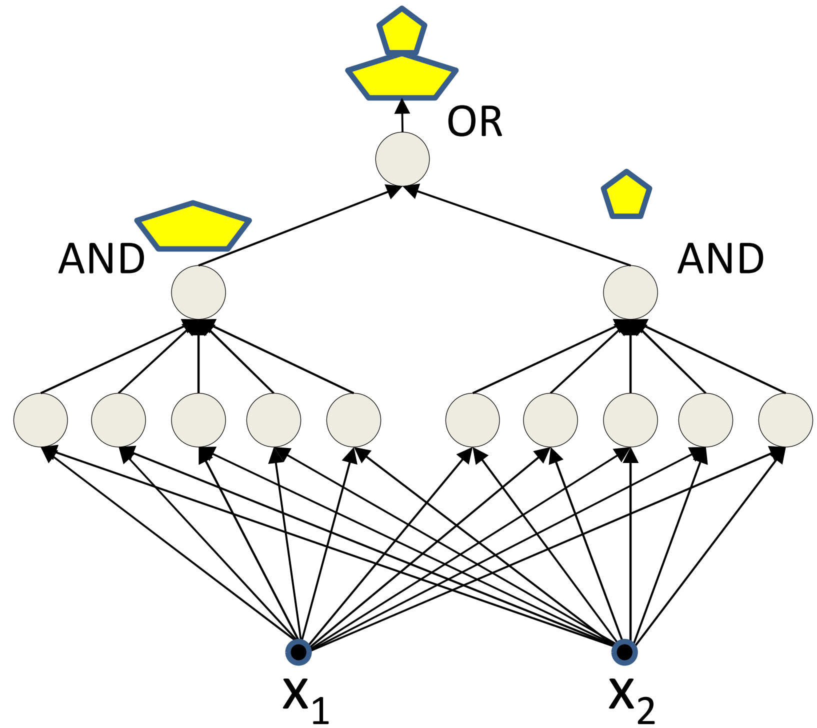




- The network must fire if the input is in the coloured area

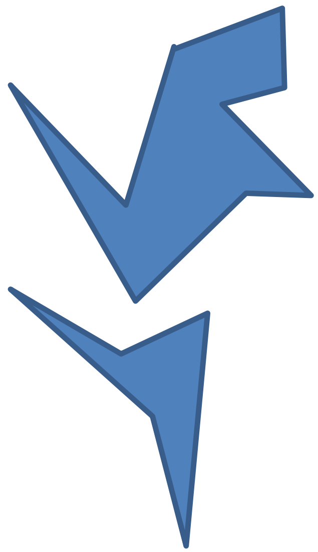
More complex decision boundaries

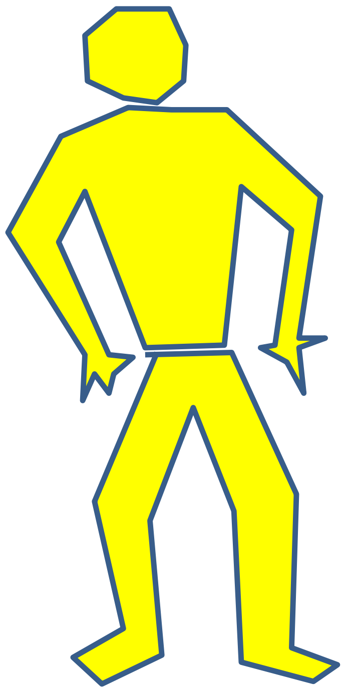


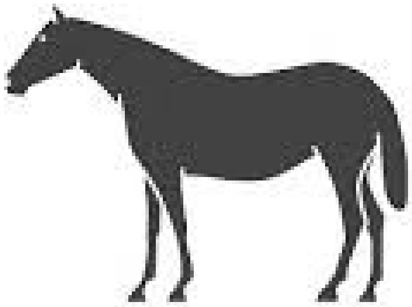


- Network to fire if the input is in the yellow area
  - "OR" two polygons
  - A third layer is required

Complex decision boundaries

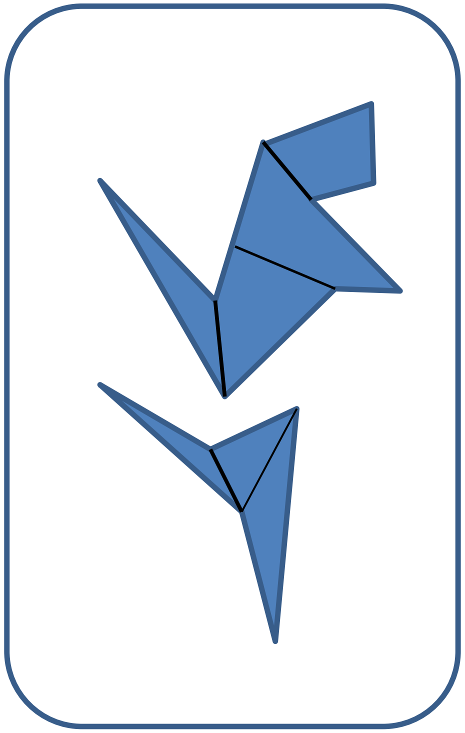


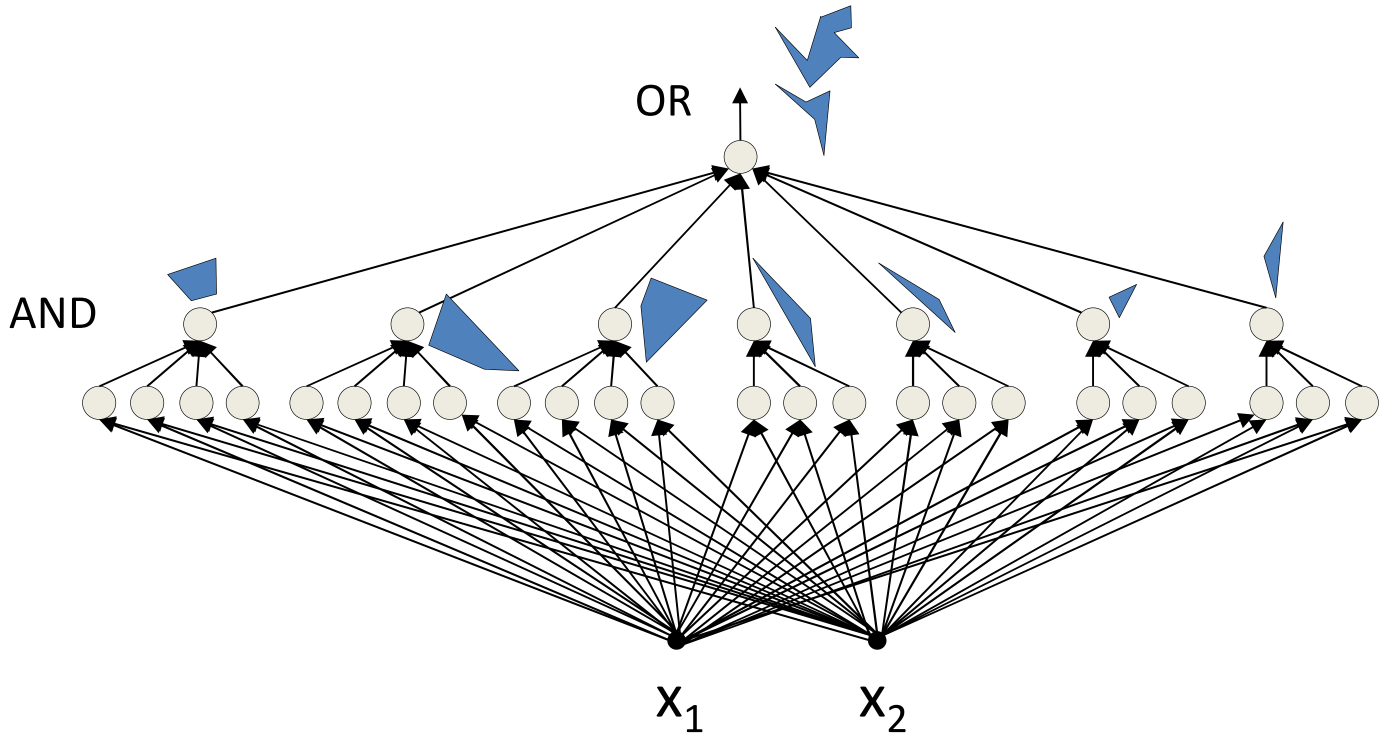




- Can compose *arbitrarily* complex decision boundaries

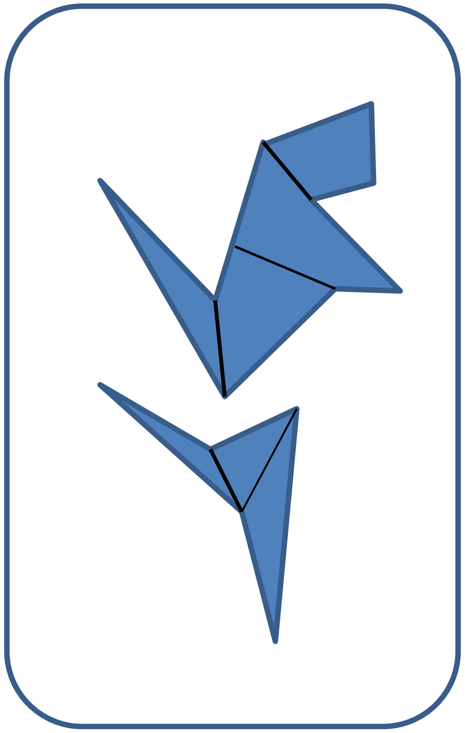
Complex decision boundaries

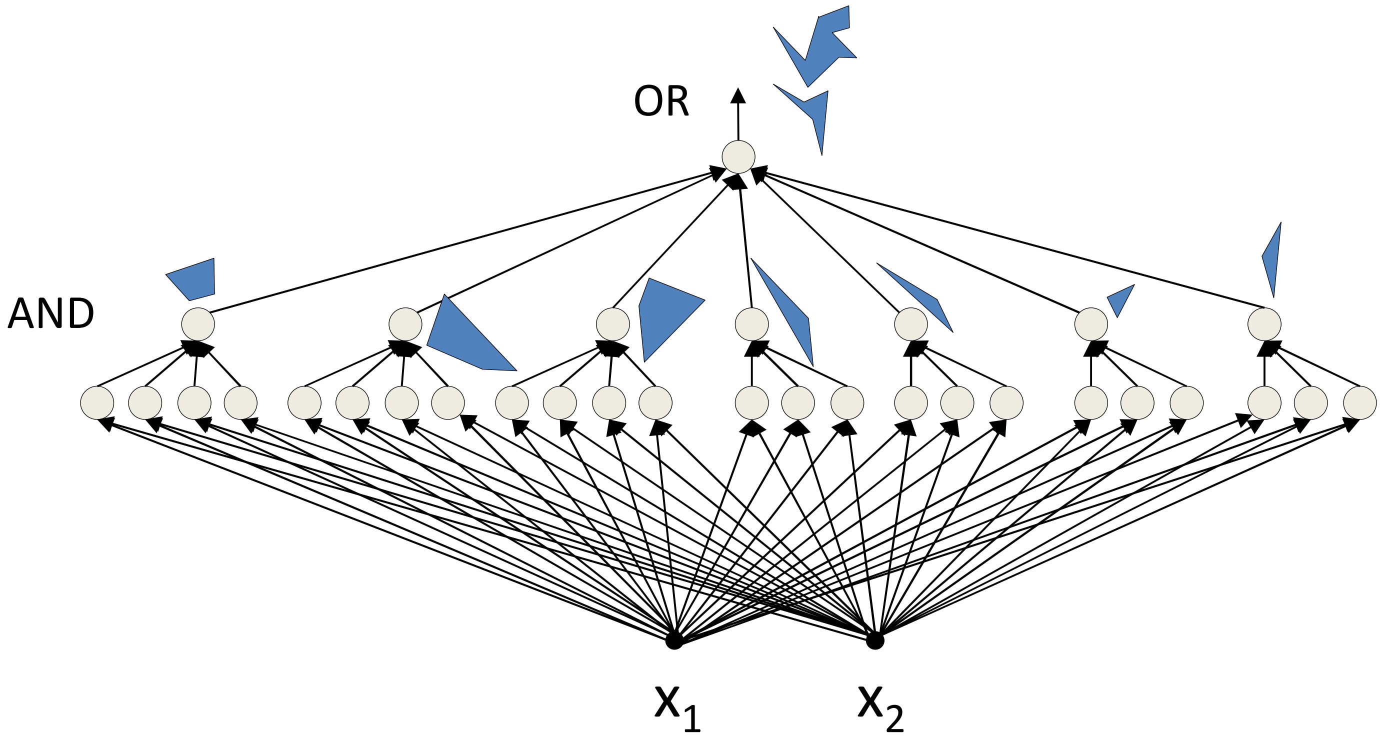




- Can compose *arbitrarily* complex decision boundaries

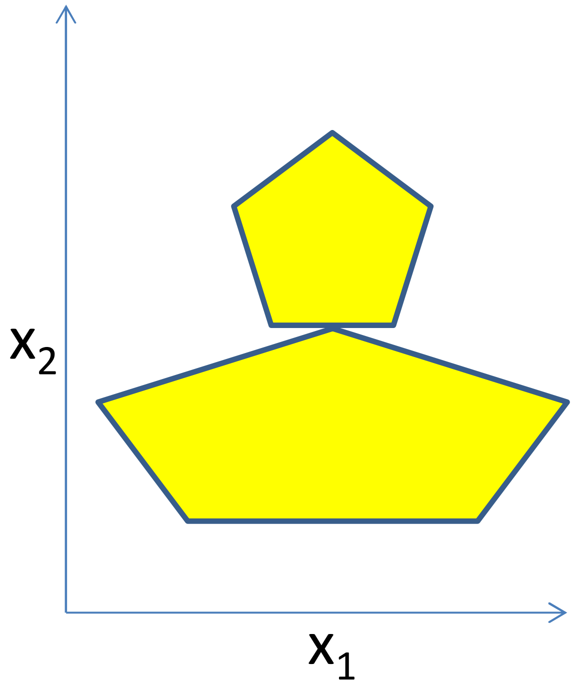
Complex decision boundaries

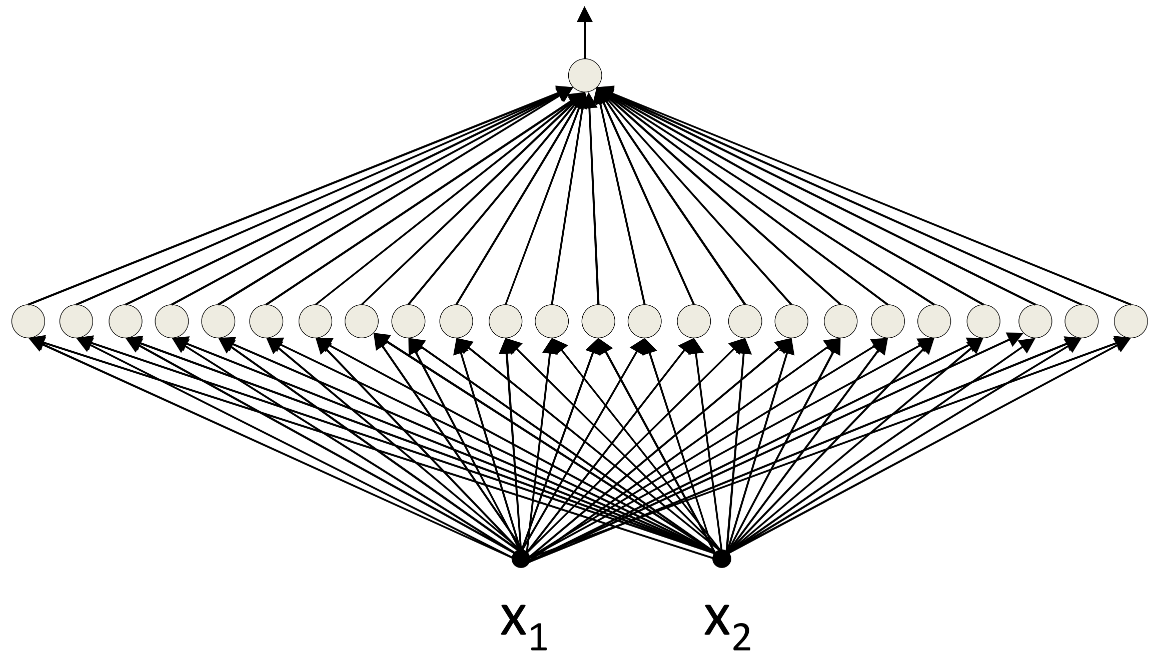




- Can compose *arbitrarily* complex decision boundaries
  - With *only one hidden layer!*
  - **How?**

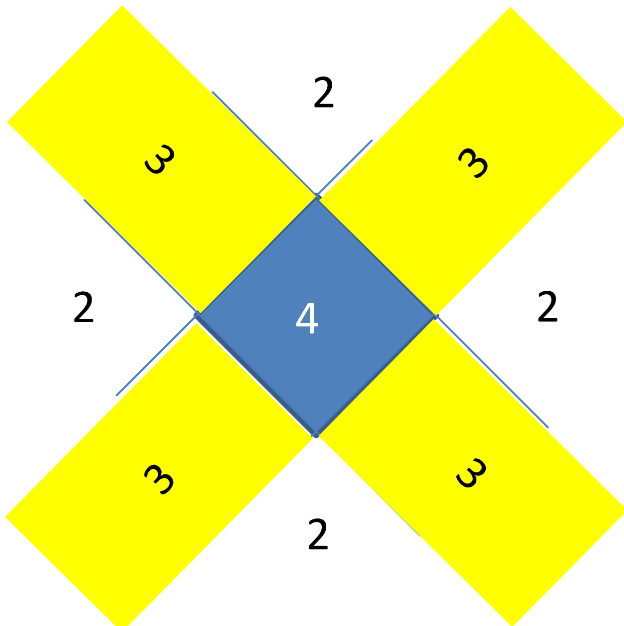
Exercise: compose this with one hidden layer

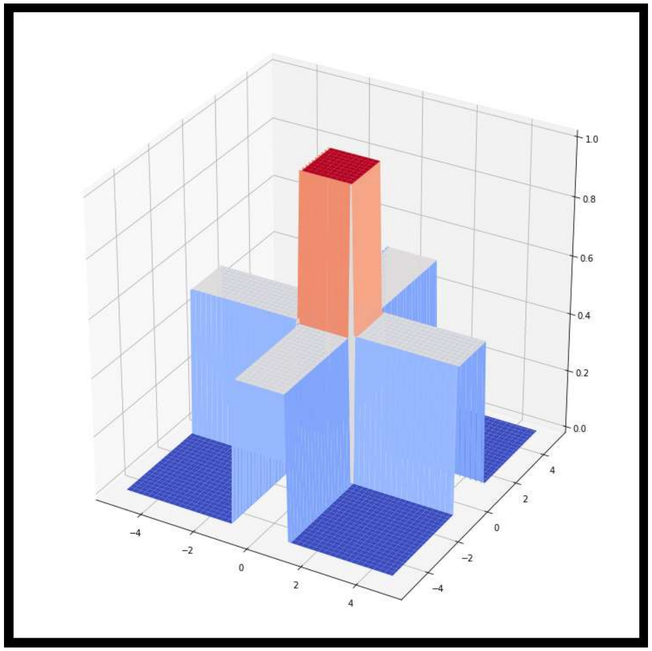




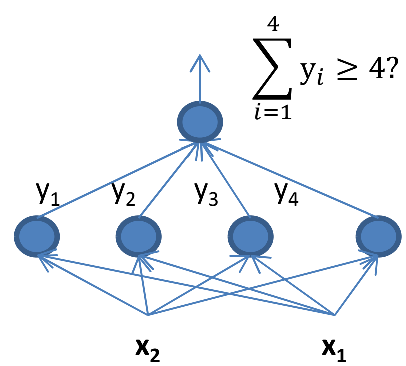
- How would you compose the decision boundary to the left with only *one* hidden layer?

Composing a Square decision boundary

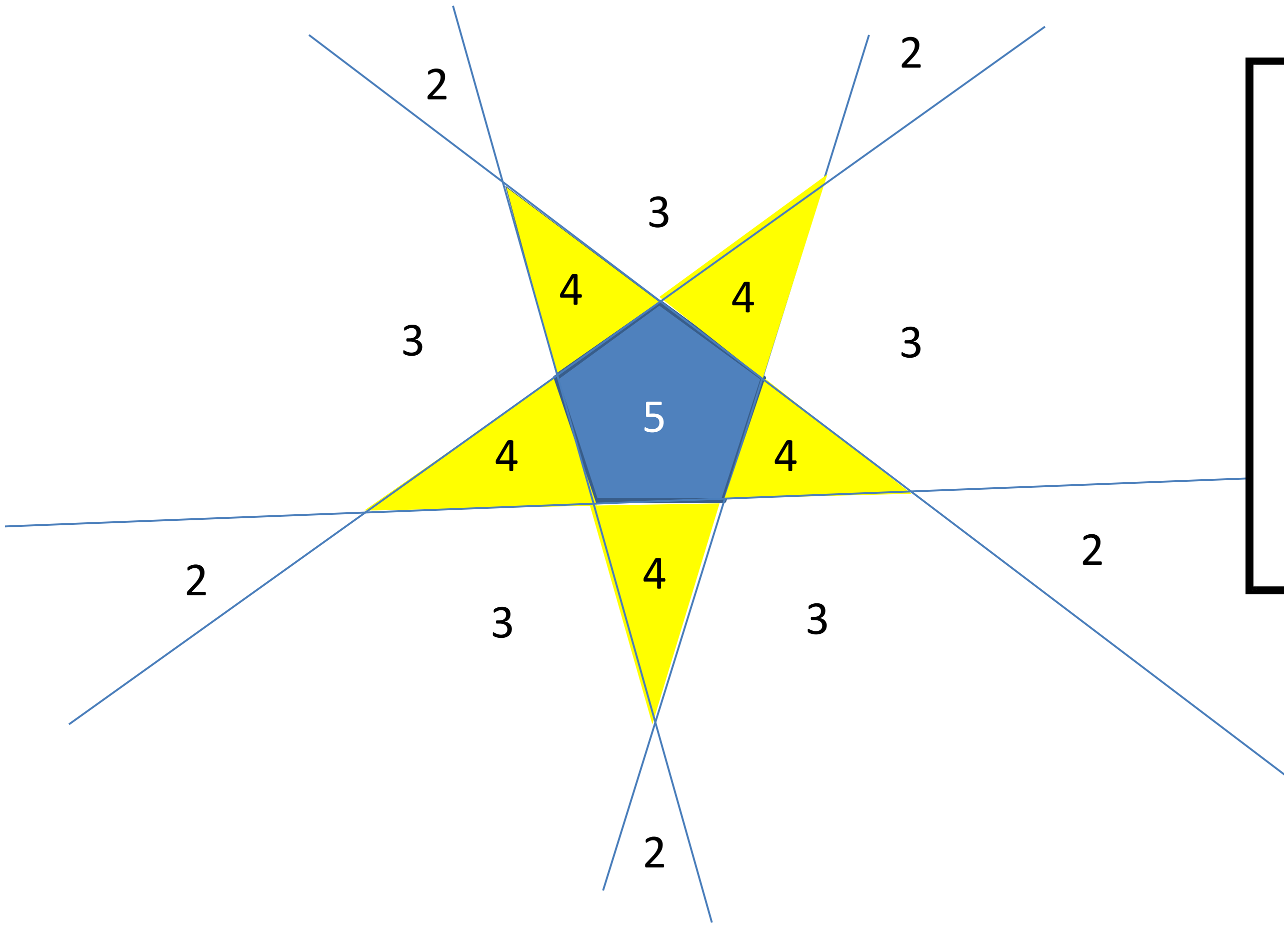


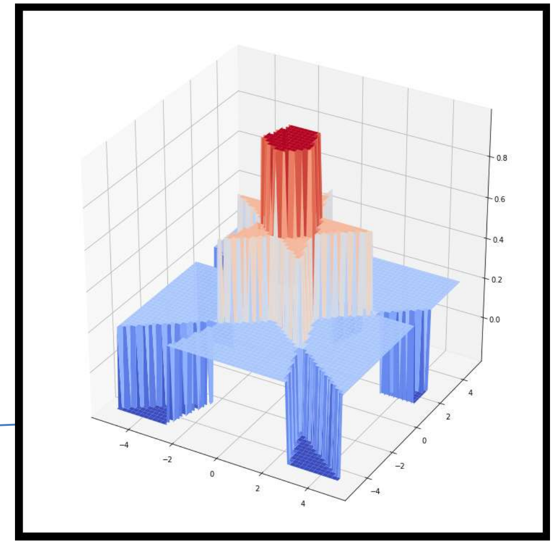


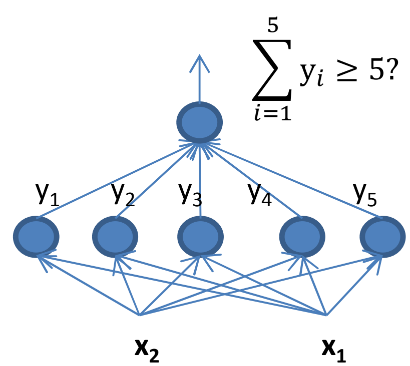
- The polygon net



Composing a pentagon

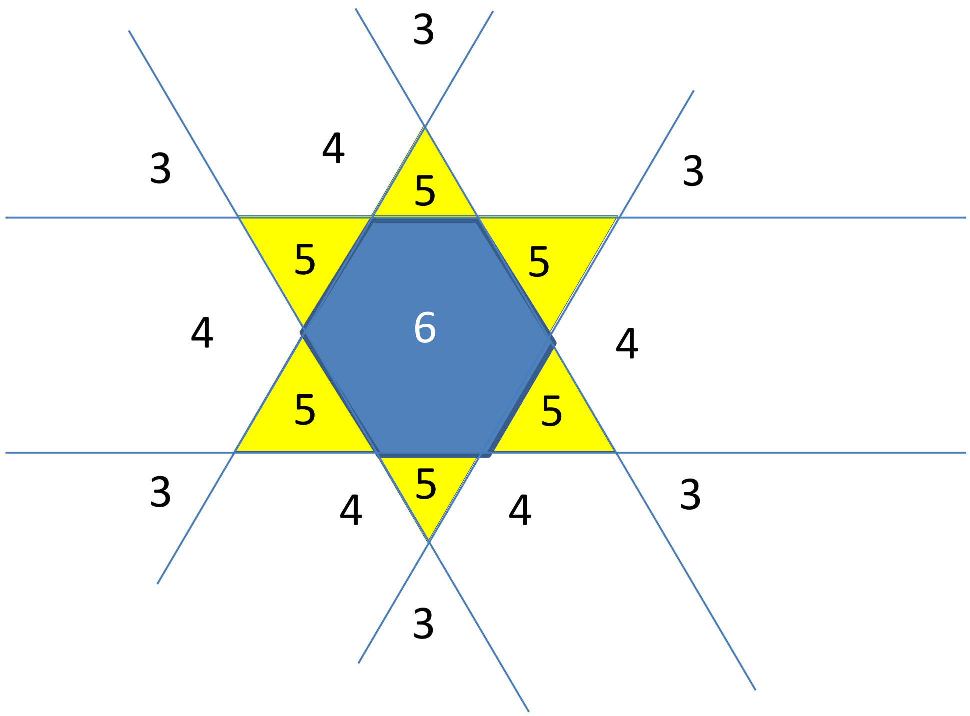


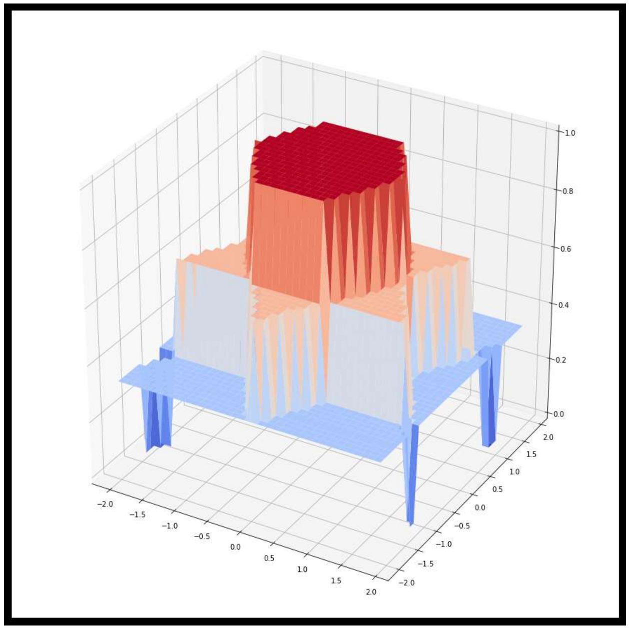


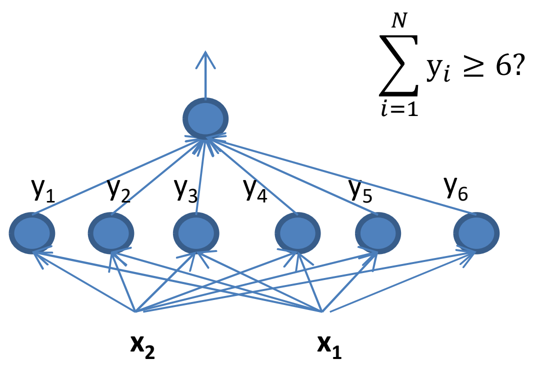


- The polygon net

Composing a hexagon

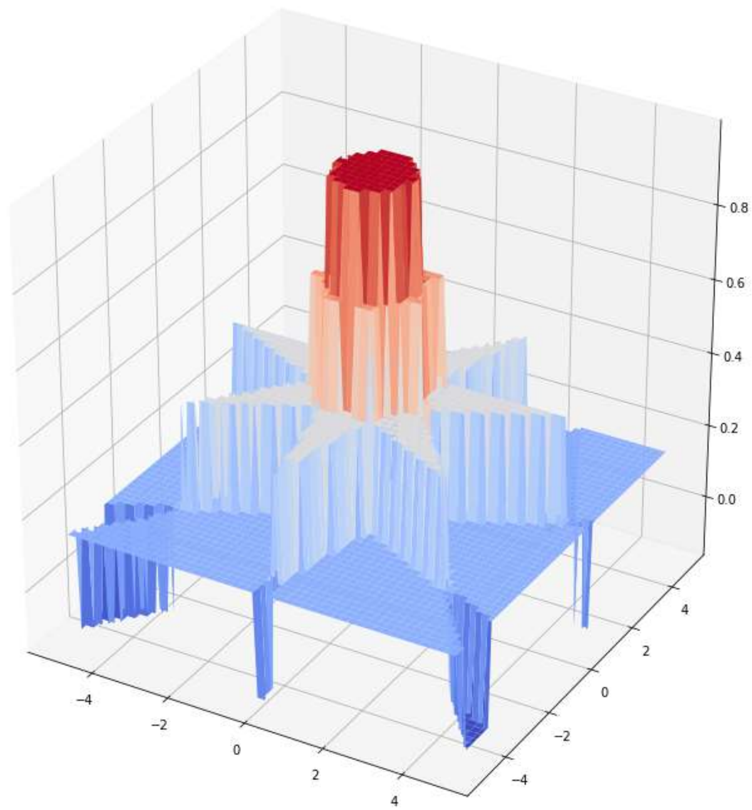






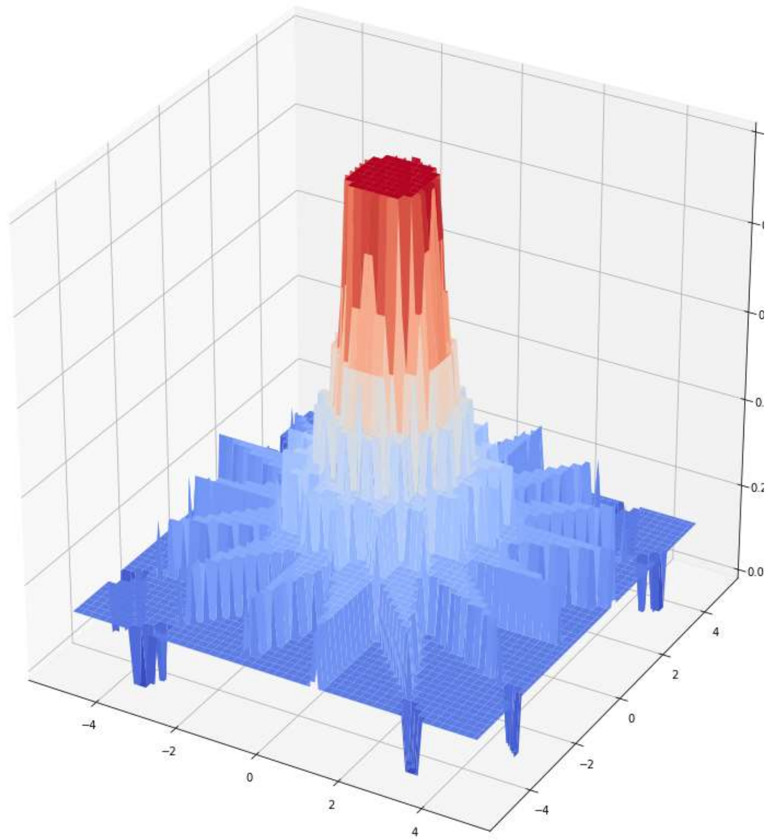
- The polygon net

How about a heptagon



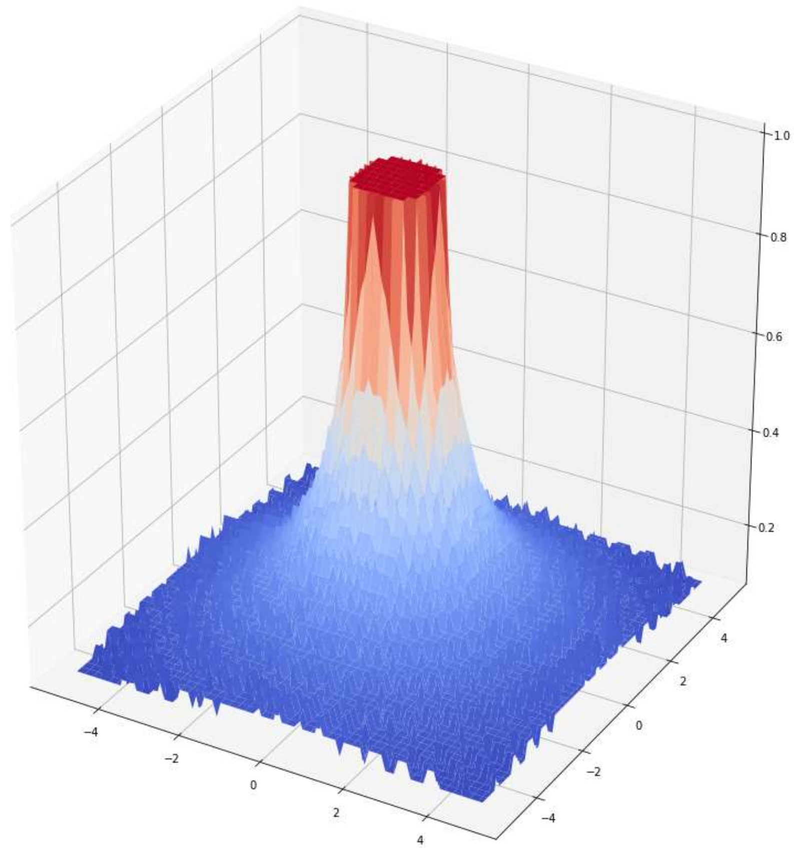
- What are the sums in the different regions?
  – A pattern emerges as we consider N > 6..
    - N is the number of sides of the polygon

16 sides



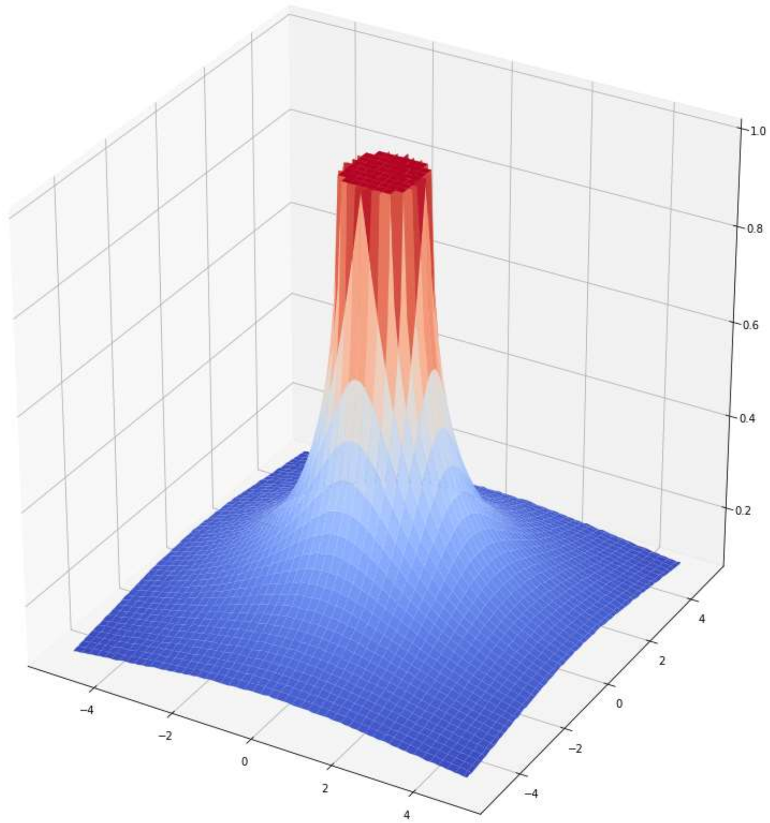
- What are the sums in the different regions?
  - A pattern emerges as we consider N > 6..

64 sides



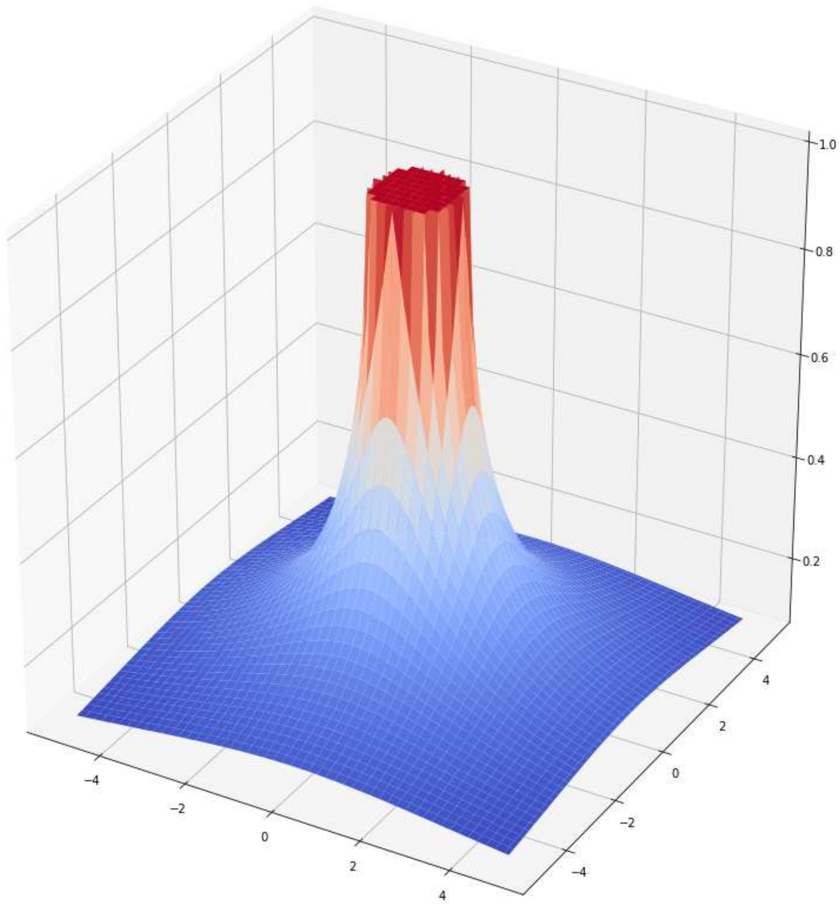
- What are the sums in the different regions?
  - A pattern emerges as we consider N > 6..

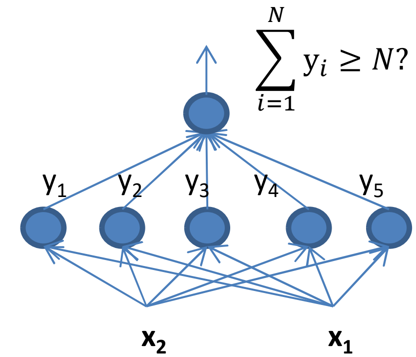
1000 sides



- What are the sums in the different regions?
  - A pattern emerges as we consider N > 6..

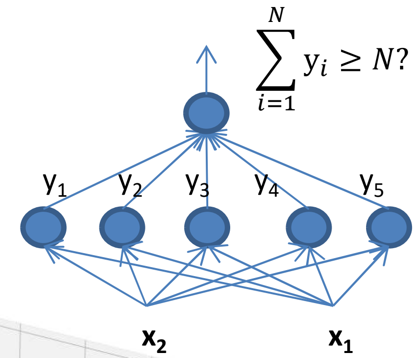
Polygon net

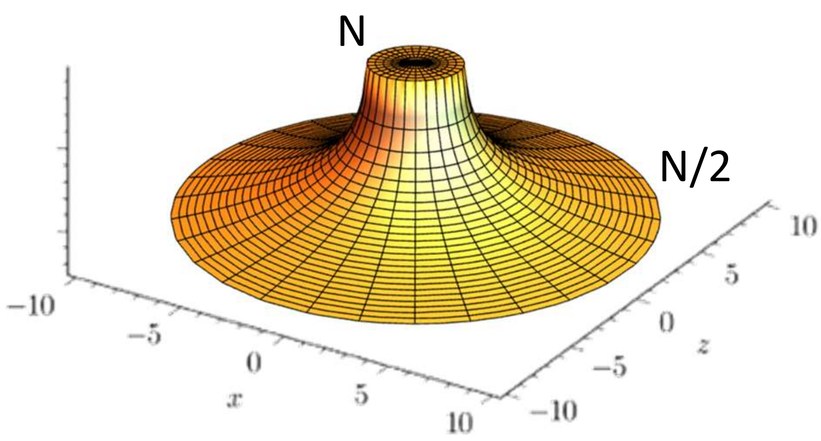


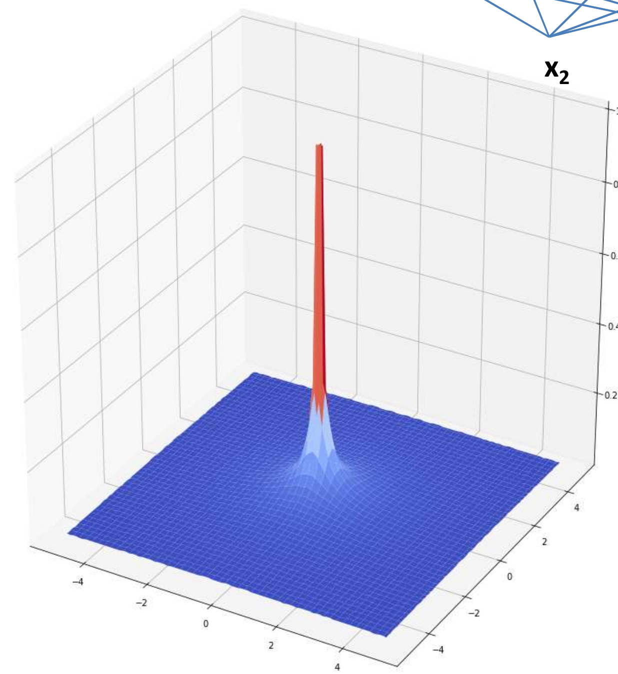


- Increasing the number of sides reduces the area outside the polygon that have $\frac{N}{2} < \sum_i y_i < N$

In the limit

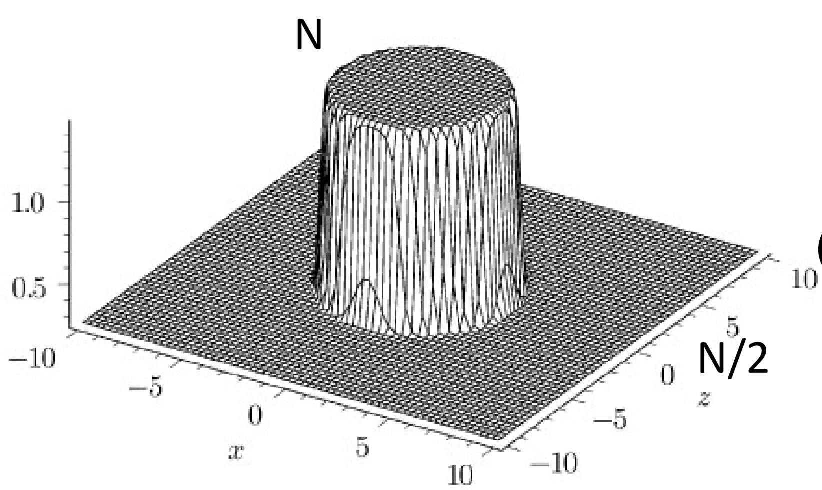


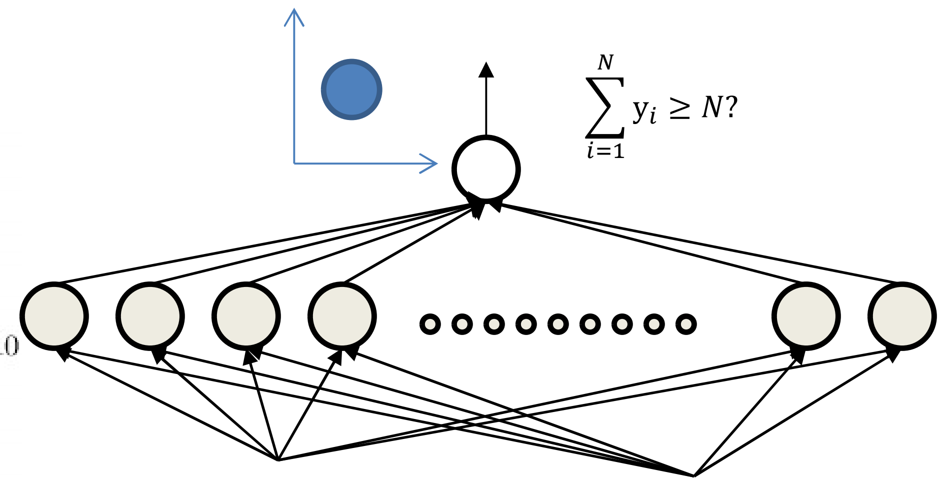




- $\sum_i y_i = N\left(1-\frac{1}{\pi}arccos\left(min\left(1,\frac{radius}{|\mathbf{x}-center|}\right)\right)\right)$
  - Value of the sum at the output unit, as a function of distance from center, as N increases
- For small radius, it's a near perfect cylinder
  - N in the cylinder, N/2 outside

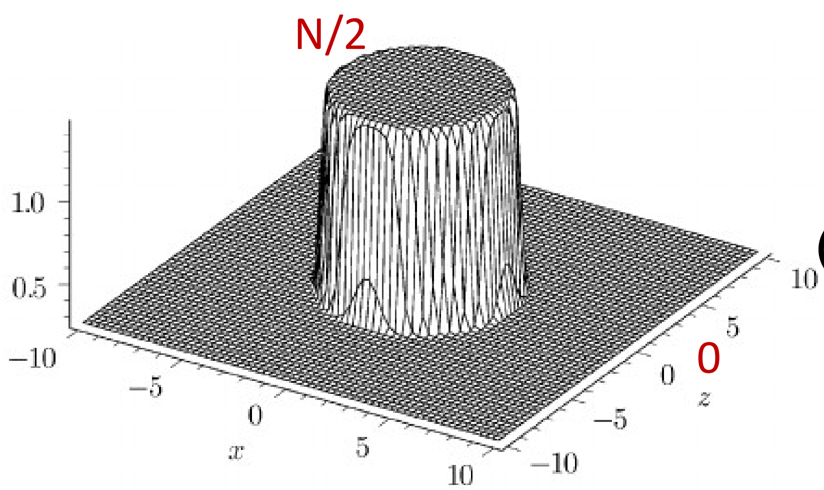
Composing a circle

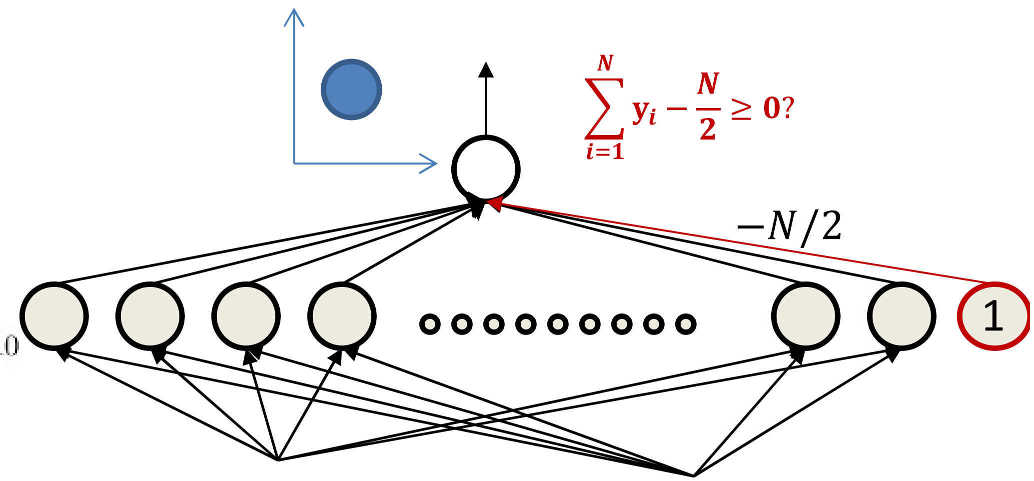




- The circle net
  - Very large number of neurons
  - *Sum is N inside the circle, N/2 outside almost everywhere*
  - Circle can be at any location

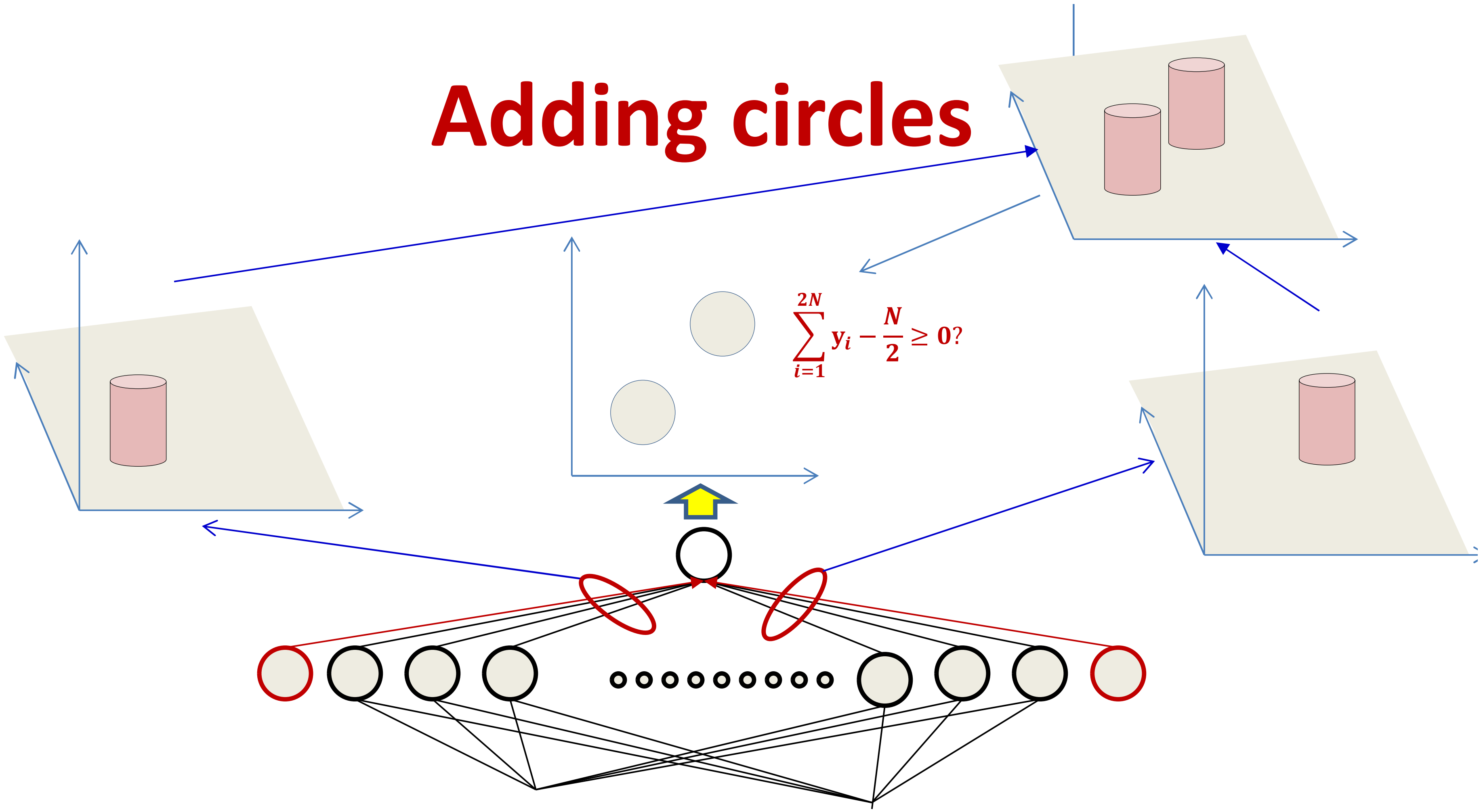
Composing a circle





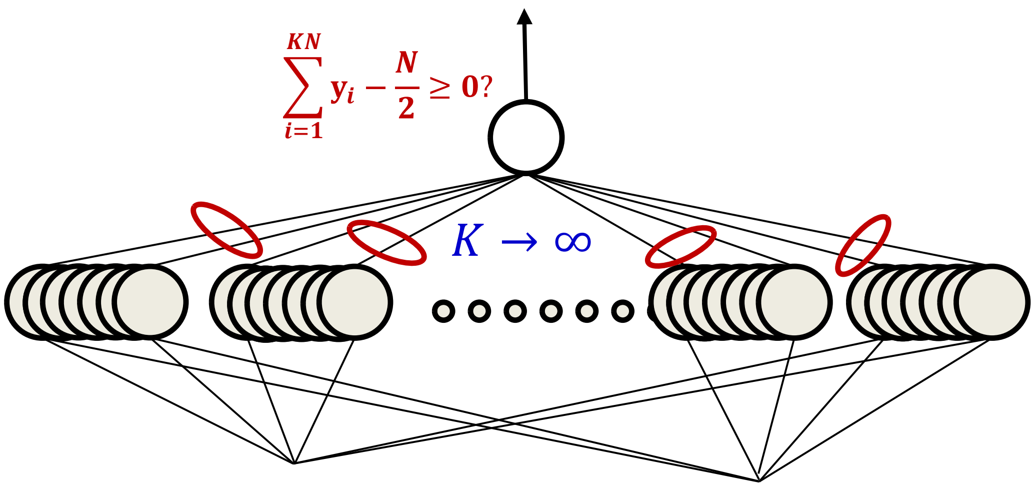
- The circle net
  - Very large number of neurons
  - *Sum is N/2 inside the circle, 0 outside almost everywhere*
  - Circle can be at any location

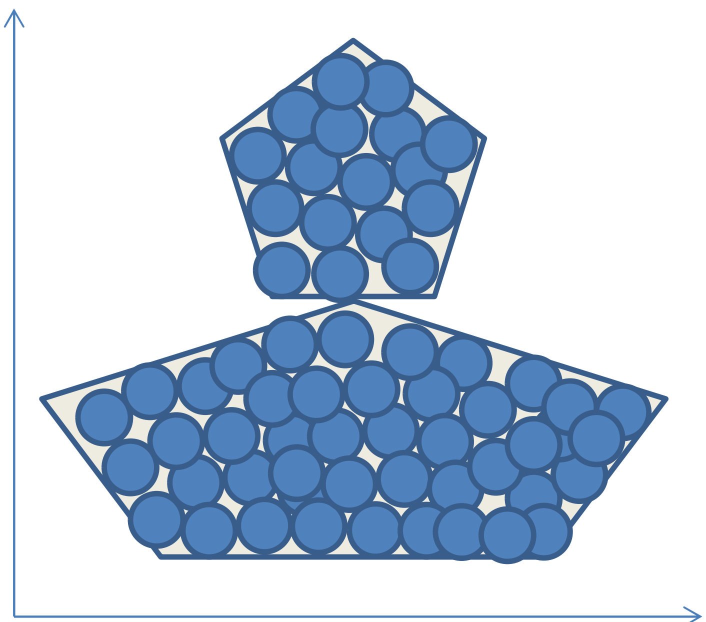
Adding circles



- The "sum" of two circles sub nets is exactly N/2 inside either circle, and 0 almost everywhere outside

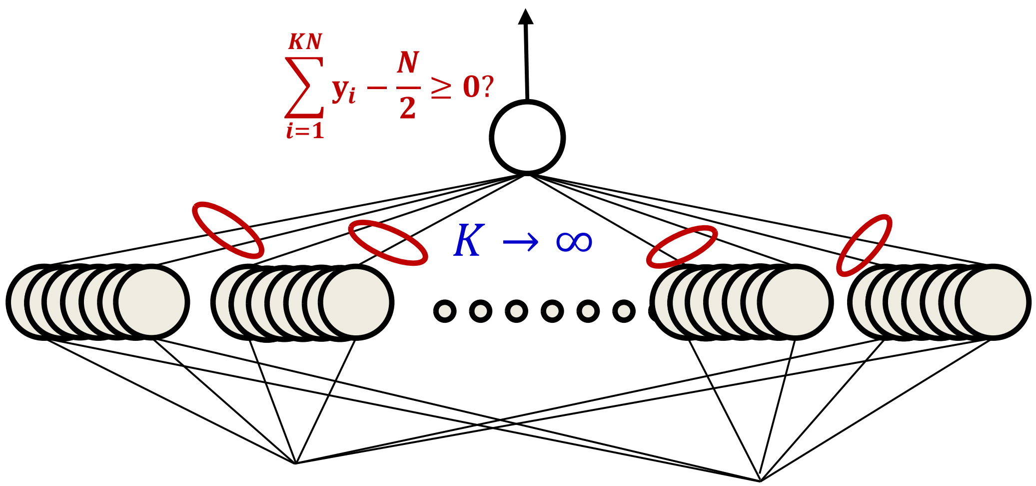
Composing an arbitrary figure

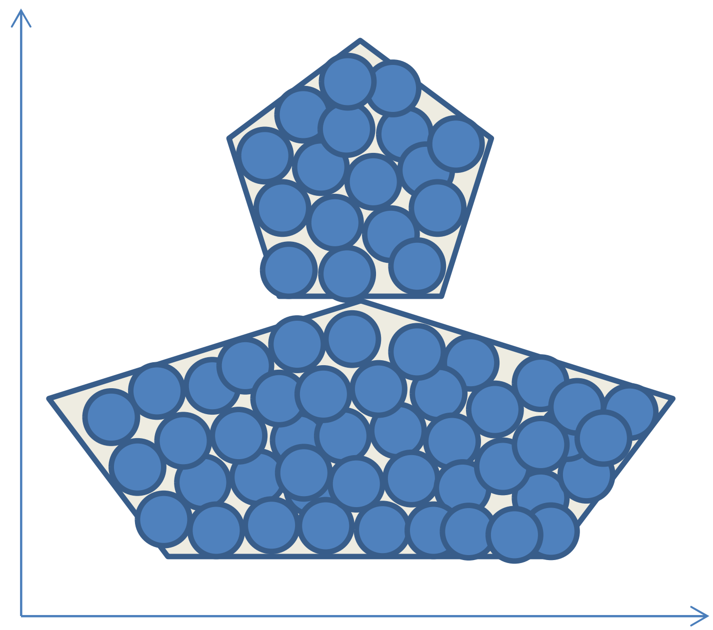




- Just fit in an arbitrary number of circles
  - More accurate approximation with greater number of smaller circles
  - Can achieve arbitrary precision

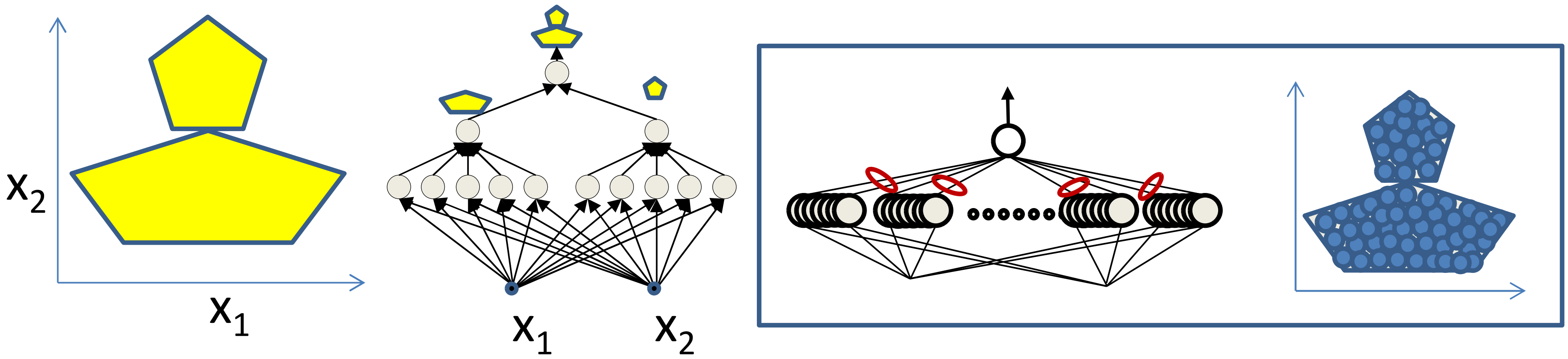
MLP: Universal classifier





- MLPs can capture *any* classification boundary
- A *one-hidden-layer MLP* can model any classification boundary
- *MLPs are universal classifiers*

Depth and the universal classifier



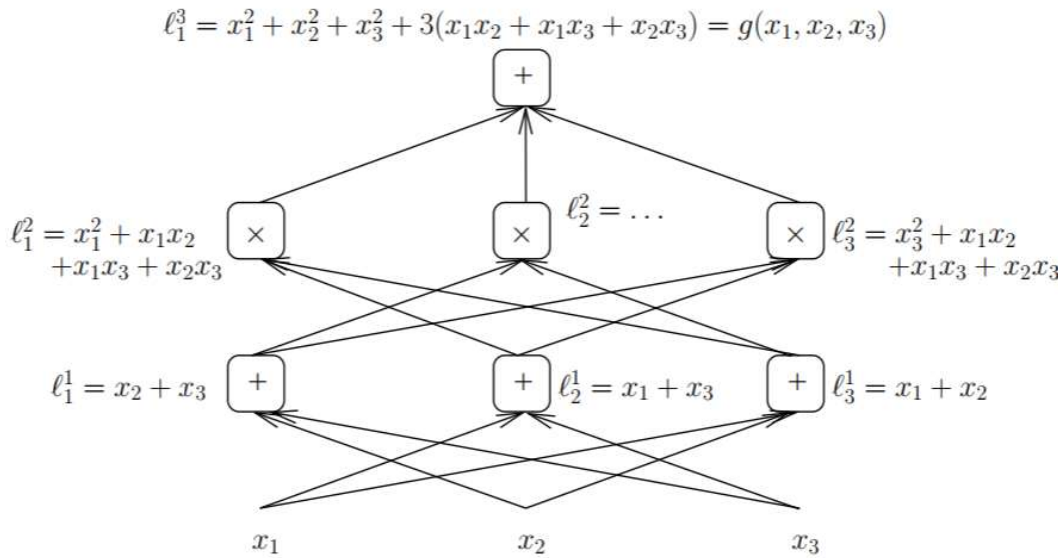
- Deeper networks can require far fewer neurons

Optimal depth..

- Formal analyses typically view these as category of *arithmetic circuits*
  - Compute polynomials over any field
    - Valiant et. al: A polynomial of degree n requires a network of depth $log^2(n)$
      - Cannot be computed with shallower networks
      - The majority of functions are very high (possibly ∞) order polynomials
    - Bengio et. al: Shows a similar result for sum-product networks
      - But only considers two-input units
      - Generalized by Mhaskar et al. to all functions that can be expressed as a binary tree

  - Depth/Size analyses of arithmetic circuits still a research problem

Special case: Sum-product nets



- "Shallow vs deep sum-product networks," Oliver Dellaleau and Yoshua Bengio
  - For networks where layers alternately perform either sums or products, a deep network may require an exponentially fewer number of layers than a shallow one

Depth in sum-product networks

Theorem 5

A certain class of functions $\mathcal{F}$ of n inputs can be represented using a deep network with $\mathcal{O}(n)$ units, whereas it would require $\mathcal{O}(2^{\sqrt{n}})$ units for a shallow network.

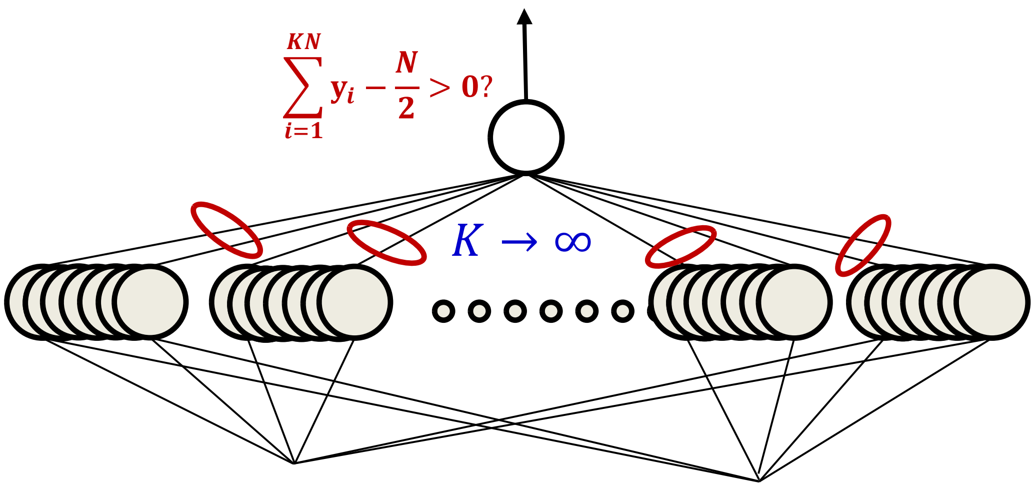
Theorem 6

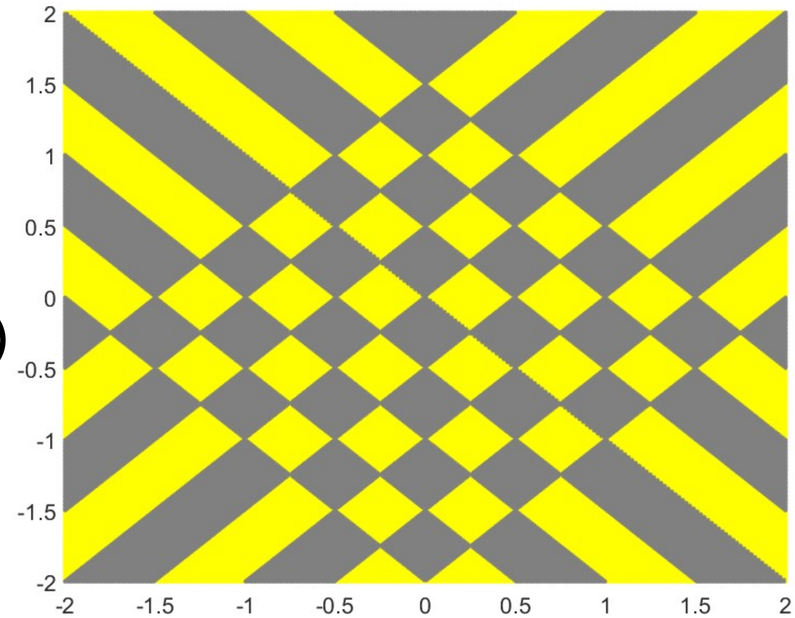
For a certain class of functions $\mathcal{G}$ of n inputs, the deep sum-product network with depth k can be represented with $\mathcal{O}(nk)$ units, whereas it would require $\mathcal{O}((n-1)^k)$ units for a shallow network.

Optimal depth in *generic* nets

- We look at a different pattern:
  - "worst case" decision boundaries
- For *threshold-activation* networks
  - Generalizes to other nets

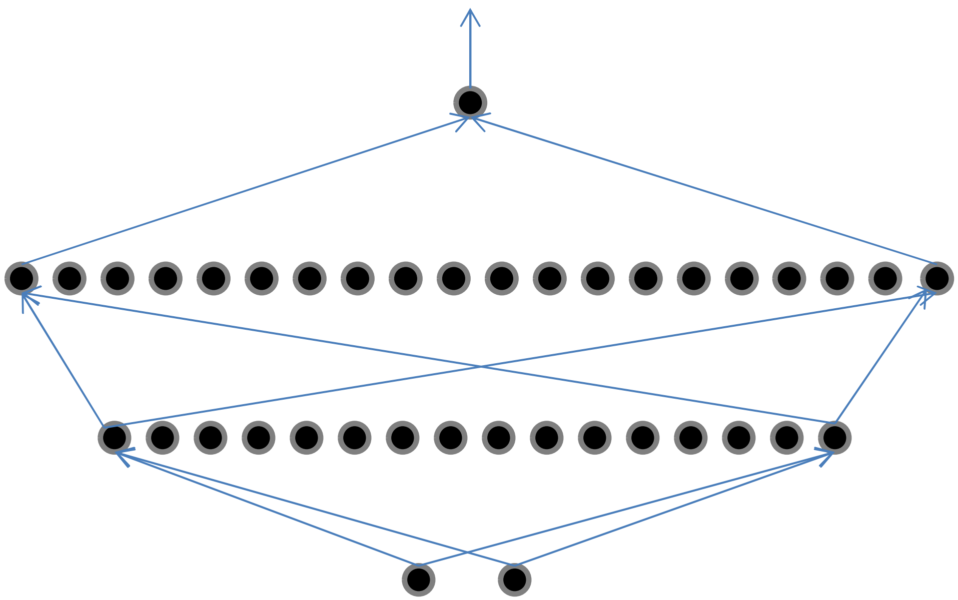
Optimal depth

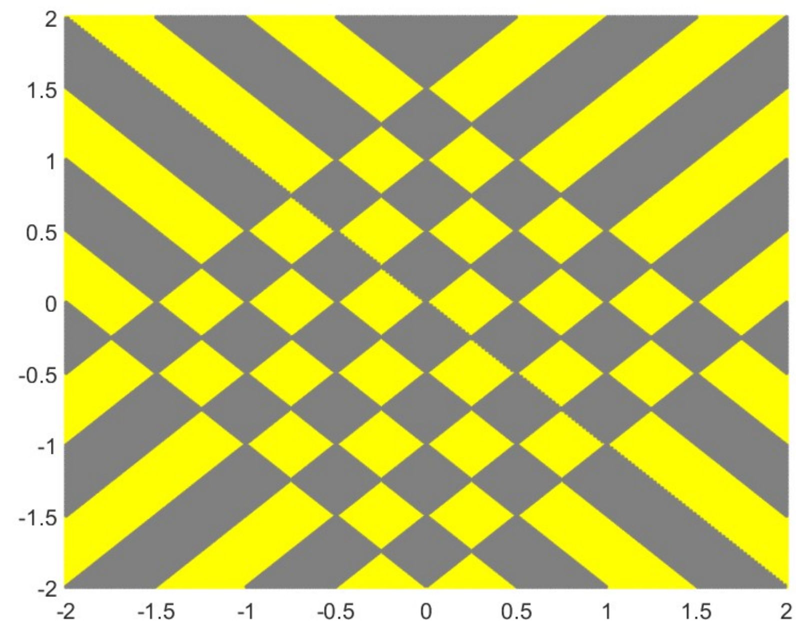




- A naïve one-hidden-layer neural network will require infinite hidden neurons

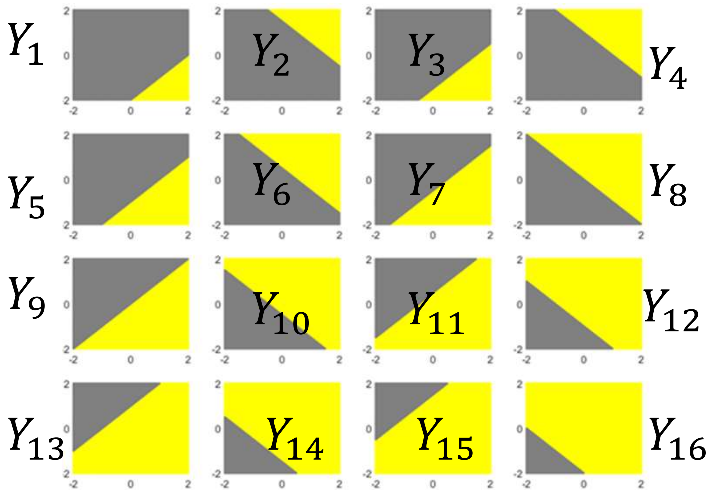
Optimal depth

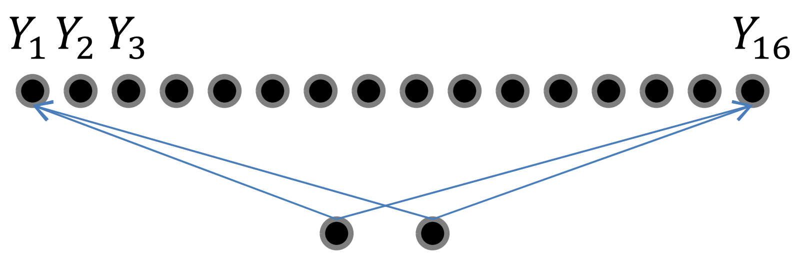


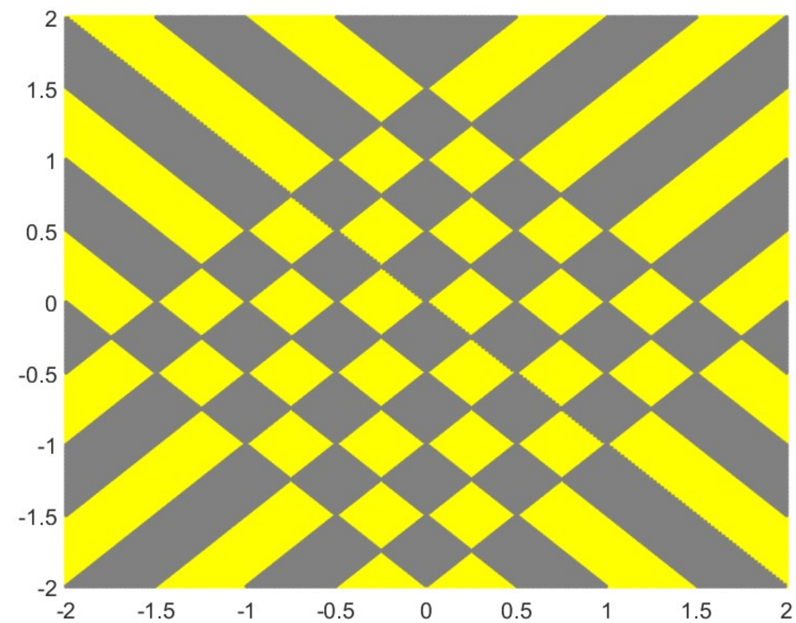


- Two hidden-layer network: 56 hidden neurons

Optimal depth

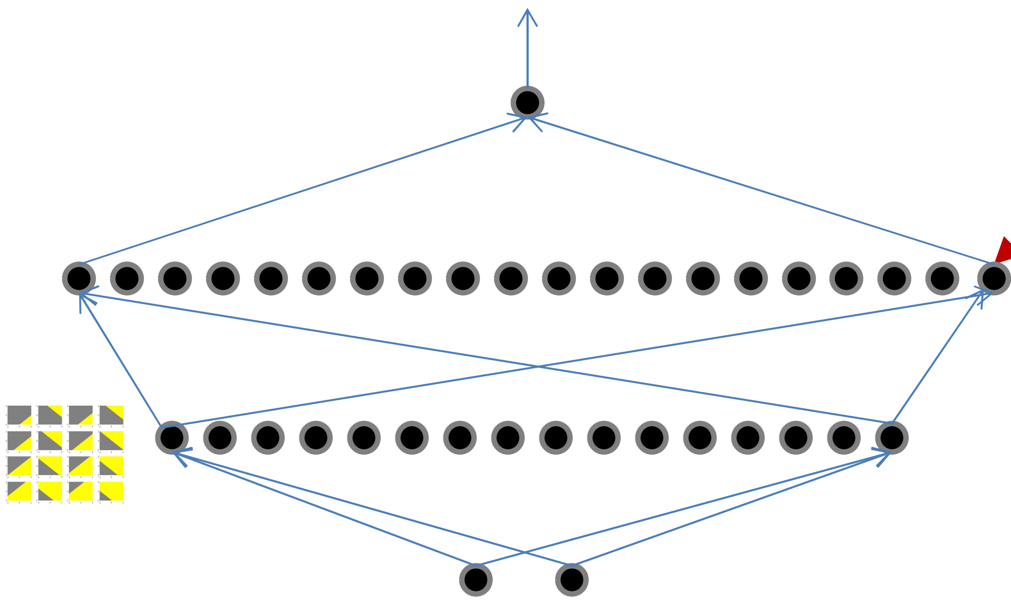


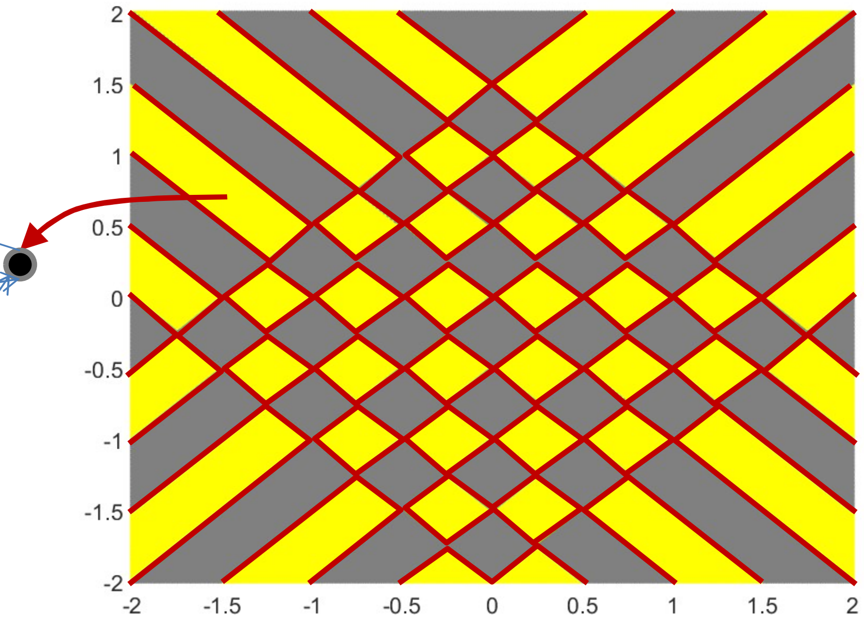




- Two-hidden-layer network: 56 hidden neurons
  - 16 neurons in hidden layer 1

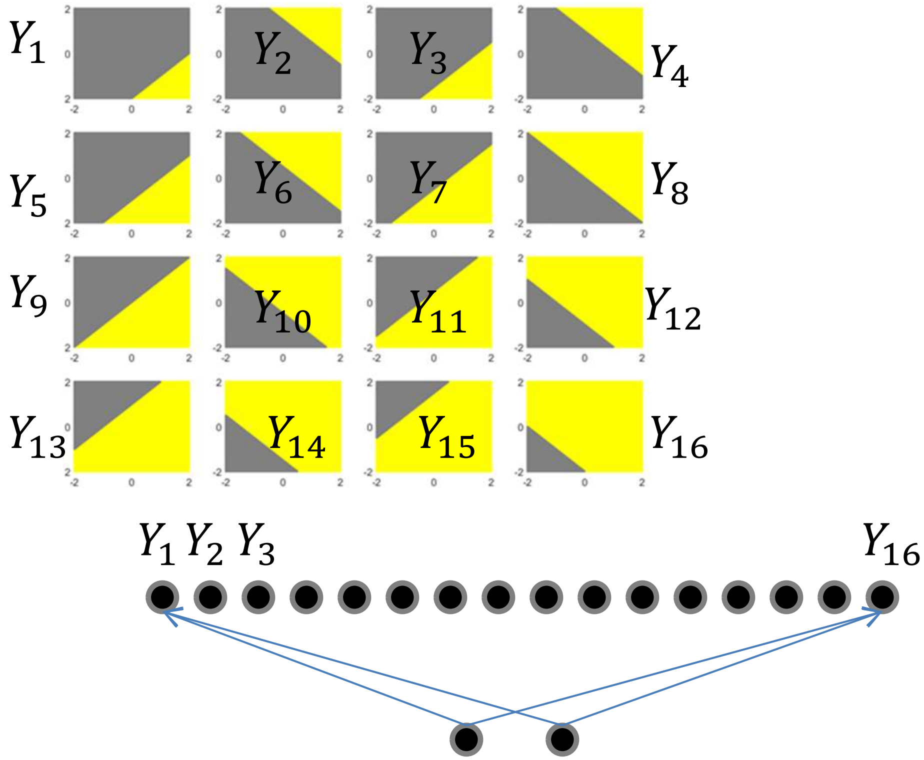
Optimal depth

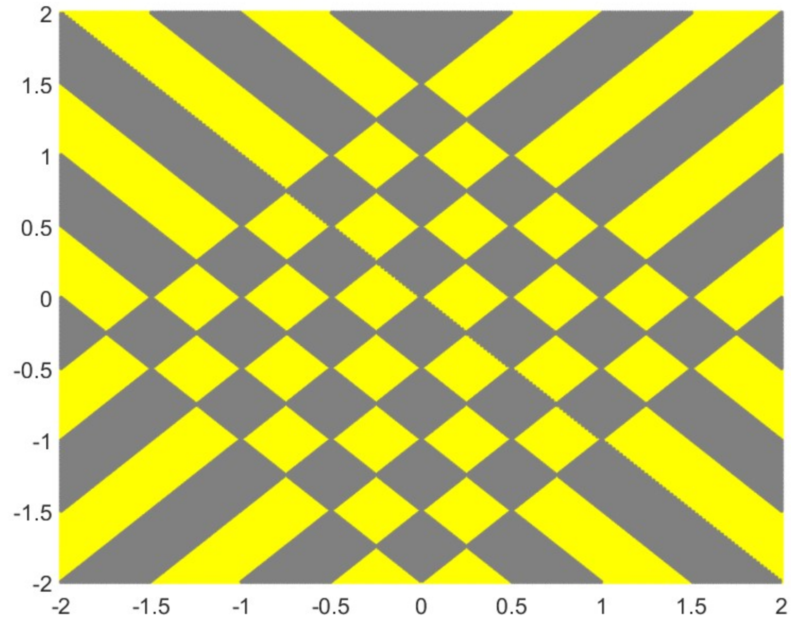




- Two-hidden-layer network: 56 hidden neurons
  - 16 in hidden layer 1
  - 40 in hidden layer 2
  - 57 total neurons, including output neuron

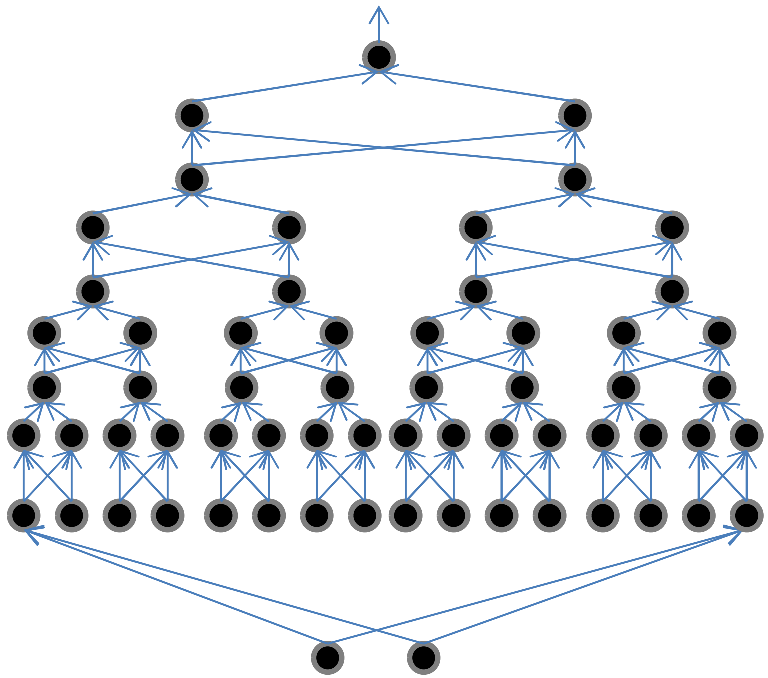
Optimal depth

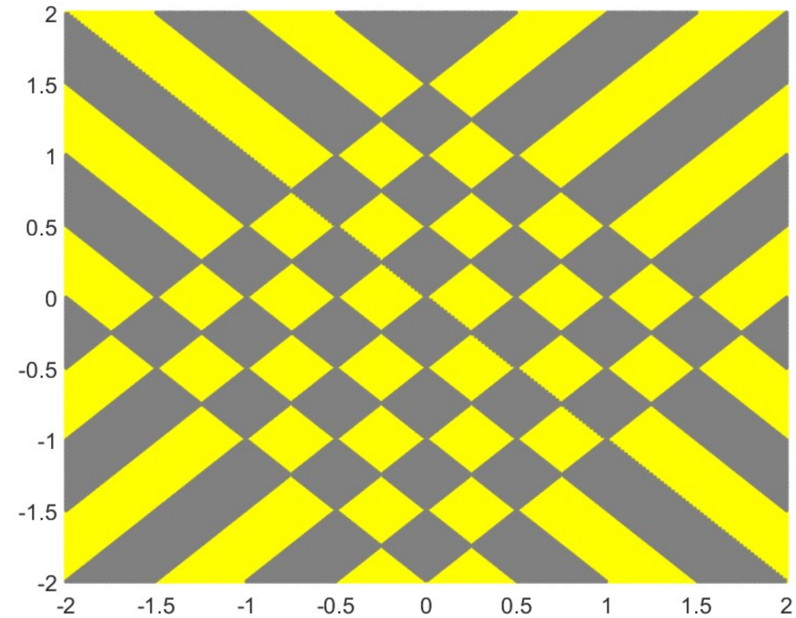




- But this is just $Y_1 \oplus Y_2 \oplus \cdots \oplus Y_{16}$

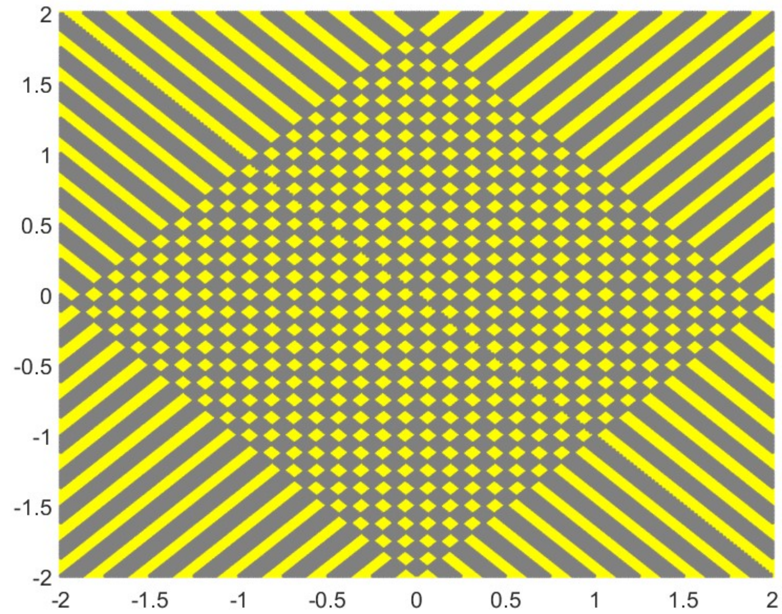
Optimal depth





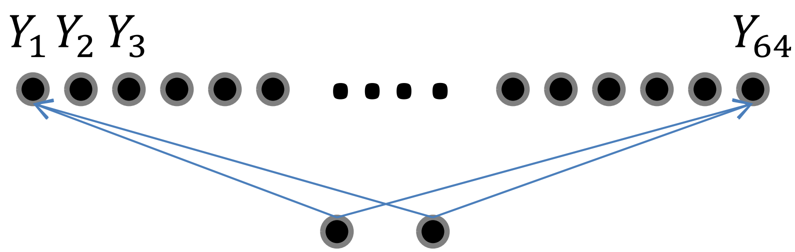
- But this is just $Y_1 \oplus Y_2 \oplus \cdots \oplus Y_{16}$
  - The XOR net will require 16 + 15x3 = 61 neurons
    - 46 neurons if we use a two-neuron XOR model

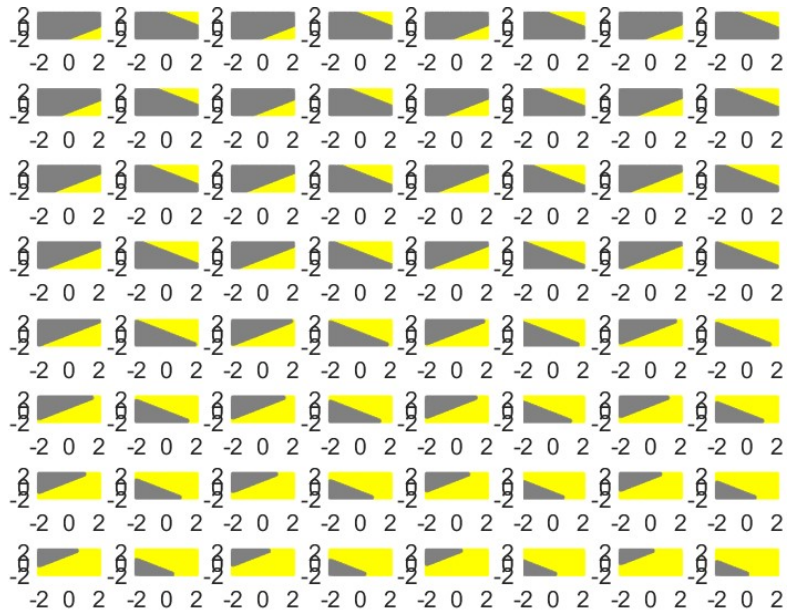
Optimal depth



- Grid formed from 64 lines
  - Network must output 1 for inputs in the yellow regions

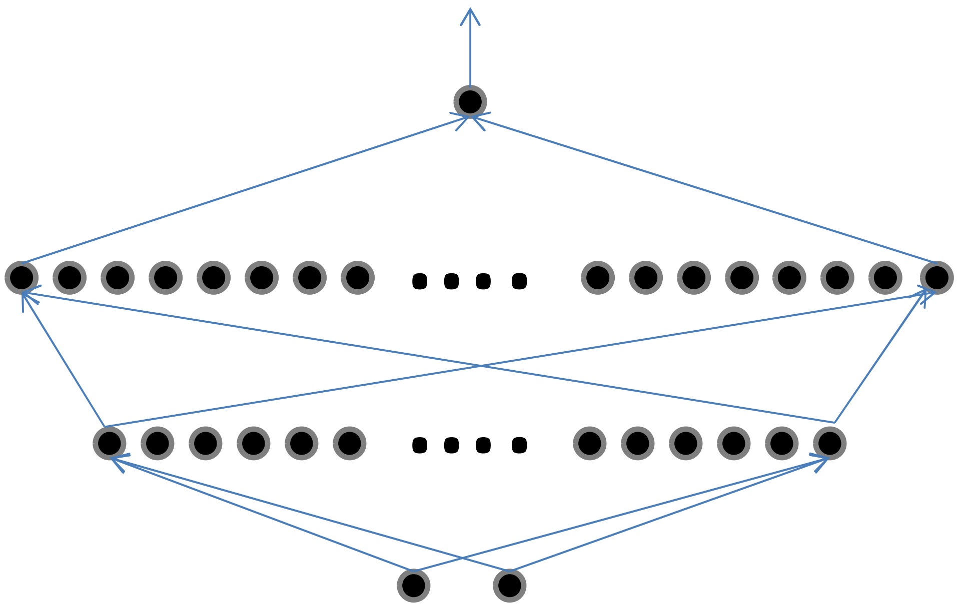
Actual linear units

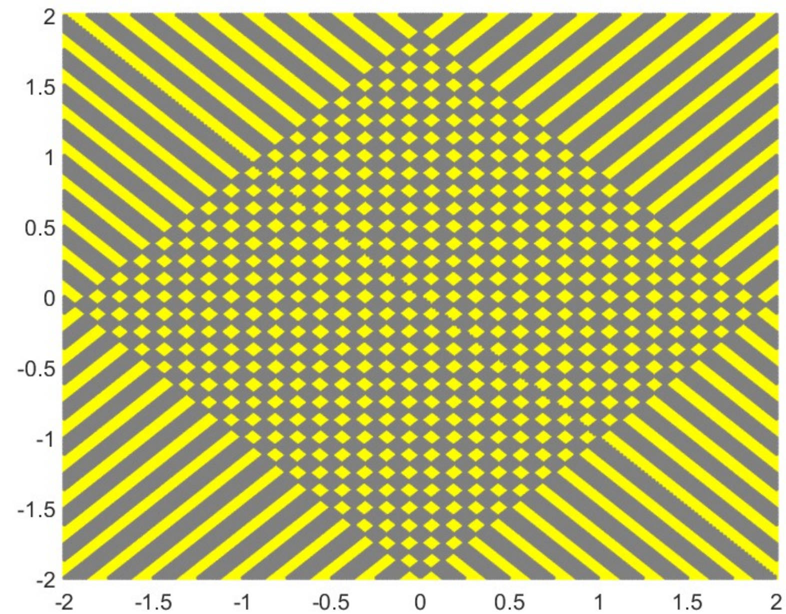




- 64 basic linear feature detectors

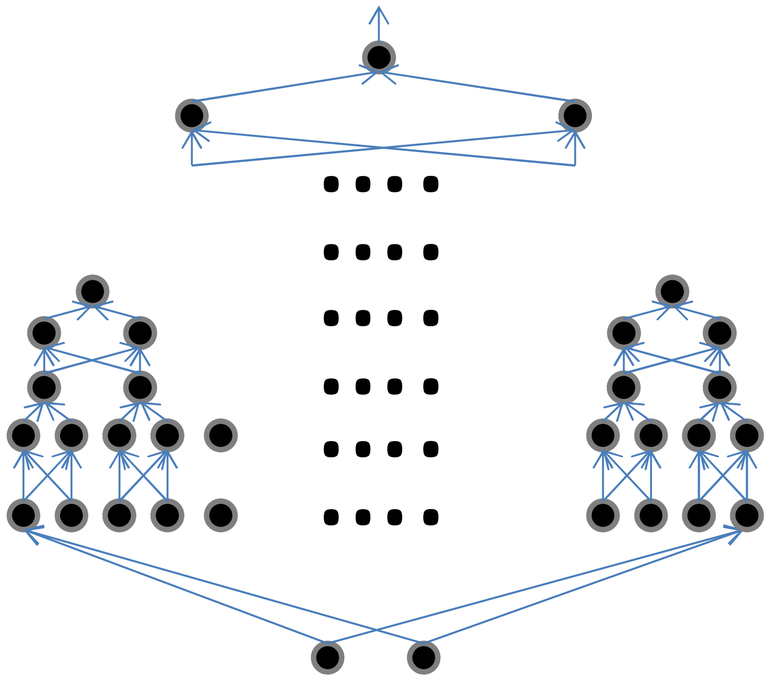
Optimal depth

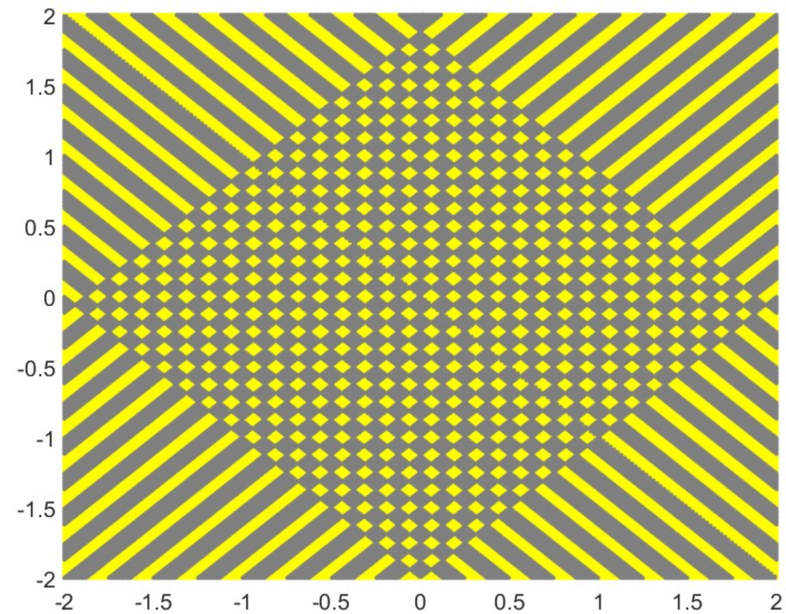




- Two hidden layers: 608 hidden neurons
  - 64 in layer 1
  - 544 in layer 2
- 609 total neurons (including output neuron)

Optimal depth

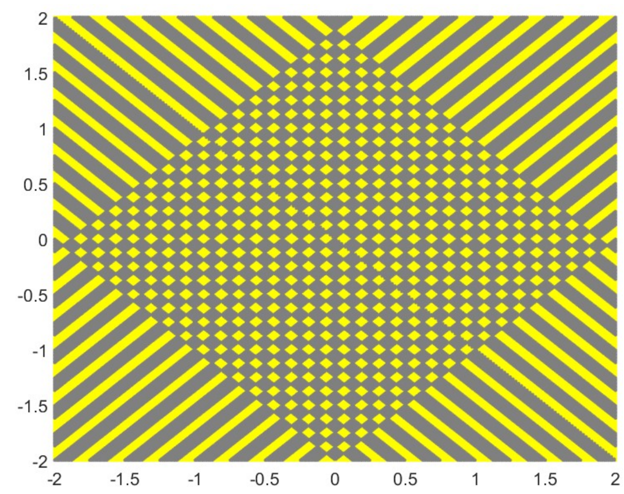




- XOR network (12 hidden layers): 253 neurons
  - 190 neurons with 2-gate XOR
- The difference in size between the deeper optimal (XOR) net and shallower nets increases with increasing pattern complexity and input dimension

Network size?

- In this problem the 2-layer net was *quadratic* in the number of lines
  - $\lfloor (N+2)^2/8 \rfloor$ neurons in 2nd hidden layer
  - Not exponential
  - Even though the pattern is an XOR
  - Why?



- The data are two-dimensional!
  - Only two *fully independent* features
  - The pattern is exponential in the *dimension of the input (two)!*
- For general case of N mutually intersecting hyperplanes in D dimensions, we will need $\mathcal{O}\left(\frac{N^D}{(D-1)!}\right)$ weights (assuming $N \gg D$).
  - Increasing input dimensions can increase the worst-case size of the shallower network exponentially, but not the XOR net
    - The size of the XOR net depends only on the number of first-level linear detectors (N)

Depth: Summary

- The number of neurons required in a shallow network is potentially exponential in the dimensionality of the input
  - (this is the worst case)
  - Alternately, exponential in the number of statistically independent features

Story so far

- Multi-layer perceptrons are *Universal Boolean Machines*
  - Even a network with a *single* hidden layer is a universal Boolean machine

- Multi-layer perceptrons are *Universal Classification Functions*
  - Even a network with a single hidden layer is a universal classifier

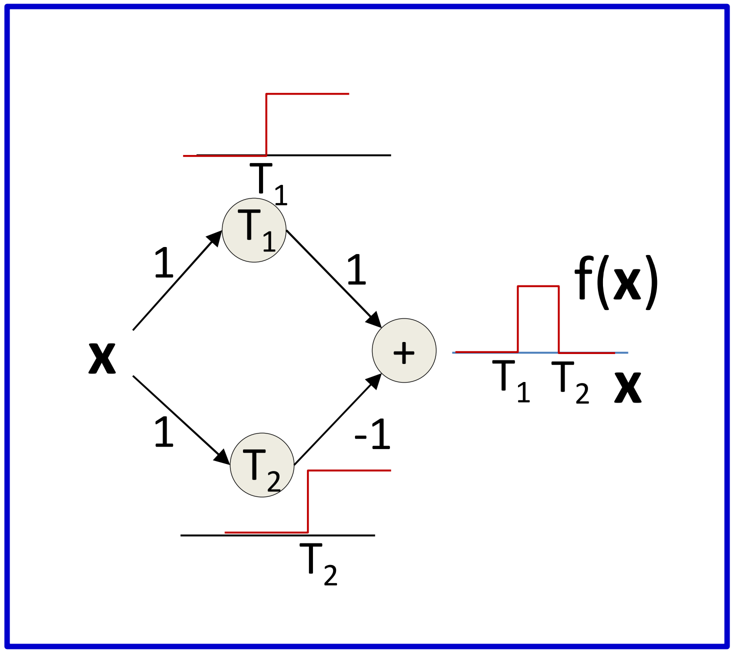
- But a single-layer network may require an exponentially large number of perceptrons than a deep one

- Deeper networks may require far fewer neurons than shallower networks to express the same function
  - Could be *exponentially* smaller
  - Deeper networks are more *expressive*

Today

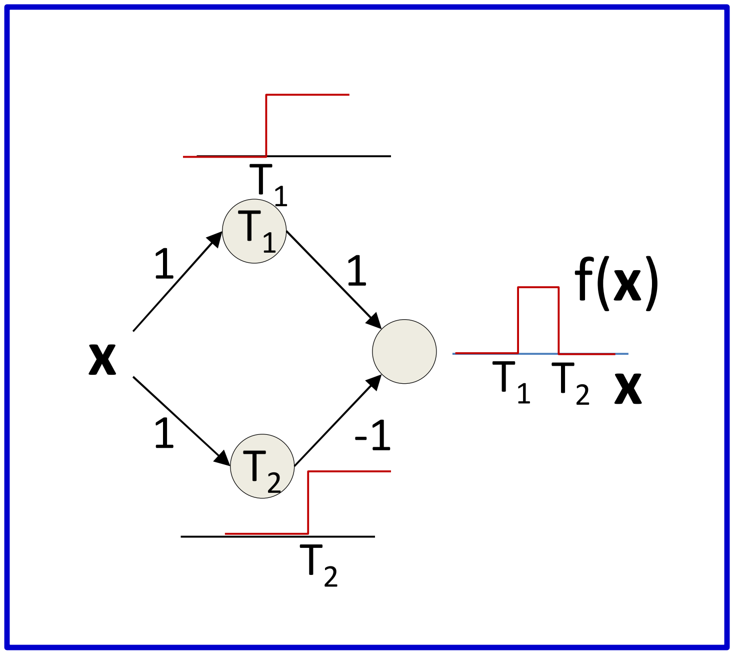
- Multi-layer Perceptrons as universal Boolean functions
  - The need for depth
- MLPs as universal classifiers
  - The need for depth
- MLPs as universal approximators
- A discussion of optimal depth and width
- Brief segue: RBF networks

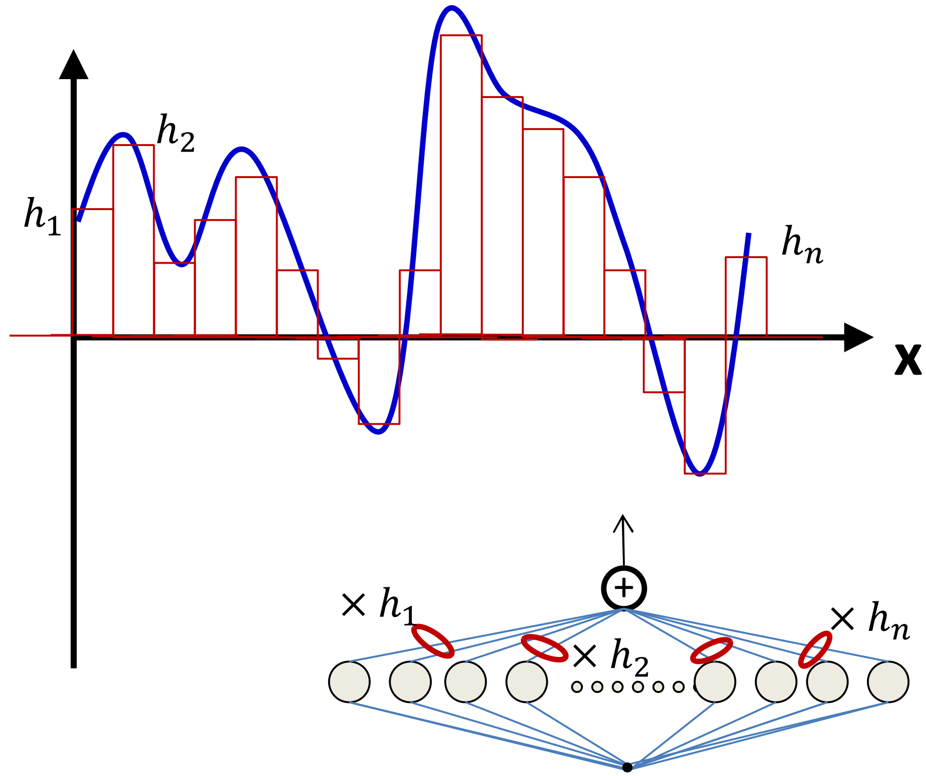
MLP as a continuous-valued regression



- 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

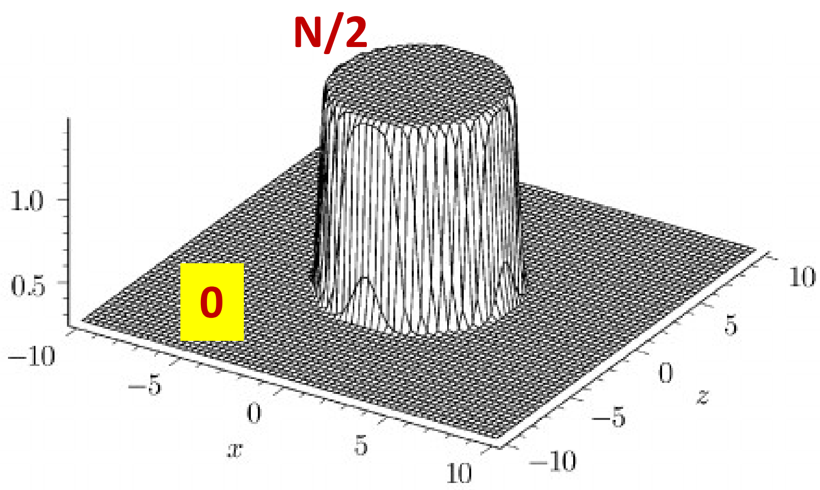
MLP as a continuous-valued regression

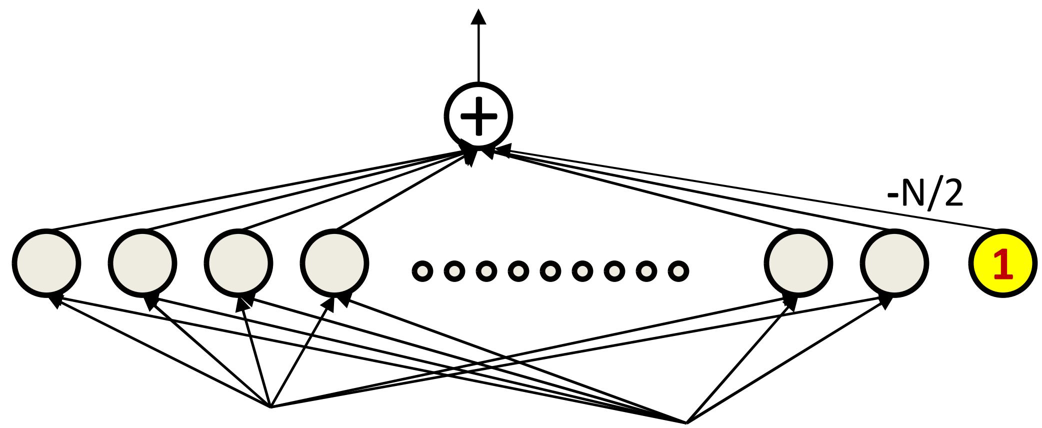




- A simple 3-unit MLP can generate a "square pulse" over an input
- **An MLP with many units can model an arbitrary function over an input**
  - To arbitrary precision
    - Simply make the individual pulses narrower
- ***A one-hidden-layer MLP can model an arbitrary function of a single input***

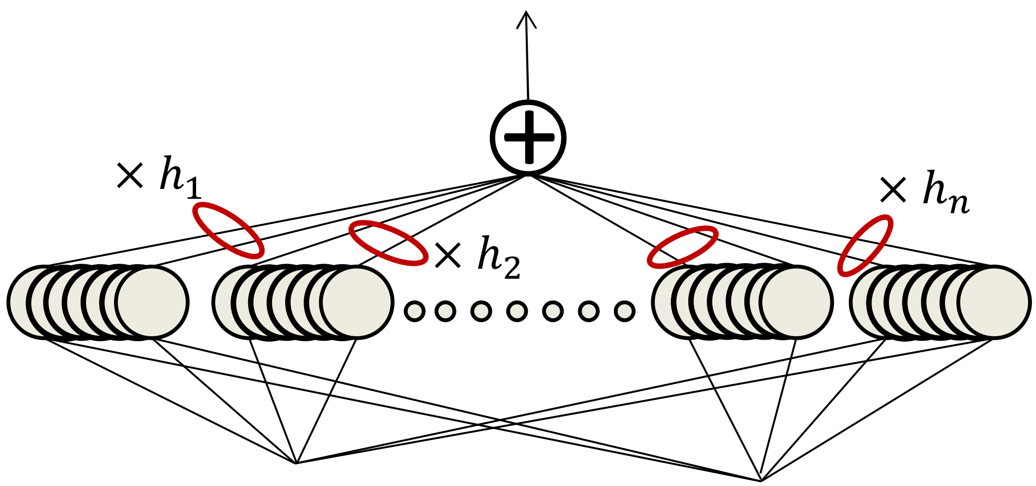
For higher dimensions

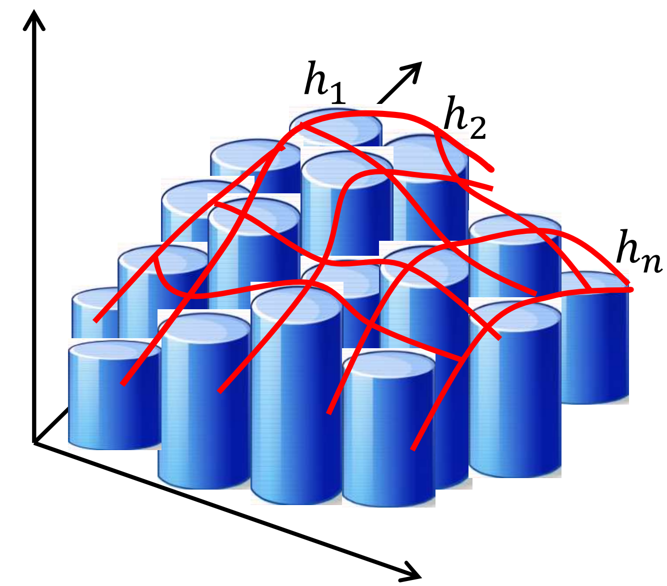




- An MLP can compose a cylinder
  - $N/2$ in the circle, 0 outside

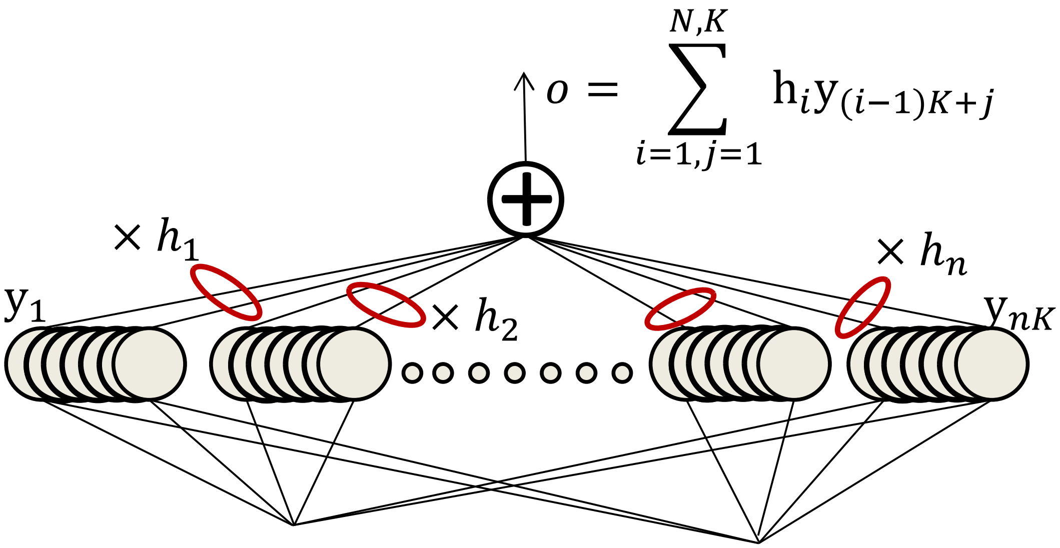
MLP as a continuous-valued function

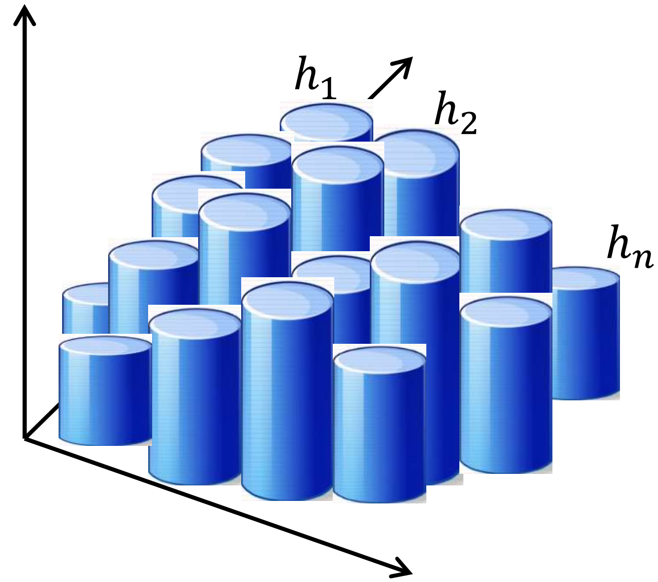




- MLPs can actually compose arbitrary functions in any number of dimensions!
  - Even with only one hidden layer
    - As sums of scaled and shifted cylinders
  - To arbitrary precision
    - By making the cylinders thinner
  - **The MLP is a universal approximator!**

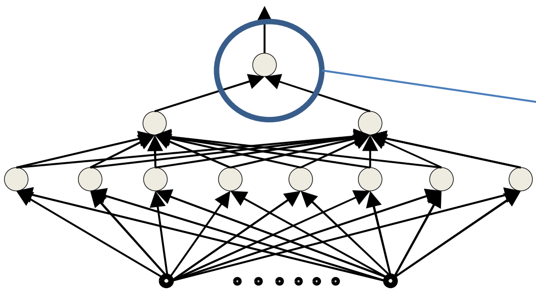
Caution: MLPs with additive output units are universal approximators

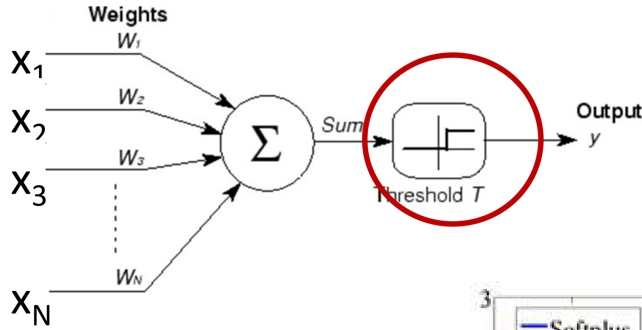




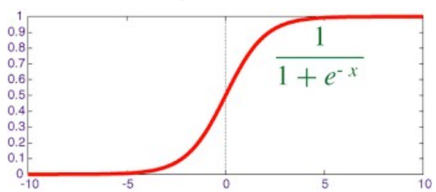
- MLPs can actually compose arbitrary functions
- But explanation so far only holds if the output unit only performs summation
  - i.e. does not have an additional "activation"

"Proper" networks: Outputs with activations

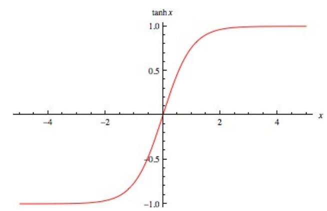


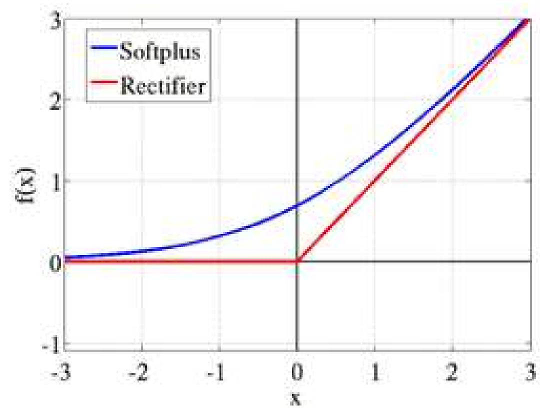


sigmoid



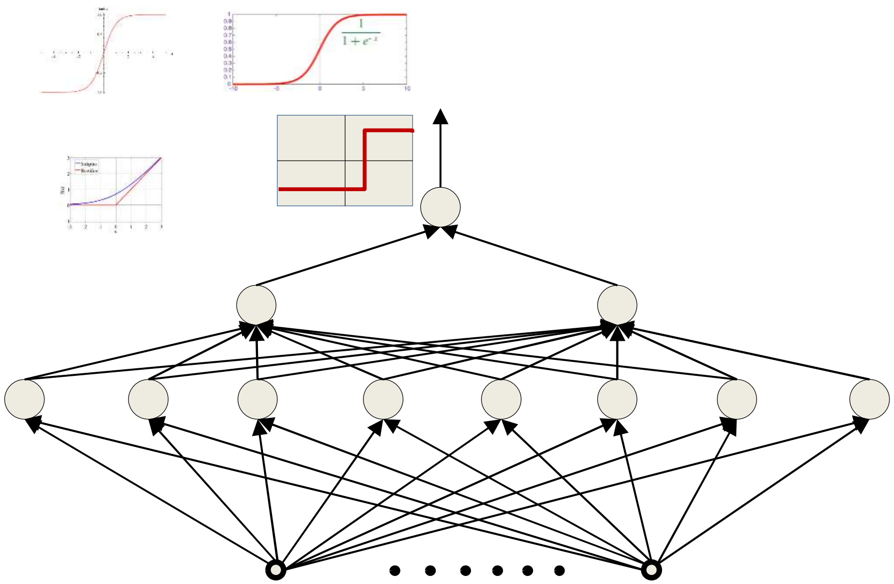
tanh





- Output neuron may have actual "activation"
  - Threshold, sigmoid, tanh, softplus, rectifier, etc.
- What is the property of such networks?

The network as a function



$$f: \{0,1\}^N \rightarrow \{0,1\} \quad Boolean$$

$$f: R^N \rightarrow \{0,1\} \quad Threshold$$

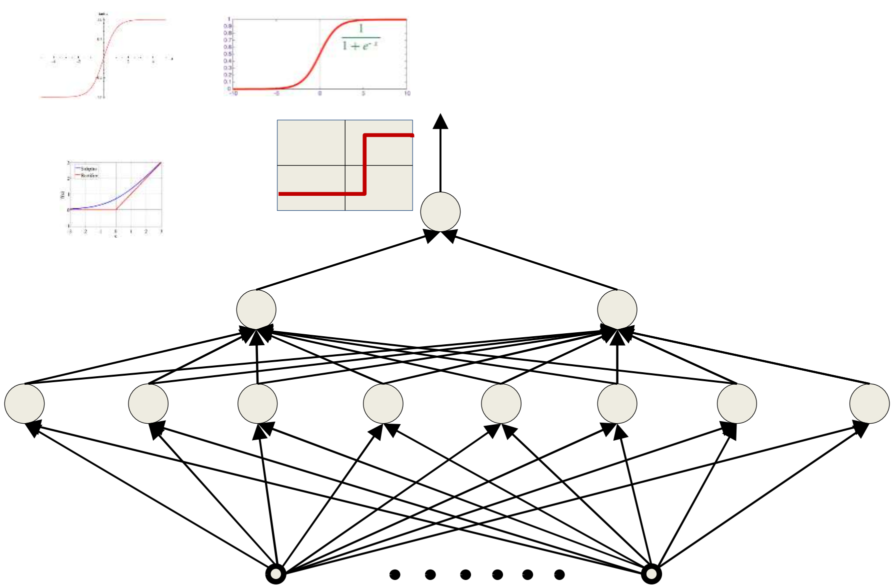
$$f: R^N \rightarrow (0,1) \quad Sigmoid$$

$$f: R^N \rightarrow (-1,1) \quad Tanh$$

$$f: R^N \rightarrow [0, \infty) \; Rectifier, softrectifier$$

- Output unit with *activation function*
  - Threshold or Sigmoid, or any other
- The network is actually a universal map from the entire domain of input values to the entire range of the output activation
  - All values the activation function of the output neuron

The network as a function



$f: \{0,1\}^N \rightarrow \{0,1\}$ *Boolean*

$f: R^N \rightarrow \{0,1\}$ *Threshold*

$f: R^N \rightarrow (0,1)$ *Sigmoid*

$f: R^N \rightarrow (-1,1)$ *Tanh*

$f: R^N \rightarrow [0, \infty)$ *Rectifier, softrectifier*

The MLP is a *Universal Approximator* for the entire *class* of functions (maps) it represents!

- Threshold or Sigmoid, or any other

- The network is actually a universal map from the entire domain of input values to the entire range of the output activation
  - All values the activation function of the output neuron

Today

- Multi-layer Perceptrons as universal Boolean functions
  - The need for depth
- MLPs as universal classifiers
  - The need for depth
- MLPs as universal approximators
- A discussion of optimal depth and width
- Brief segue: RBF networks

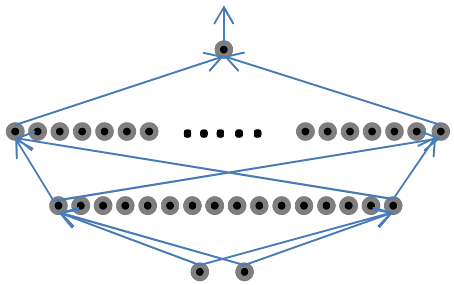
The issue of depth

- Previous discussion showed that a *single-hidden-layer* MLP is a universal function approximator
  - Can approximate any function to arbitrary precision
  - But may require infinite neurons in the layer

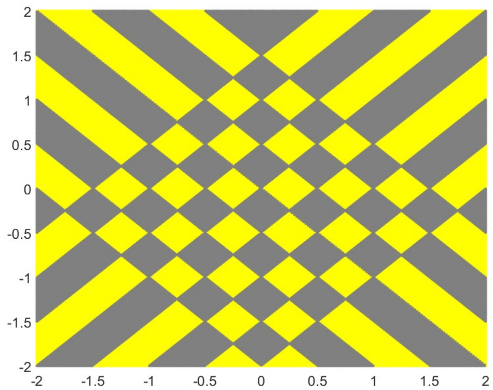
- More generally, deeper networks will require far fewer neurons for the same approximation error
  - True for Boolean functions, classifiers, and real-valued functions

- But there are limitations…

Sufficiency of architecture

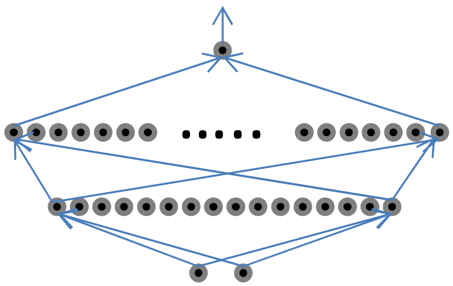


A network with 16 or more neurons in the first layer is capable of representing the figure to the right perfectly

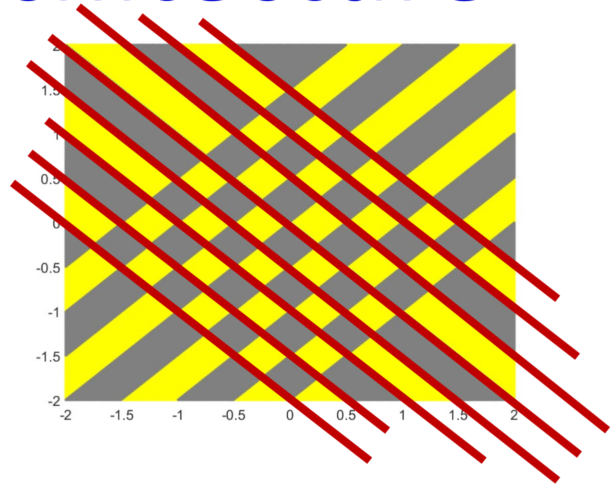


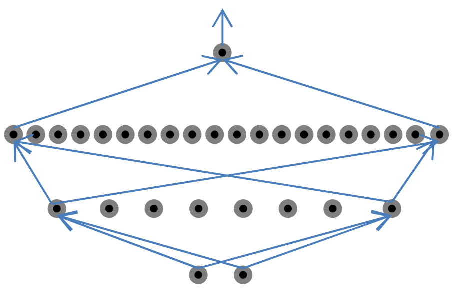
- A neural network *can* represent any function provided it has sufficient *capacity*
  – I.e. sufficiently broad and deep to represent the function
- Not all architectures can represent any function

Sufficiency of architecture



A network with 16 or more neurons in the first layer is capable of representing the figure to the right perfectly





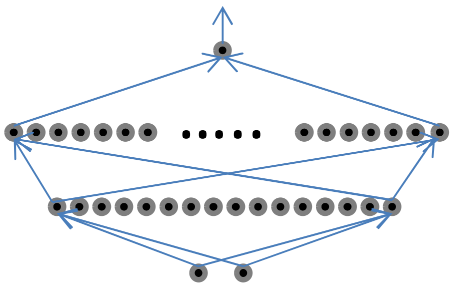
A network with less than 16 neurons in the first layer cannot represent this pattern exactly

- *With caveats..*

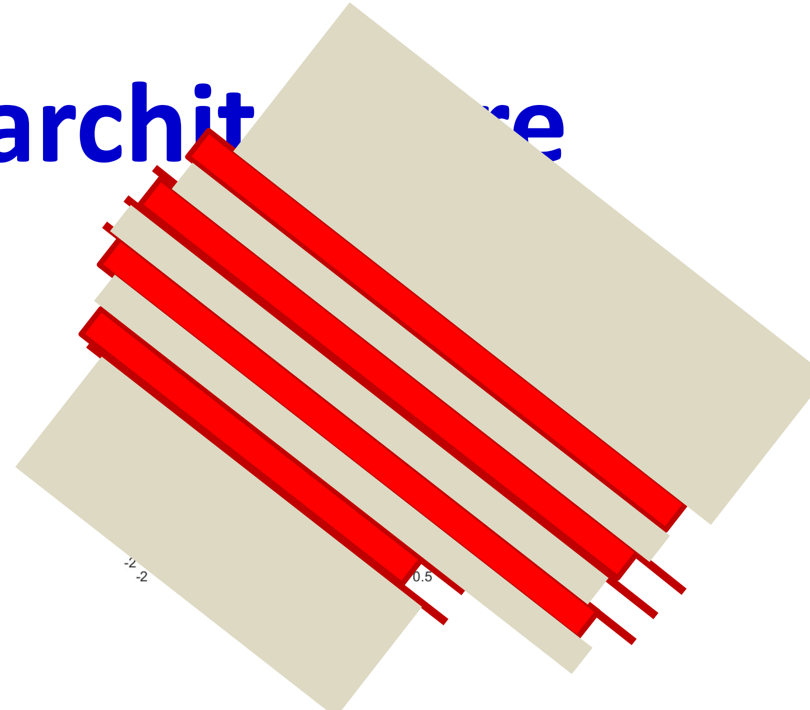
Why?

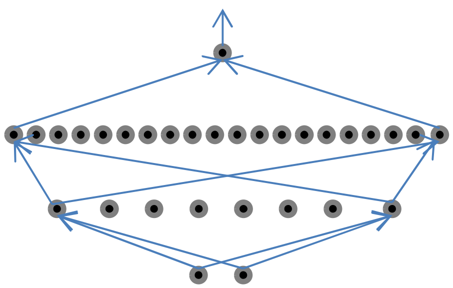
- A neural network *can* represent any function provided it has sufficient *capacity*
  - I.e. sufficiently broad and deep to represent the function
- Not all architectures can represent any function

Sufficiency of architecture



A network with 16 or more neurons in the first layer is capable of representing the figure to the right perfectly



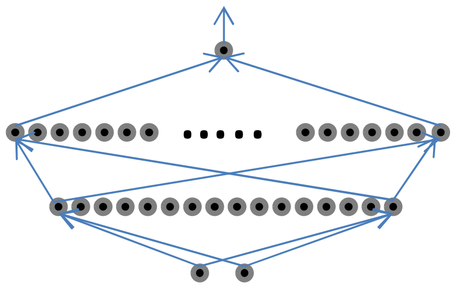


A network with less than 16 neurons in the first layer cannot represent this pattern exactly

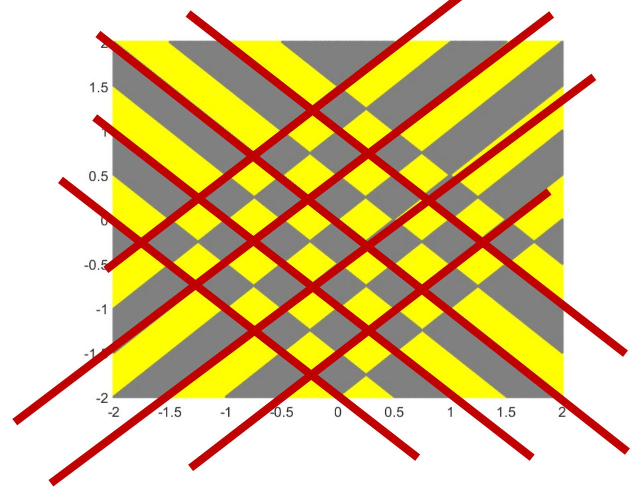
❖ *With caveats..* → **We will revisit this idea shortly**

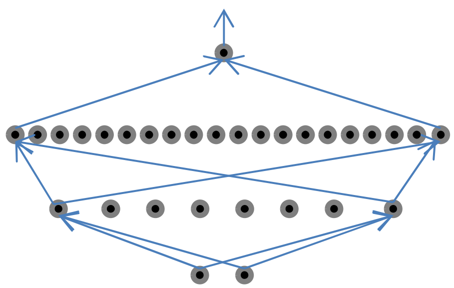
- A neural network *can* represent any function provided it has sufficient *capacity*
  - I.e. sufficiently broad and deep to represent the function
- Not all architectures can represent any function

Sufficiency of architecture



A network with 16 or more neurons in the first layer is capable of representing the figure to the right perfectly





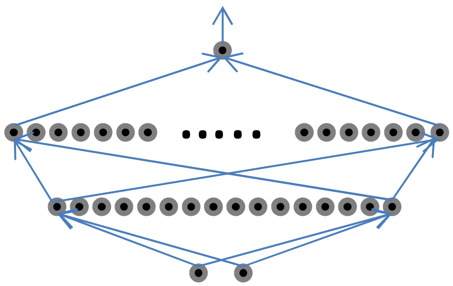
A network with less than 16 neurons in the first layer cannot represent this pattern exactly

❖ *With caveats..*

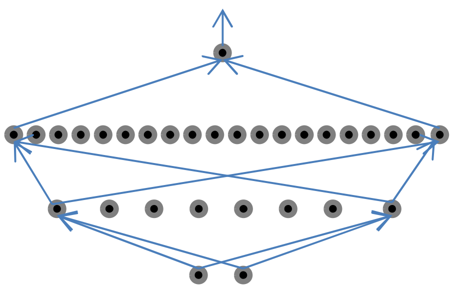
Why?

- A neural network *can* represent any function provided it has sufficient *capacity*
  - I.e. sufficiently broad and deep to represent the function
- Not all architectures can represent any function

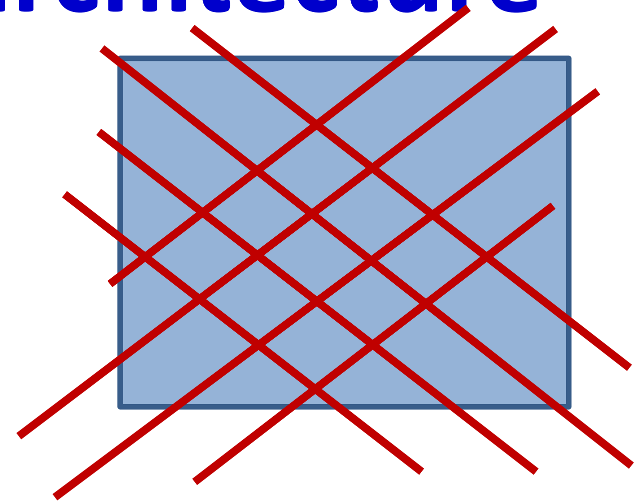
Sufficiency of architecture



A network with 16 or more neurons in the first layer is capable of representing the figure to the right perfectly



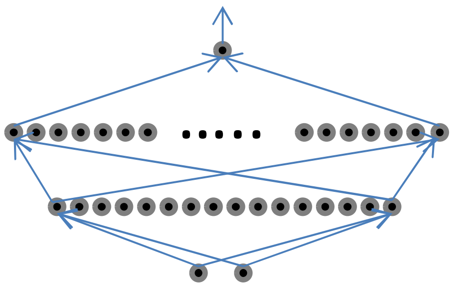
A network with less than 16 neurons in the first layer cannot represent this pattern exactly
❖ With caveats..



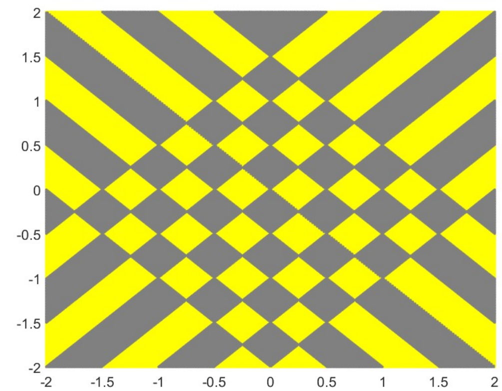
Why?

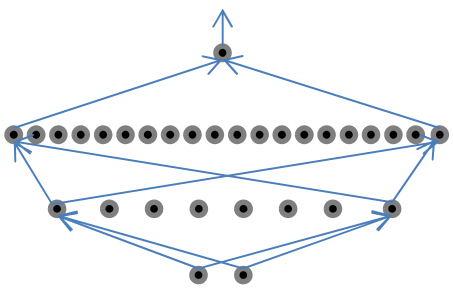
- A neural network *can* represent any function provided it has sufficient *capacity*
  – I.e. sufficiently broad and deep to represent the function
- Not all architectures can represent any function

Sufficiency of architecture

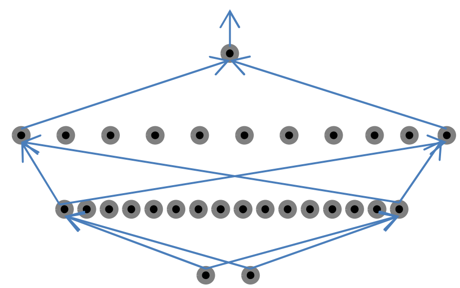


A network with 16 or more neurons in the first layer is capable of representing the figure to the right perfectly





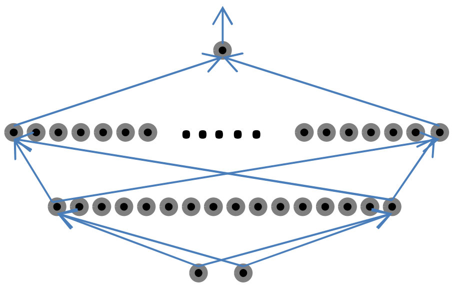
A network with less than 16 neurons in the first layer cannot represent this pattern exactly
- *With caveats..*



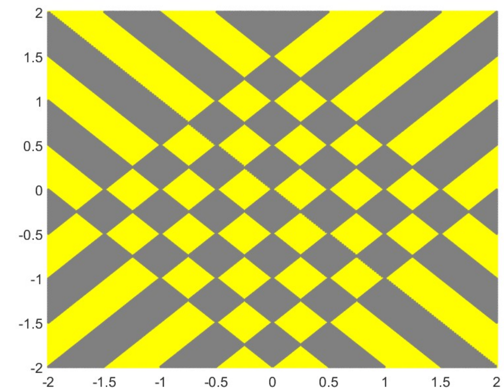
A 2-layer network with 16 neurons in the first layer cannot represent the pattern with less than 40 neurons in the second layer

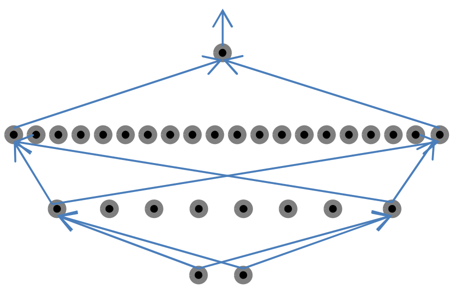
- A neural network *can* represent any function provided it has sufficient *capacity*
  - I.e. sufficiently broad and deep to represent the function
- Not all architectures can represent any function

Sufficiency of architecture

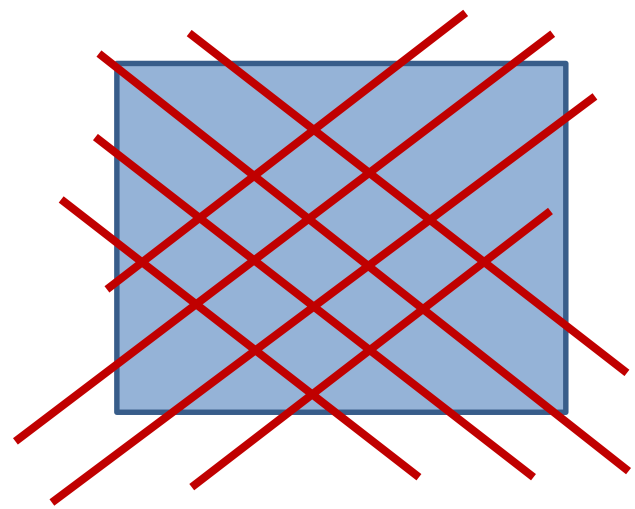


A network with 16 or more neurons in the first layer is capable of representing the figure to the right perfectly



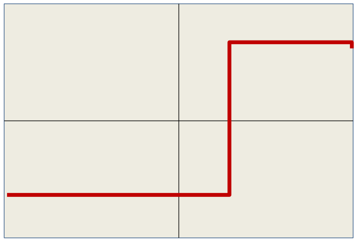


A network with less than 16 neurons in the first layer cannot represent this pattern exactly
❖ With caveats..



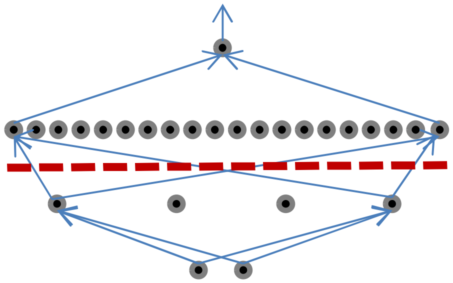
Why?

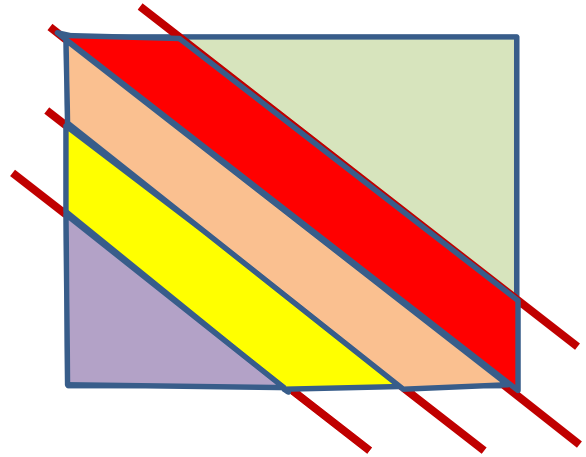
Sufficiency of architecture



This effect is because we use the threshold activation

It *gates* information in the input from later layers

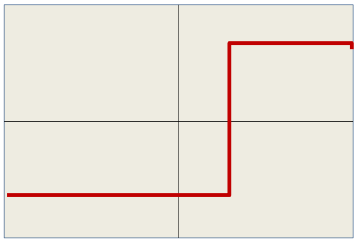




The pattern of outputs within any colored region is identical

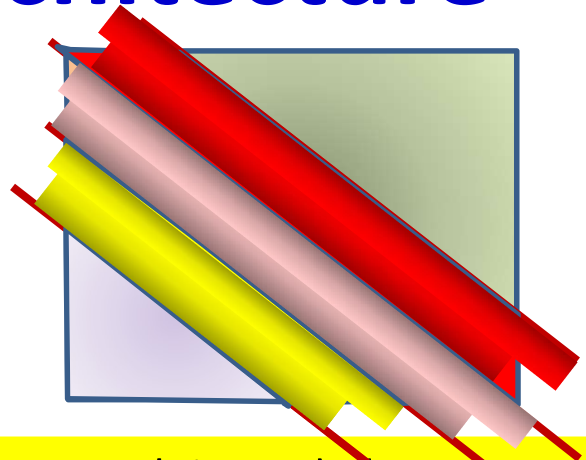
Subsequent layers do not obtain enough information to partition them

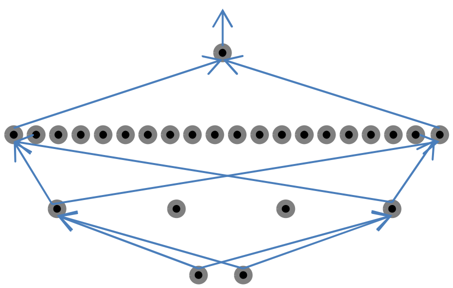
Sufficiency of architecture



This effect is because we use the threshold activation

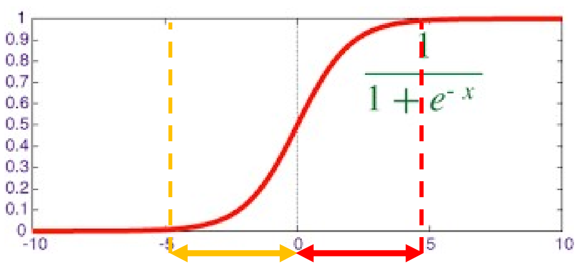
It *gates* information in the input from later layers

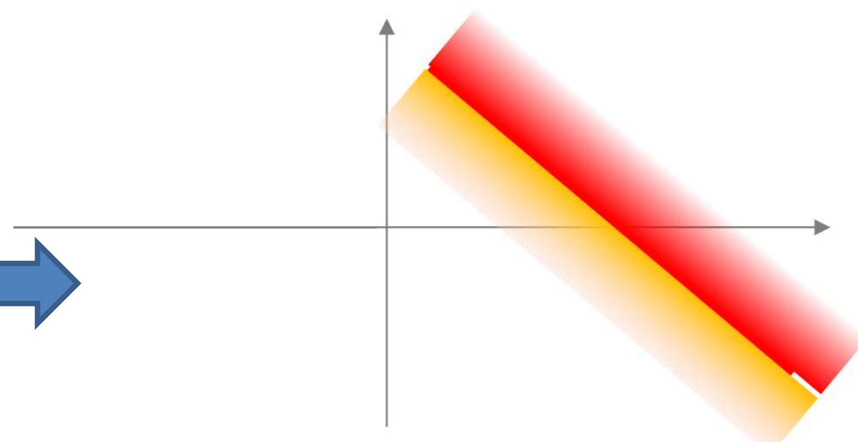




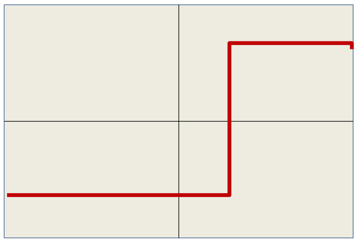
Continuous activation functions result in graded output at the layer

The gradation provides information to subsequent layers, to capture information “missed” by the lower layer (i.e. it “passes” information to subsequent layers).



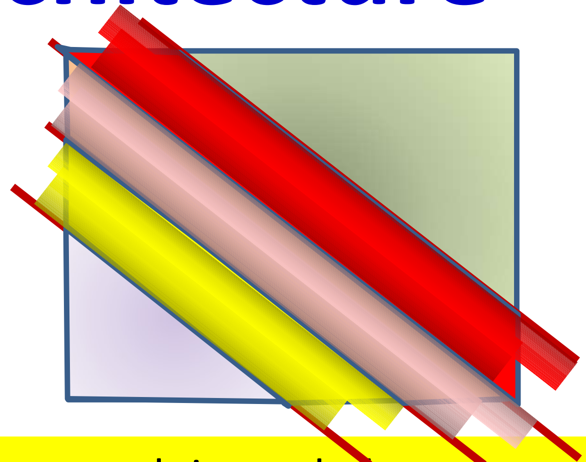


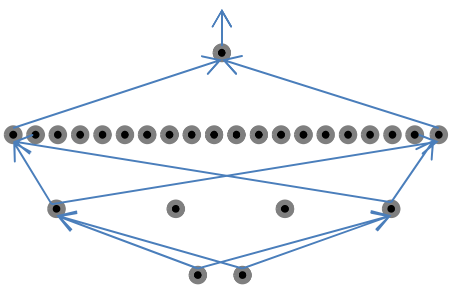
Sufficiency of architecture



This effect is because we use the threshold activation

It *gates* information in the input from later layers

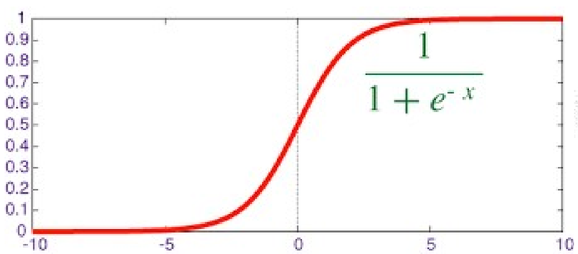


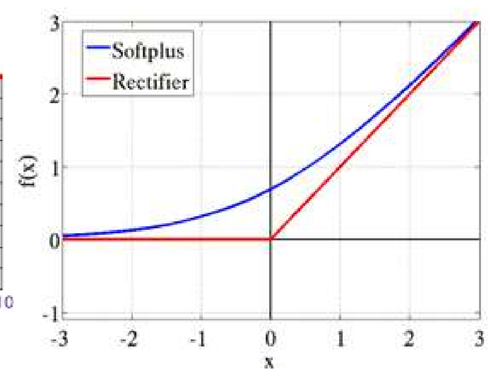


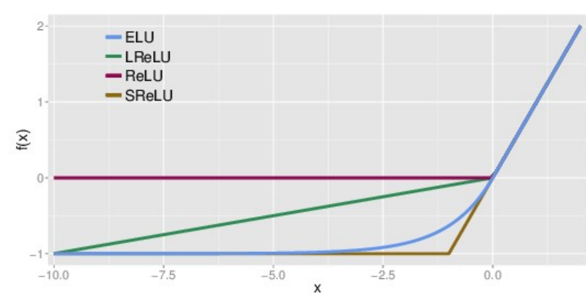
Continuous activation functions result in graded output at the layer

The gradation provides information to subsequent layers, to capture information "missed" by the lower layer (i.e. it "passes" information to subsequent layers).

Activations with more gradation (e.g. RELU) pass more information





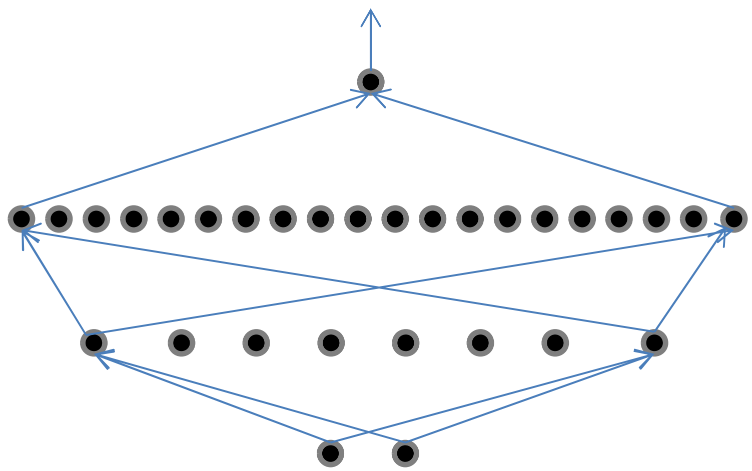


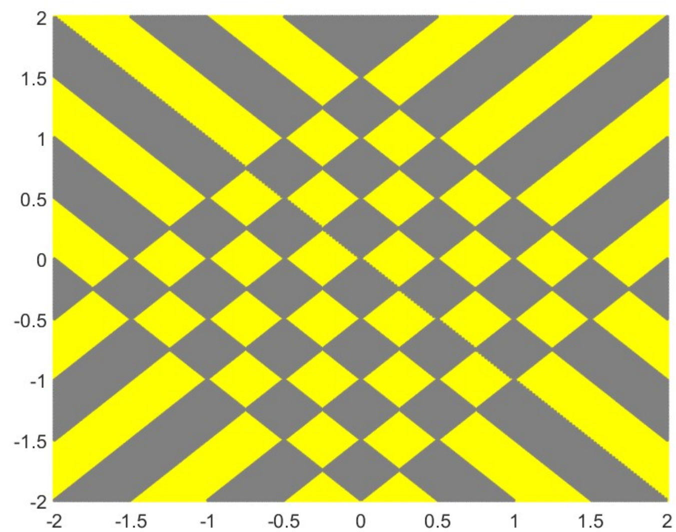
Width vs. Activations vs. Depth

- Narrow layers can still pass information to subsequent layers if the activation function is sufficiently graded

- But will require greater depth, to permit later layers to capture patterns

Sufficiency of architecture





- The *capacity* of a network has various definitions
  - *Information* or *Storage* capacity: how many patterns can it remember
  - VC dimension
    - bounded by the square of the number of weights in the network
  - From our perspective: largest number of disconnected convex regions it can represent

- A network with insufficient capacity *cannot* exactly model a function that requires a greater minimal number of convex hulls than the capacity of the network
  - But can approximate it with error

The "capacity" of a network

- VC dimension
- A separate lecture
  - Koiran and Sontag (1998): For "linear" or threshold units, VC dimension is proportional to the number of weights
    - For units with piecewise linear activation it is proportional to the square of the number of weights
  - Batlett, Harvey, Liaw, Mehrabian "Nearly-tight VC-dimension bounds for piecewise linear neural networks" (2017):
    - For any W, L s.t. $W > CL > C^2$, there exisits a RELU network with $\leq L$ layers, $\leq W$ weights with VC dimension $\geq \frac{WL}{C} \log_2(\frac{W}{L})$
  - Friedland, Krell, "A Capacity Scaling Law for Artificial Neural Networks" (2017):
    - VC dimension of a linear/threshold net is $\mathcal{O}(MK)$, M is the overall number of hidden neurons, K is the weights per neuron

Lessons today

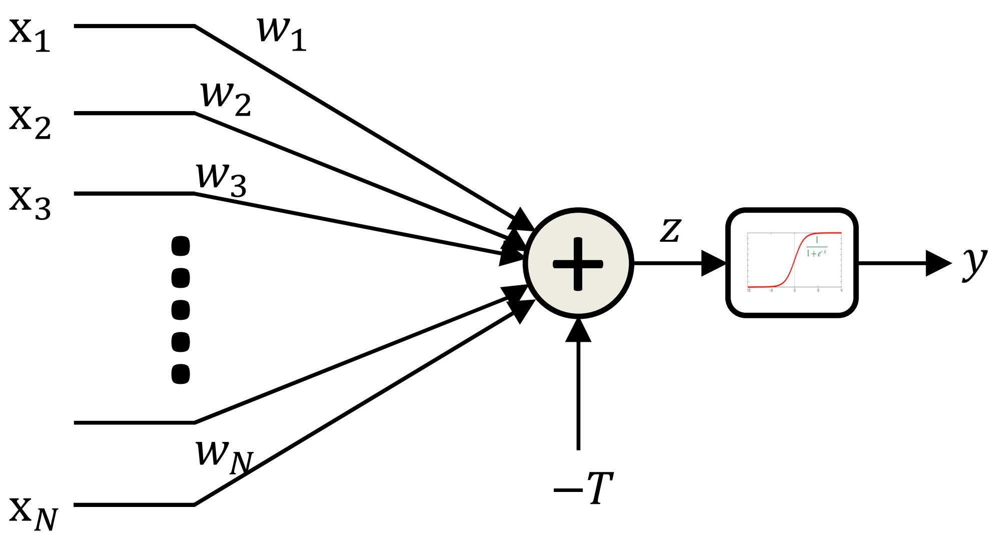
- MLPs are universal Boolean function
- MLPs are universal classifiers
- MLPs are universal function approximators

- A *single-layer* MLP can approximate anything to arbitrary precision
  - But could be exponentially or even infinitely wide in its inputs size
- Deeper MLPs can achieve the same precision with far fewer neurons
  - Deeper networks are more expressive
  - More graded activation functions result in more expressive networks

Today

- Multi-layer Perceptrons as universal Boolean functions
  - The need for depth
- MLPs as universal classifiers
  - The need for depth
- MLPs as universal approximators
- A discussion of optimal depth and width
- Brief segue: RBF networks

Perceptrons so far

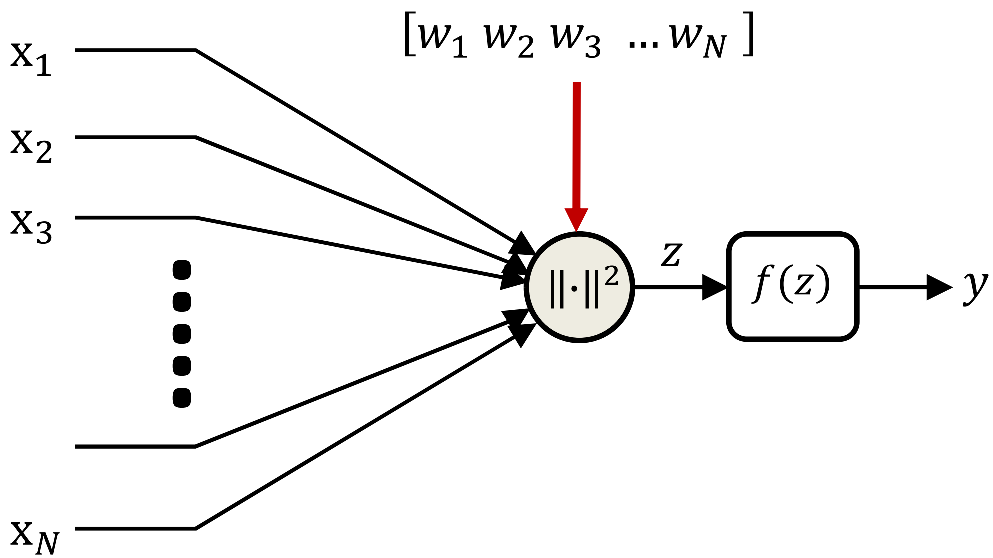


$$z = \sum_i w_i x_i - T$$

$$y = f(z)$$

- The output of the neuron is a function of a linear combination of the inputs and a bias

An alternate type of neural unit: Radial Basis Functions



$$z = \|\mathbf{x} - \mathbf{w}\|^2$$

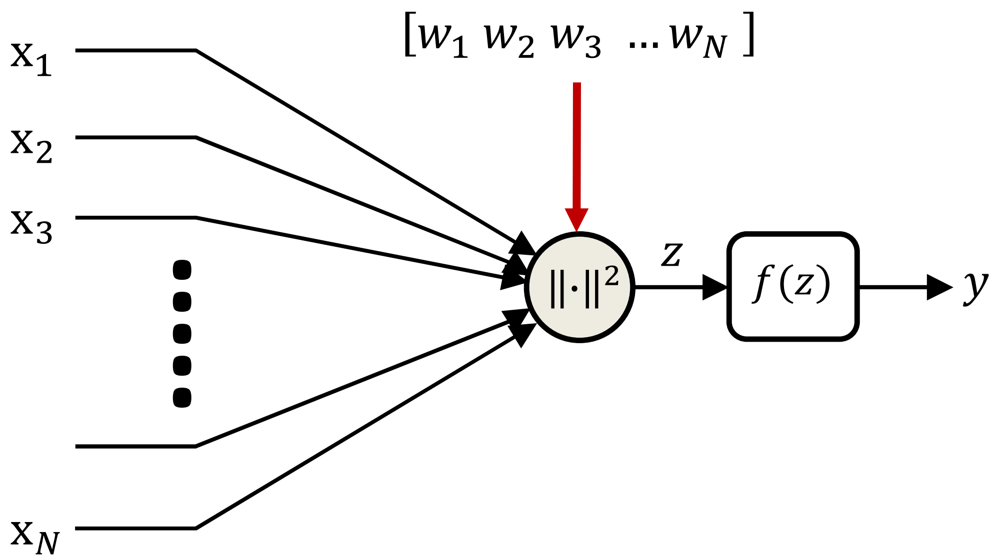
$$y = f(z)$$

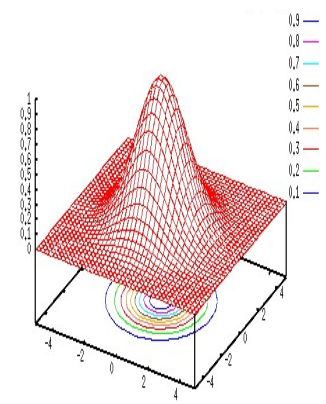
Typical activation

$$f(z) = \exp(-\beta z)$$

- The output is a function of the distance of the input from a "center"
  - The "center" $\mathbf{w}$ is the parameter specifying the unit
  - The most common activation is the exponent
    - β is a "bandwidth" parameter
  - But other similar activations may also be used
    - Key aspect is radial symmetry, instead of linear symmetry

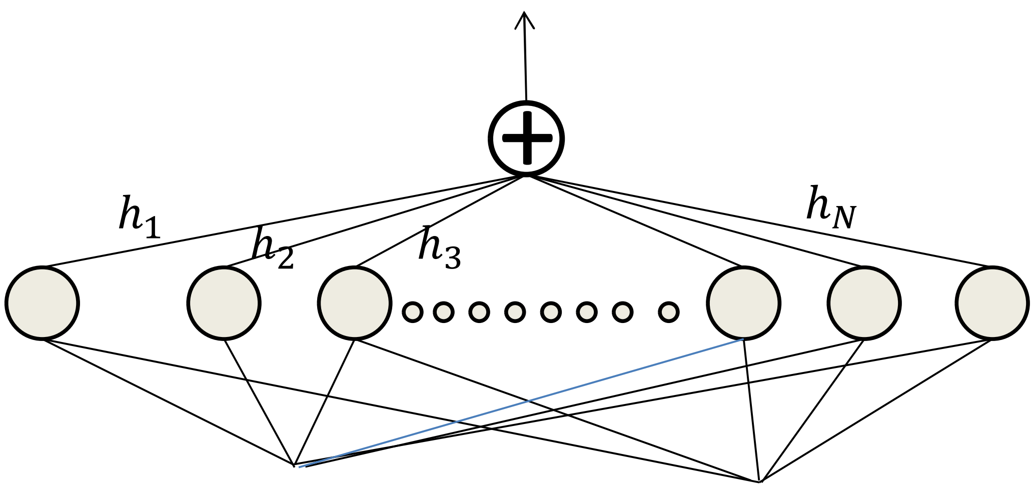
An alternate type of neural unit:
Radial Basis Functions

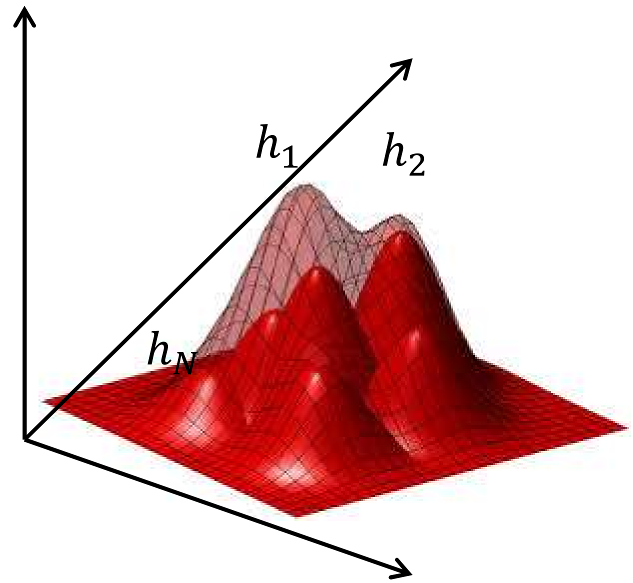




- Radial basis functions can compose cylinder-like outputs with just a single unit with appropriate choice of bandwidth (or activation function)
  - As opposed to $N \rightarrow \infty$ units for the linear perceptron

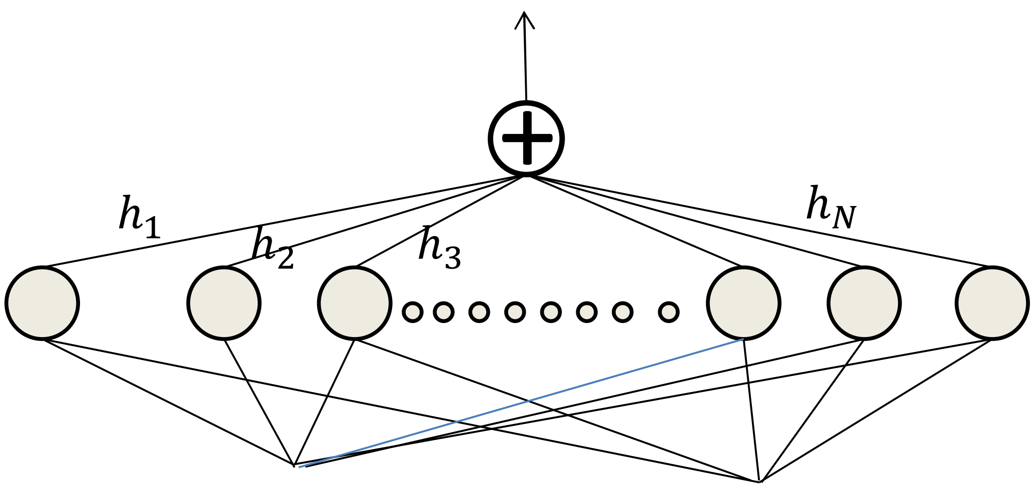
RBF networks as universal approximators

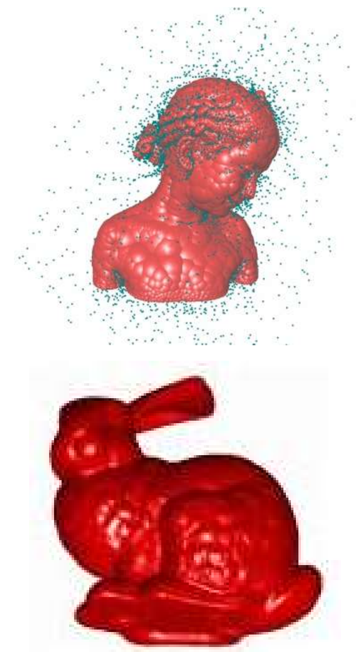




- RBF networks are more effective approximators of continuous-valued functions
  - A one-hidden-layer net only requires *one* unit per "cylinder"

RBF networks as universal approximators





- RBF networks are more effective approximators of continuous-valued functions
  - A one-hidden-layer net only requires *one* unit per "cylinder"

RBF networks

- More effective than conventional linear perceptron networks in some problems

- We will revisit this topic, time permitting

Lessons today

- MLPs are universal Boolean function
- MLPs are universal classifiers
- MLPs are universal function approximators

- A *single-layer* MLP can approximate anything to arbitrary precision
  - But could be exponentially or even infinitely wide in its inputs size
- Deeper MLPs can achieve the same precision with far fewer neurons
  - Deeper networks are more expressive

- RBFs are good, now lets get back to linear perceptrons… ☺

Next up

- We *know* MLPs can emulate any function
- But how do we *make* them emulate a specific desired function
  - E.g. a function that takes an image as input and outputs the labels of all objects in it
  - E.g. a function that takes speech input and outputs the labels of all phonemes in it
  - Etc…
- *Training an MLP*