

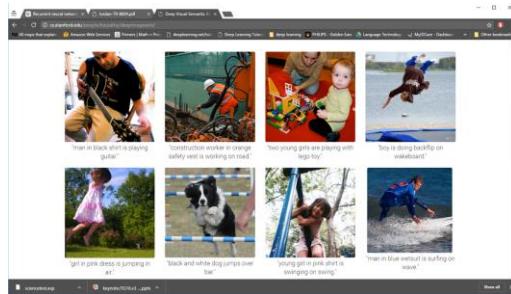
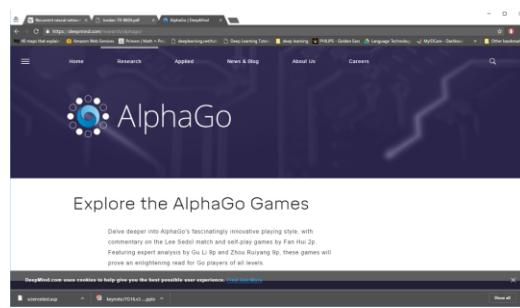
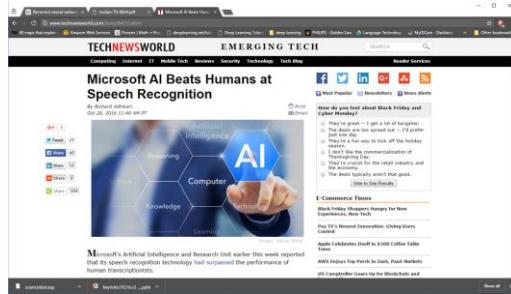
Neural Networks: What can a network represent

Deep Learning, Spring 2023

Topics for the day

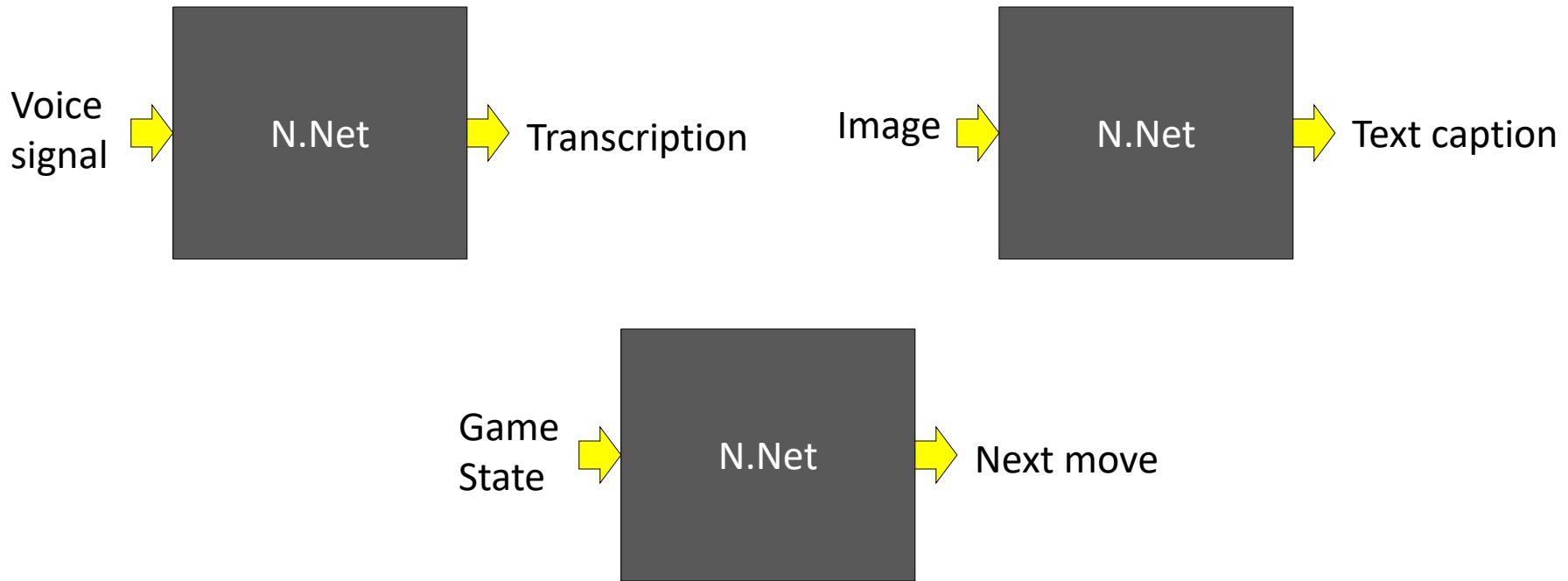
- What can neural networks represent
- And what are the restrictions
 - In terms of “depth”, “width” and “activations”

Recap : Neural networks have taken over AI



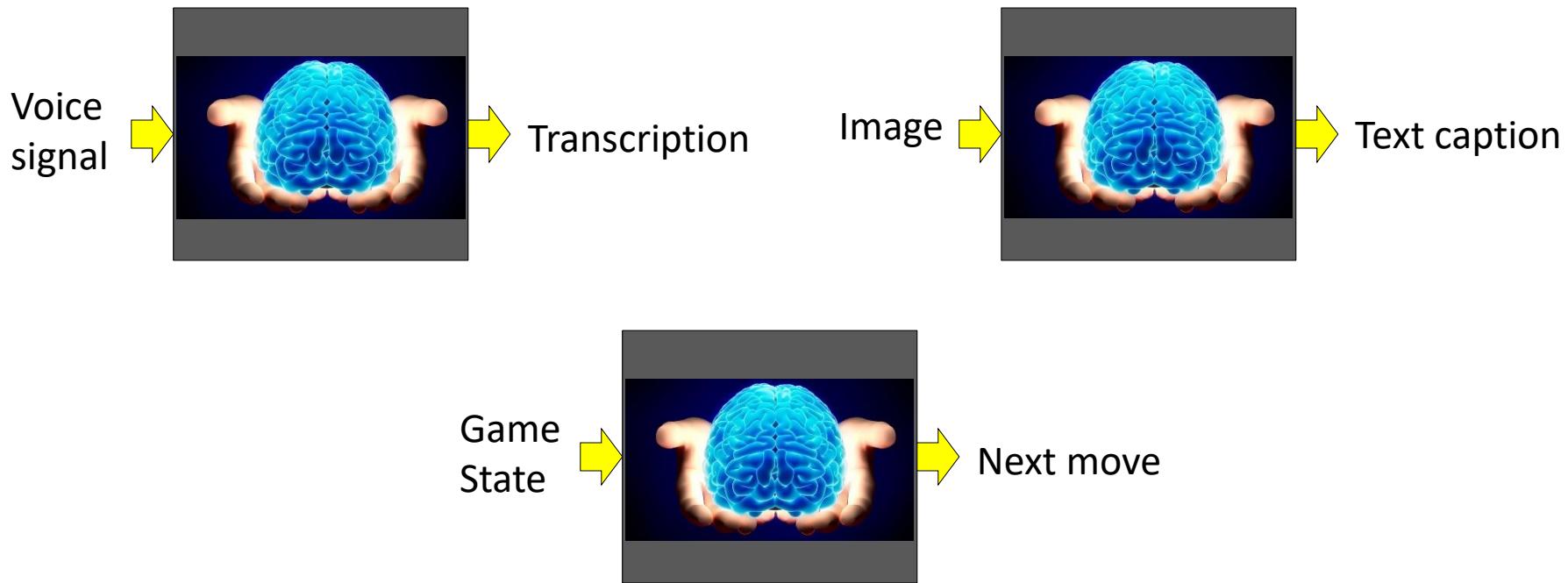
- Tasks that are made possible by NNs, aka deep learning
 - Tasks that were once assumed to be purely in the human domain of expertise

So what are neural networks??



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

The human perspective



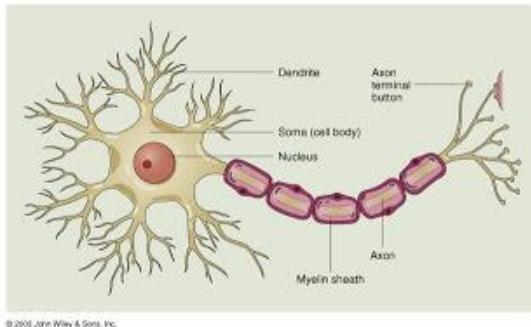
- In a human, those functions are computed by the brain...

Recap : NNets and the brain



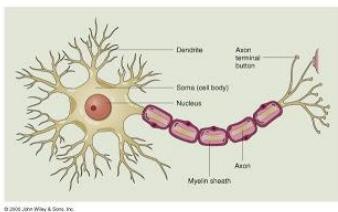
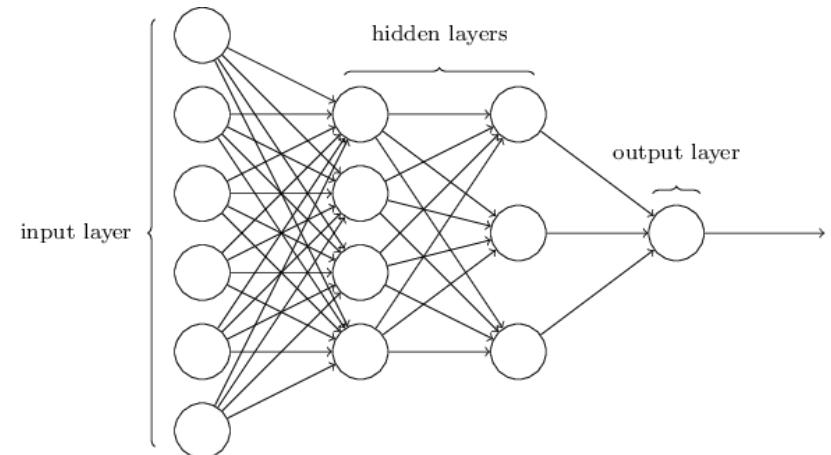
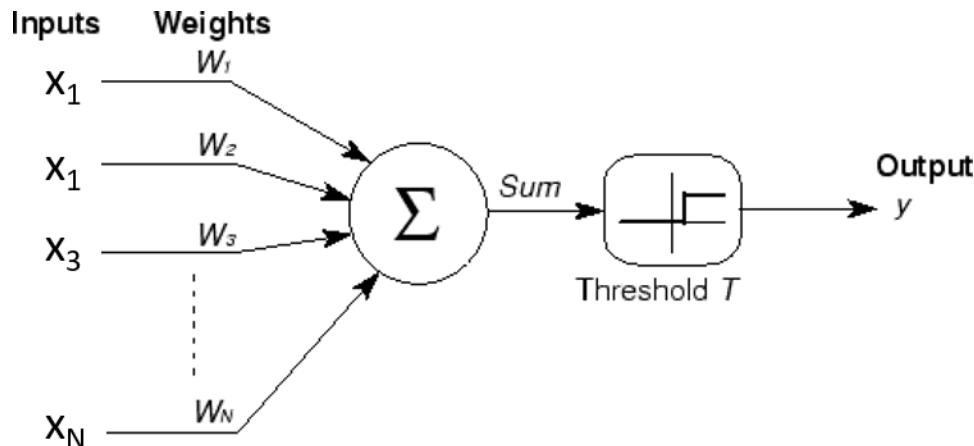
- In their basic form, NNets mimic the networked structure in the brain

Recap : The brain



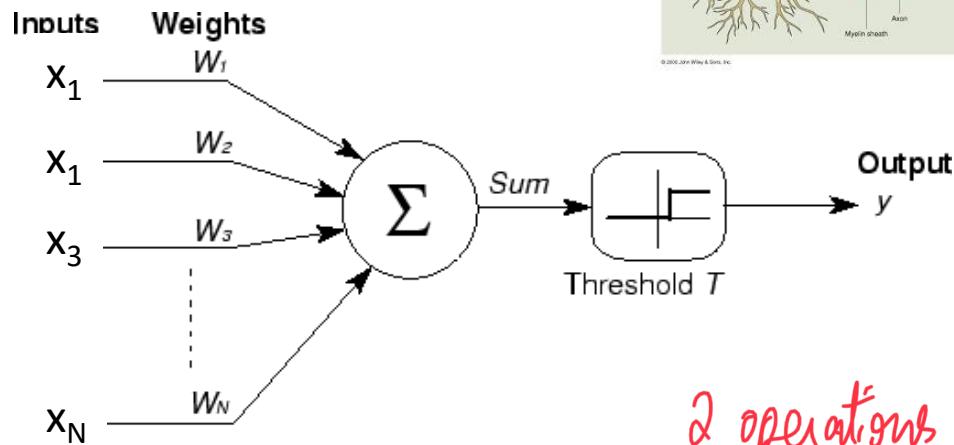
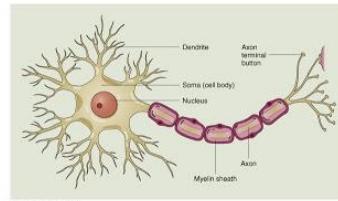
- The Brain is composed of networks of neurons

Recap : Nnets and the brain



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

Recap: the perceptron

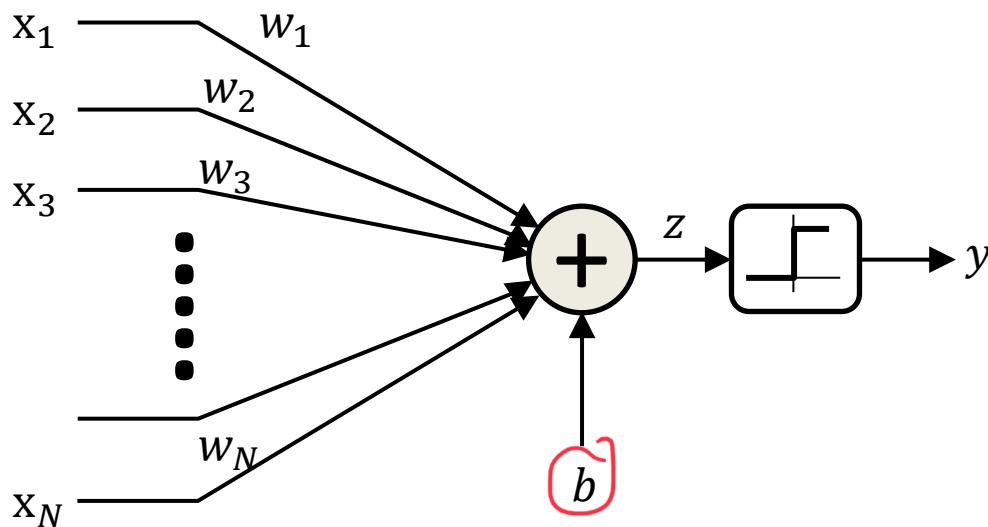


$$y = \begin{cases} 1 & \text{if } \sum_i w_i x_i \geq T \\ 0 & \text{else} \end{cases}$$

2 operations : 1. Compute weighted sum
2. Compare w/ threshold

- 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



Linear vs Affine?

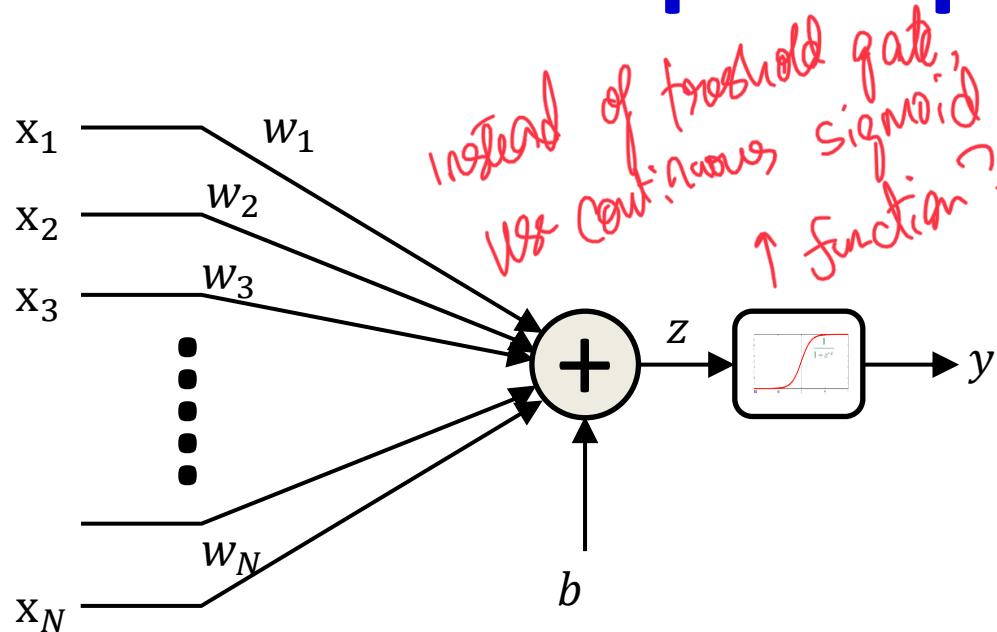
$$z = \sum_i w_i x_i + b$$

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

- A threshold unit $\xrightarrow{\text{linear function with a bias}}$
 - “Fires” if the affine function of inputs is positive
 - The bias is the negative of the threshold T in the previous slide

(Qn) Is it more preferable to have smooth (cont act func?)

The “soft” perceptron (logistic)

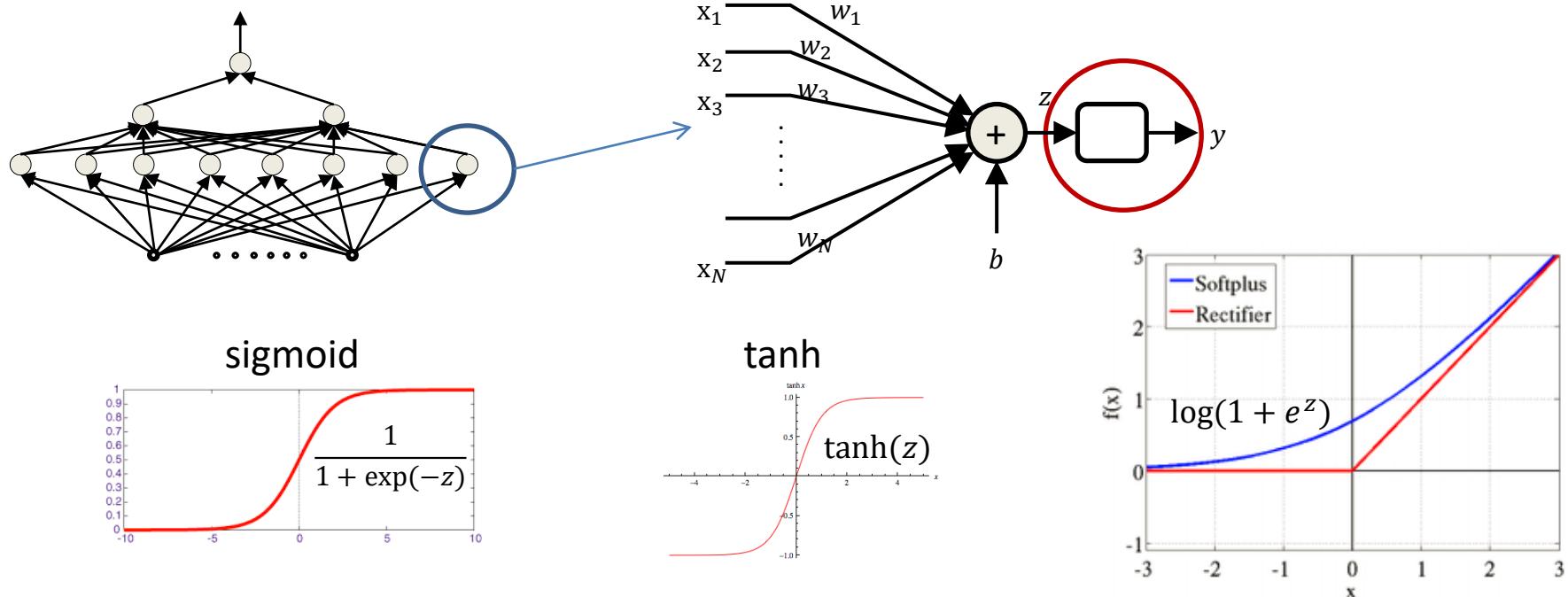


$$z = \sum_i w_i x_i + b$$

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

- 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 bias)

Other “activations”



- 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

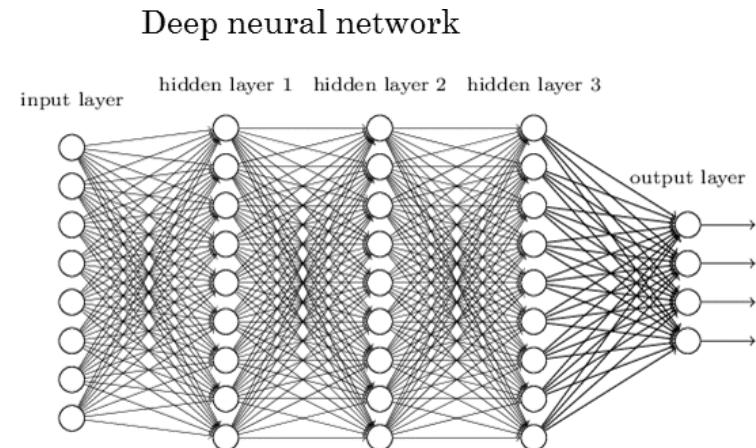
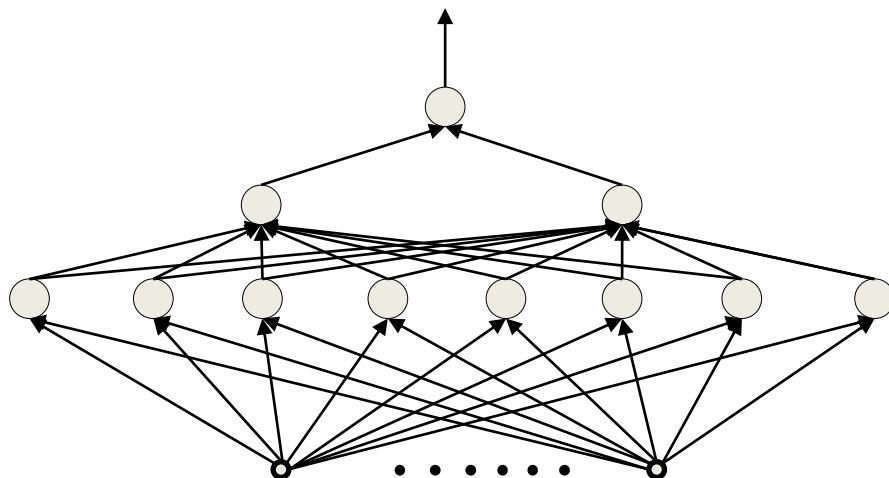
Poll 1

- Mark all true statements
 - $-3x + 7y$ is a linear combination of x and y
 - $-3x + 7y + 4$ is a linear combination of x and y
 - $-3x + 7y$ is an affine function of x and y
 - $-3x + 7y + 4$ is an affine function of x and y

Poll 1

- Mark all true statements
 - $-3x + 7y$ is a linear combination of x and y
 - $-3x + 7y + 4$ is a linear combination of x and y
 - $-3x + 7y$ is an affine function of x and y
 - $-3x + 7y + 4$ is an affine function of x and y

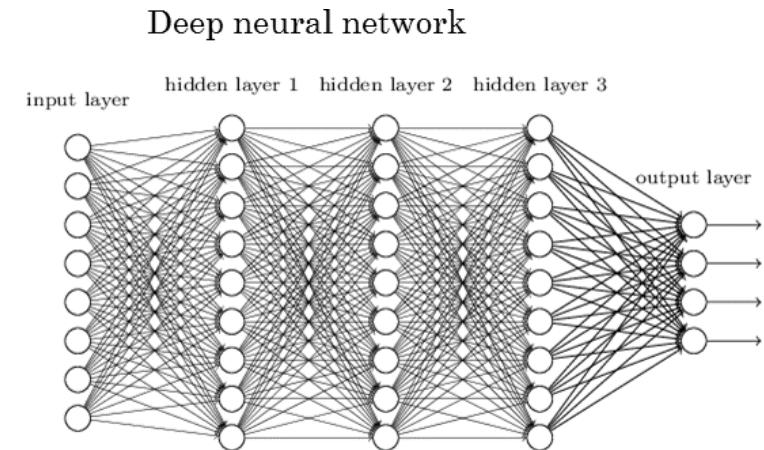
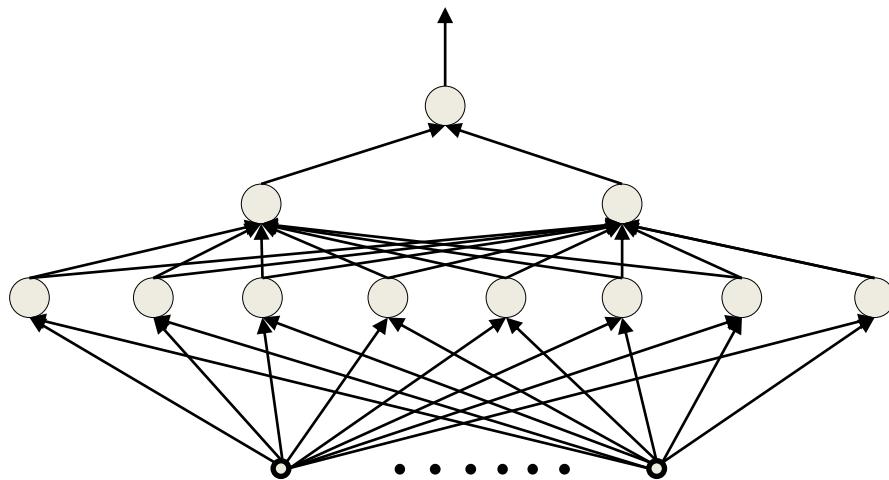
The *multi-layer* perceptron



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



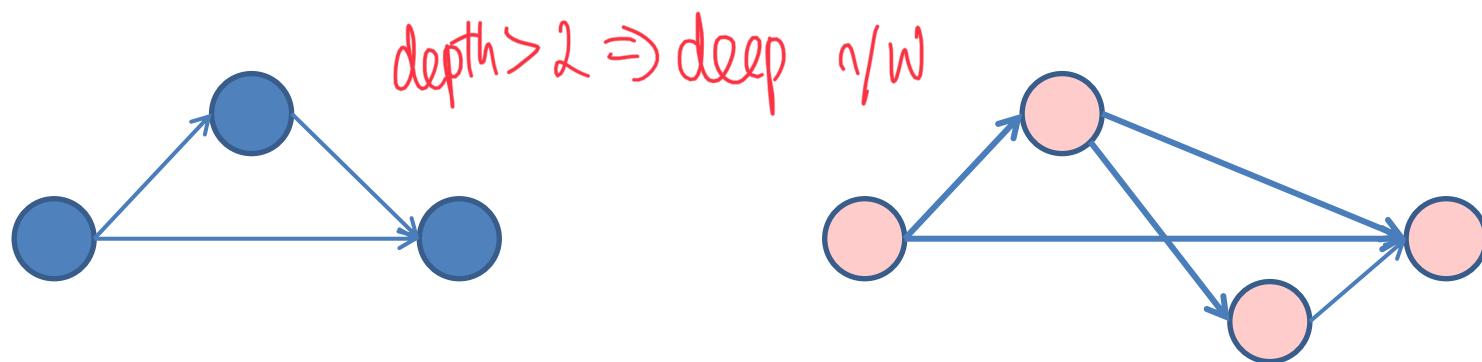
Defining “depth”



- What is a “deep” network

Deep Structures

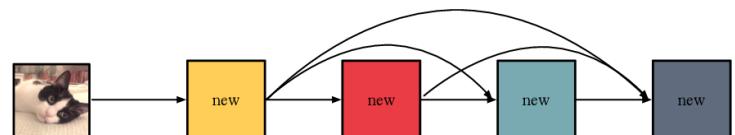
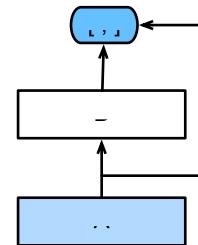
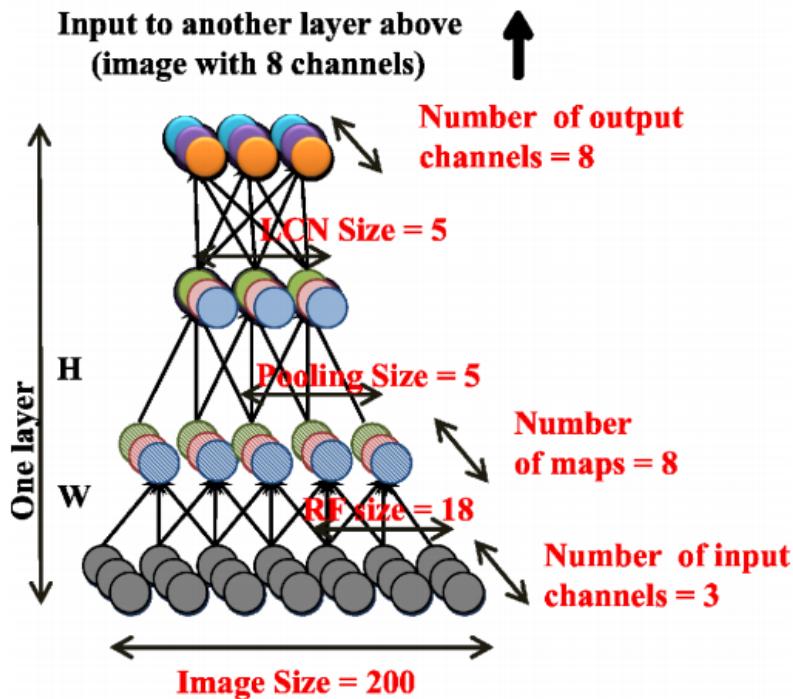
- In any directed graph with input source nodes and output sink nodes, “depth” is the length of the longest path from a source to a sink
 - A “source” node in a directed graph is a node that has only outgoing edges
 - A “sink” node is a node that has only incoming edges



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

Deep Structures

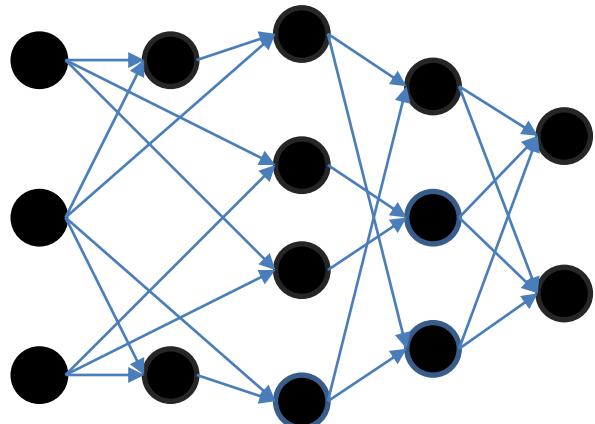
- *Deep structure*
 - *The input is the “source”,*
 - *The output nodes are “sinks”*



- “Deep” → Depth of output neurons is greater than 2

What is a layer?

- A “layer” is the set of neurons that are all at the same depth with respect to the input (sink)
 - “Depth” of a layer – the depth of the neurons in the layer w.r.t. input

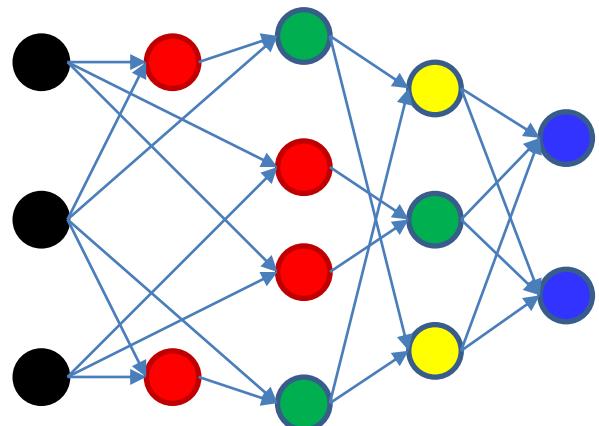


Input:
Layer 1:
Layer 2:
Layer 3:
Layer 4:

- “Deep” → At least 3 layers
 - Output layer depth is at least 3

What is a layer?

- A “layer” is the set of neurons that are all at the same depth with respect to the input (sink)
 - “Depth” of a layer – the depth of the neurons in the layer w.r.t. input

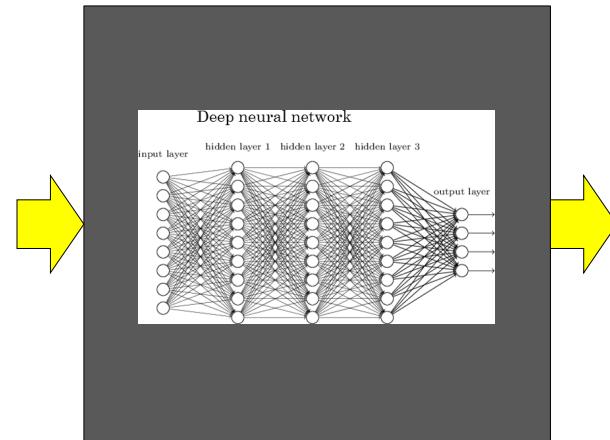
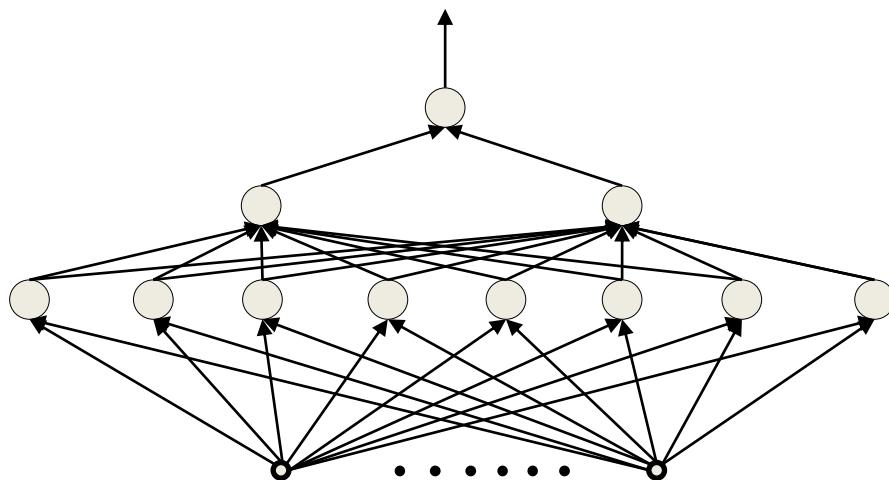


Input: Black
Layer 1: Red
Layer 2: Green
Layer 3: Yellow
Layer 4: Blue

- “Deep” → At least 3 layers
 - Output layer depth is at least 3

Multiple o/p neurons that are at same depth \Rightarrow Archi
we consider

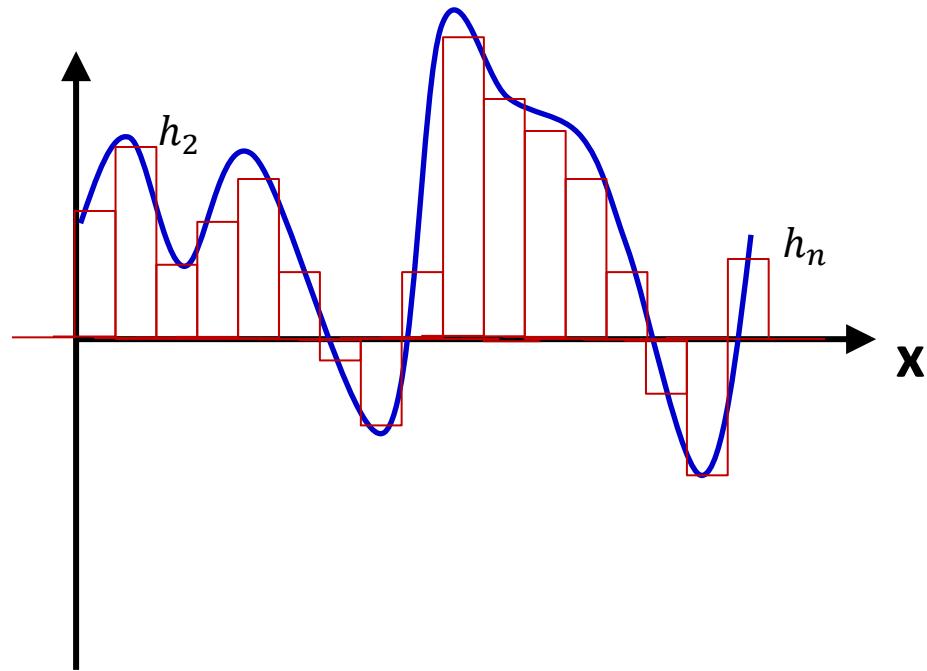
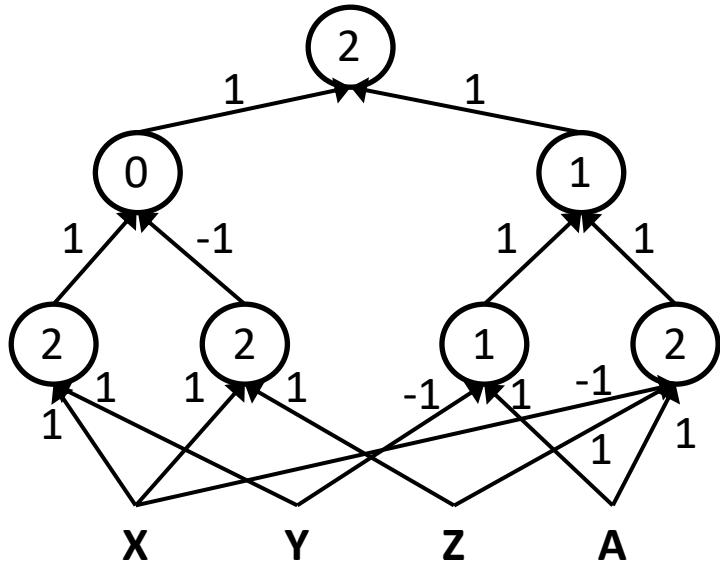
The multi-layer perceptron



- 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

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



- 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

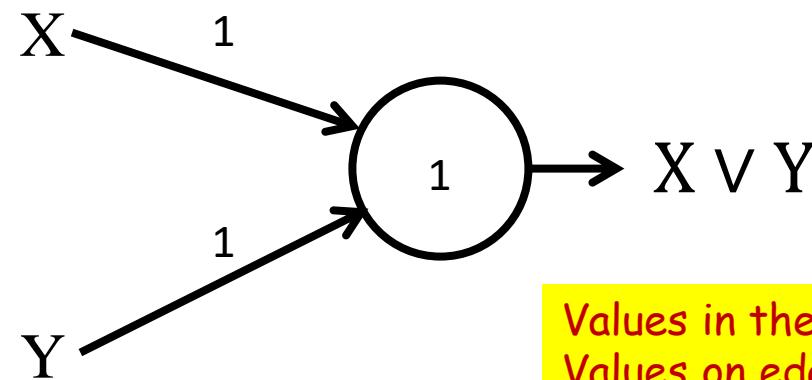
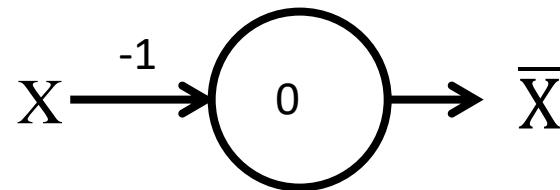
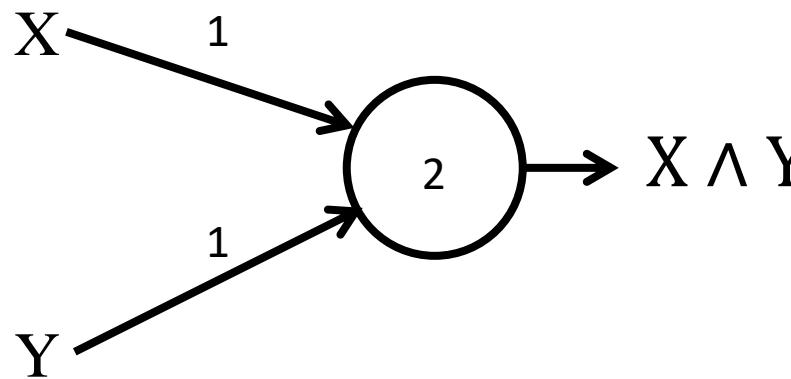
Today

- 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?

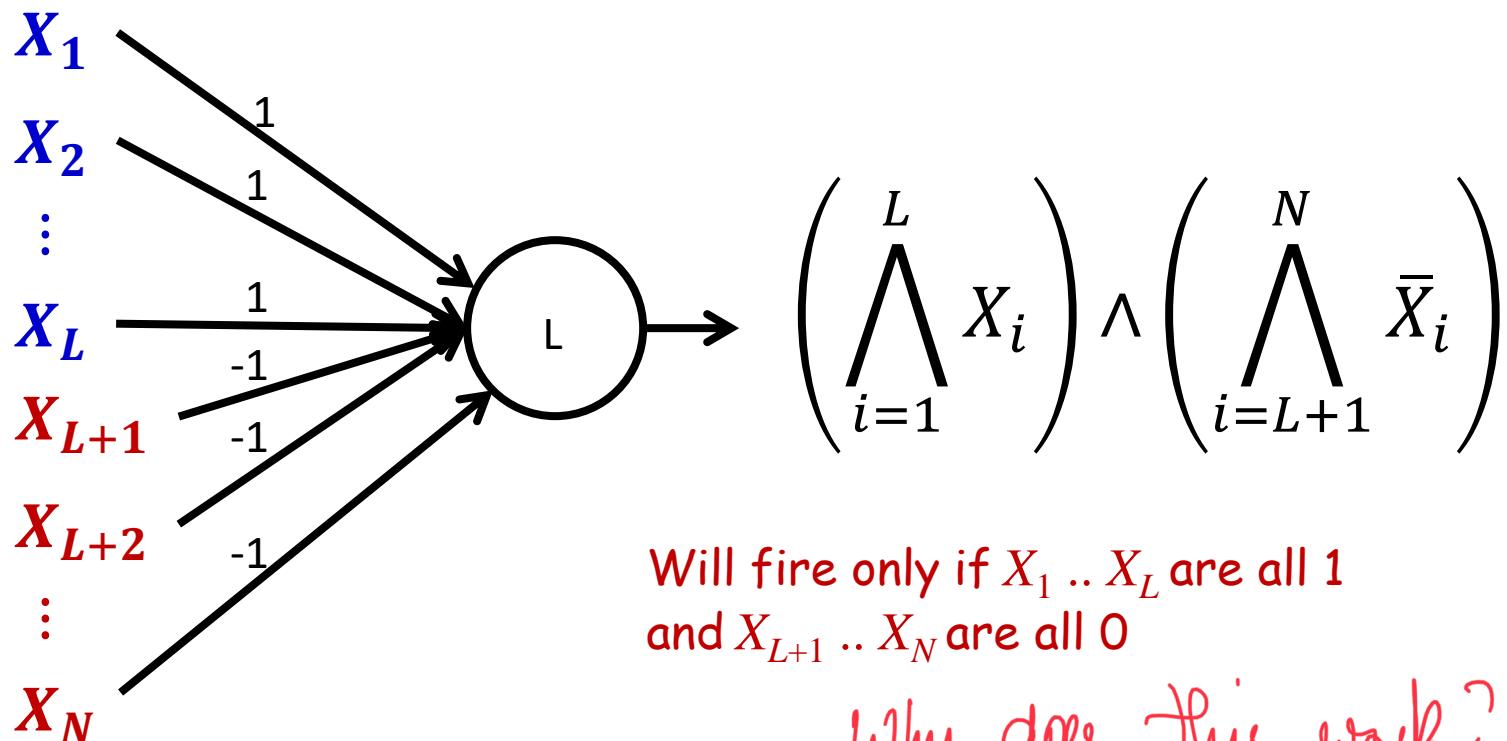
The perceptron as a Boolean gate



Values in the circles are thresholds
Values on edges are weights

- A perceptron can model any simple binary Boolean gate

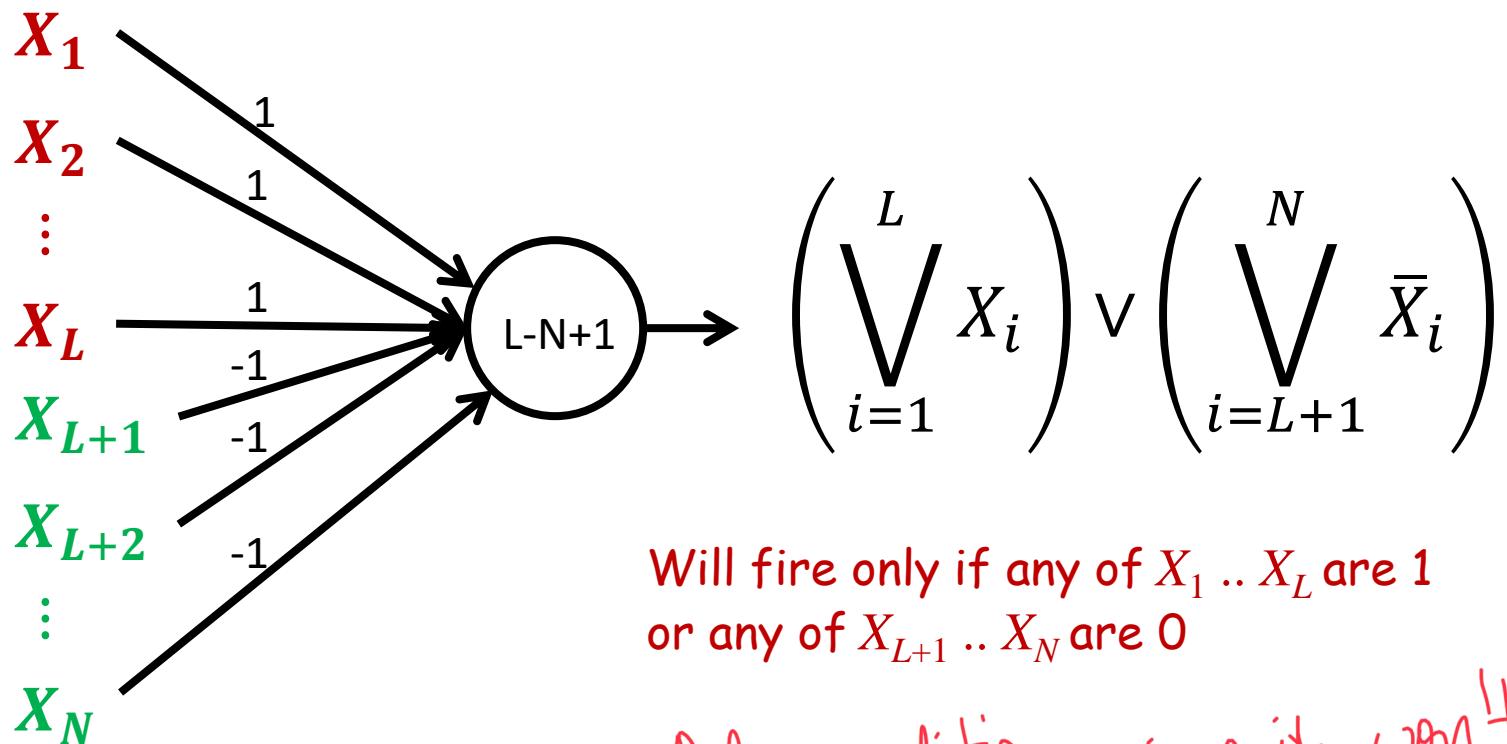
Perceptron as a Boolean gate



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

Why does this work?
- - Threshold = L

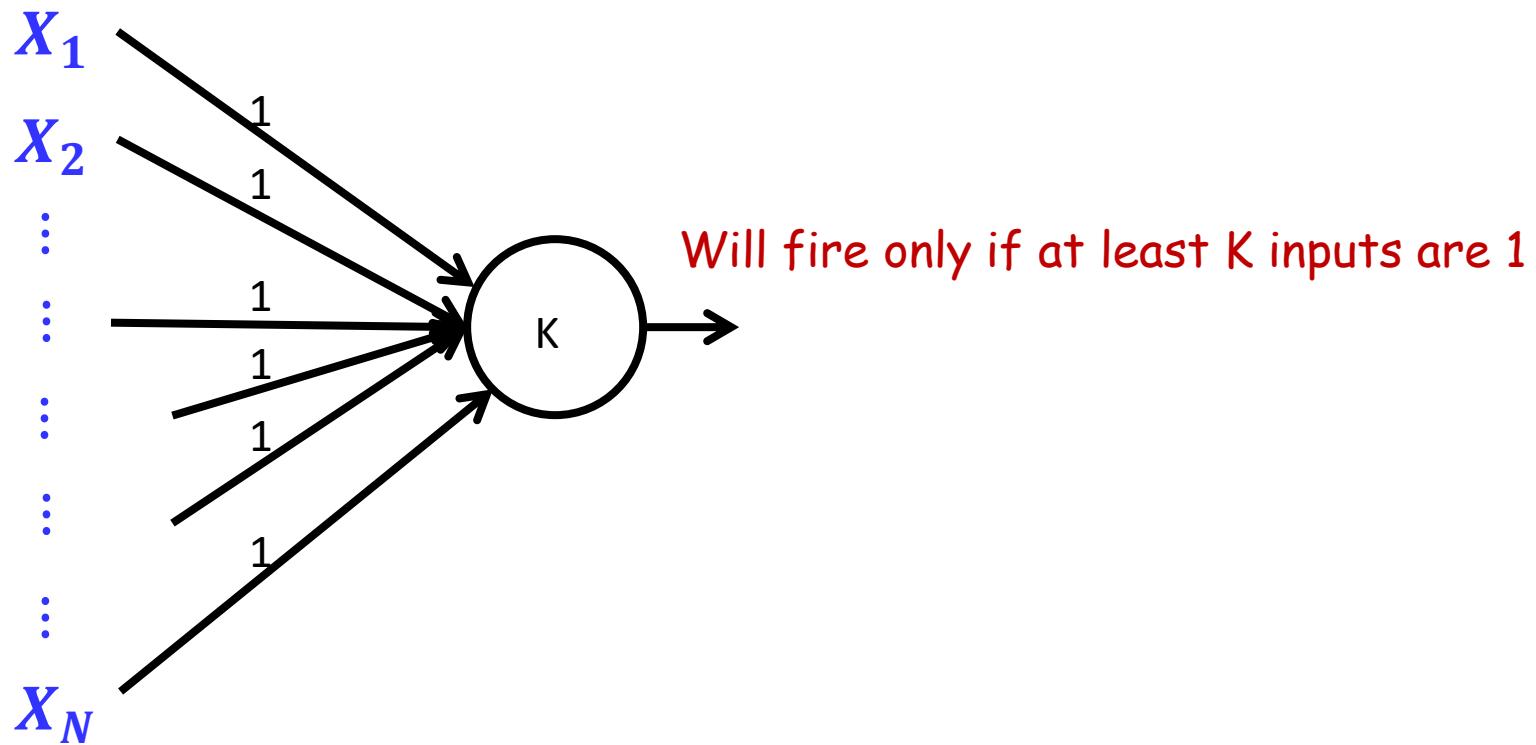
Perceptron as a Boolean gate



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

Only condition where it won't
fire is $\Rightarrow L=0 \& L+1=N-1$

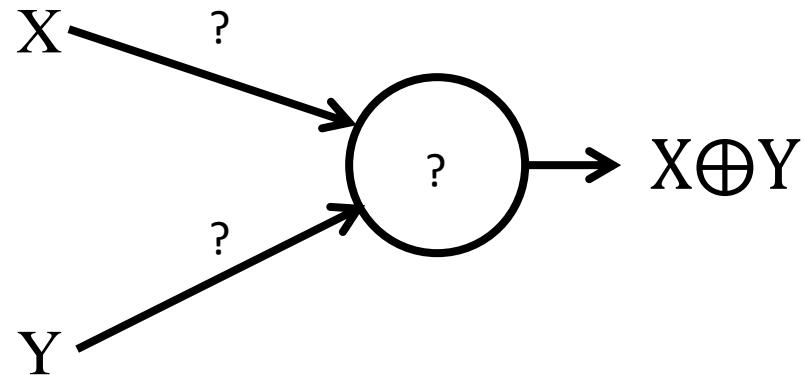
Perceptron as a Boolean Gate



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

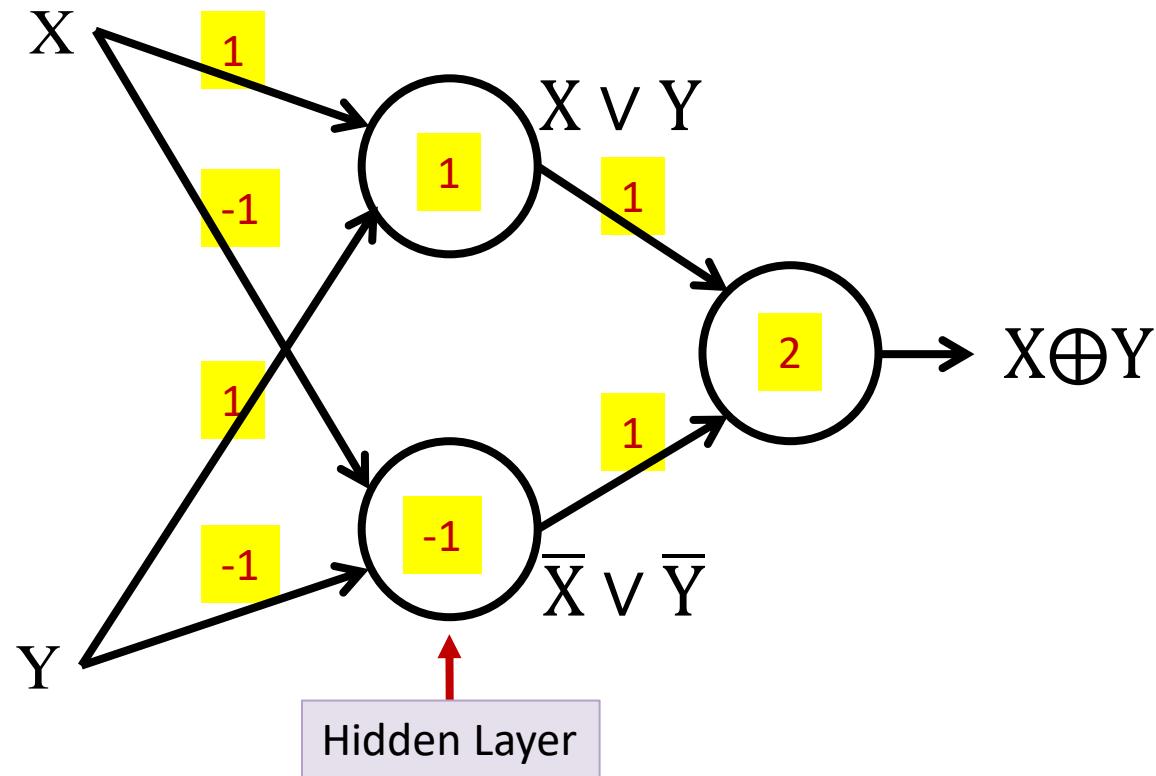
If you're trying to implement majority gate w/ boolean gates
you need a lot of exponential size; while percep is linear

The perceptron is not enough



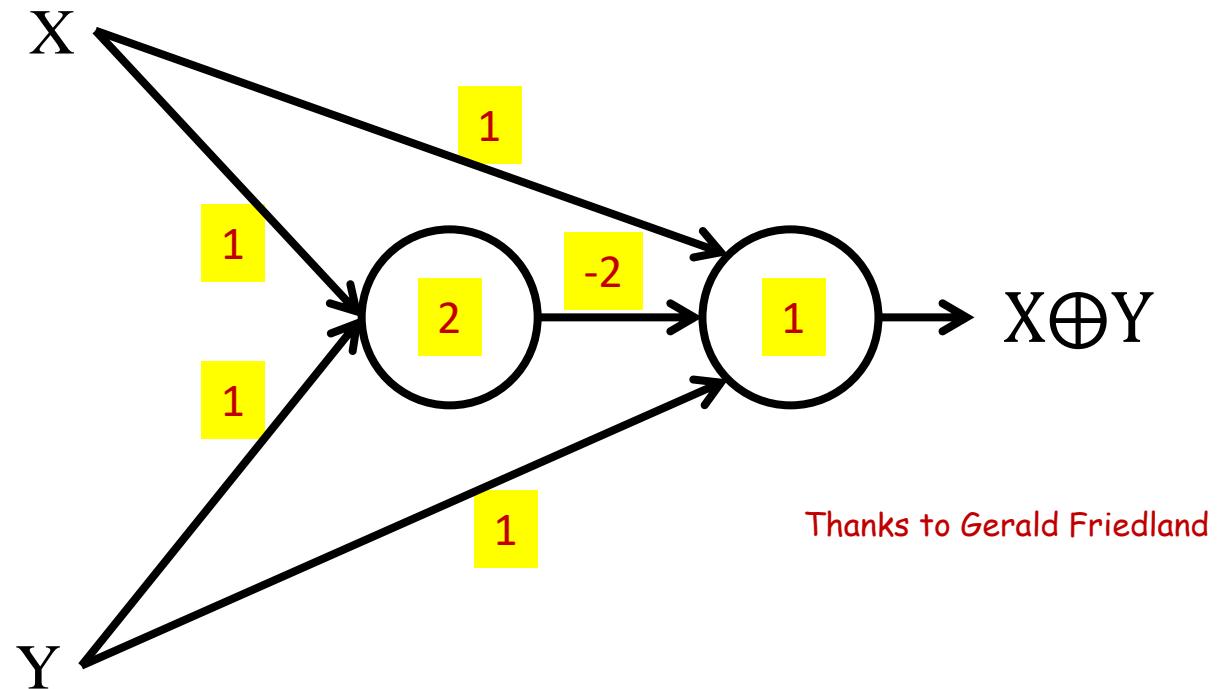
- Cannot compute an XOR

Multi-layer perceptron



- MLPs can compute the XOR *(with Network)*

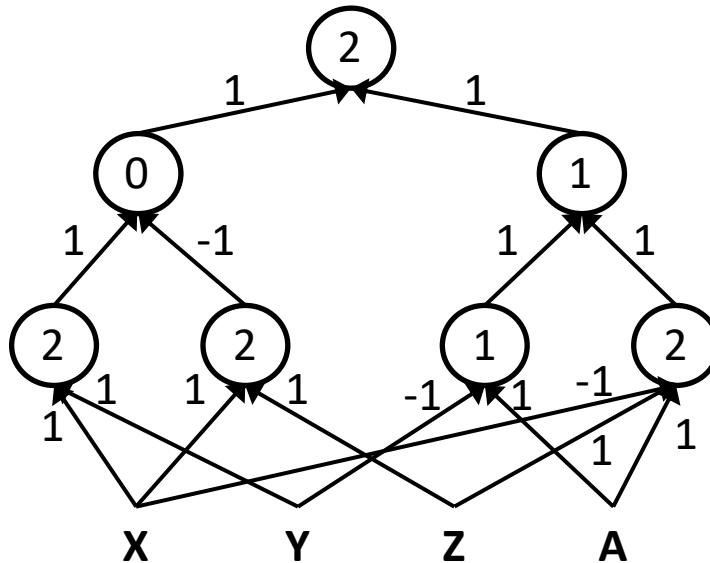
Multi-layer perceptron XOR



- With 2 neurons
 - 5 weights and two thresholds

Multi-layer perceptron

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

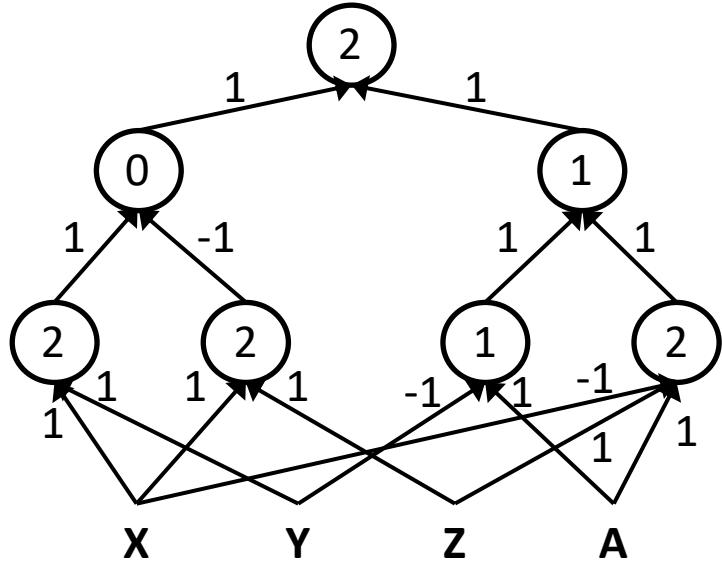


- MLPs can compute more complex Boolean functions
- MLPs can compute any Boolean function
 - Since they can emulate individual gates
- **MLPs are universal Boolean functions**

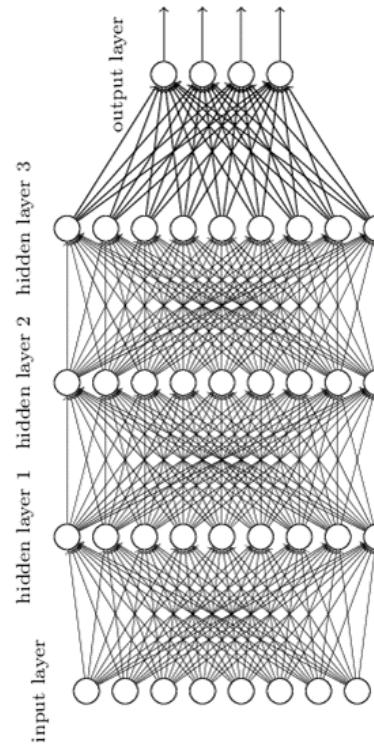
If you give a
boolean func if it is possible to design an MLP to o/p that func

MLP as Boolean Functions

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



Deep neural network



- 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

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

- A Boolean function is just a truth table

↳ tables are arranged of

i/p & o/p where only rows w/ D/P is displayed

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

$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \bar{X}_1 X_2 \bar{X}_3 X_4 X_5 + \bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5 + X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 X_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + X_1 X_2 \bar{X}_3 \bar{X}_4 X_5$$

- Expressed in disjunctive normal form

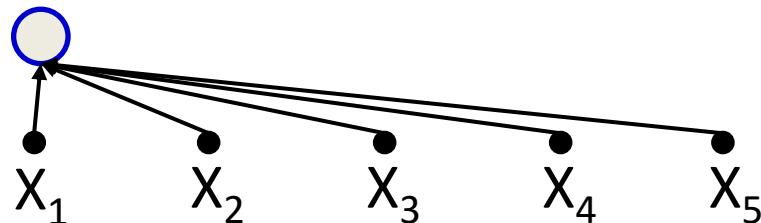
How many layers for a Boolean MLP?

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

$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \bar{X}_1 X_2 \bar{X}_3 X_4 X_5 + \bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + X_1 X_2 \bar{X}_3 \bar{X}_4 X_5$$



- Expressed in disjunctive normal form

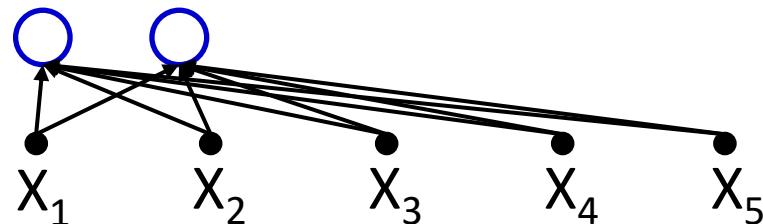
How many layers for a Boolean MLP?

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

$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \textcircled{X}_1 X_2 \bar{X}_3 X_4 X_5 + \bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5 + X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 X_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + X_1 X_2 \bar{X}_3 \bar{X}_4 X_5$$



- Expressed in disjunctive normal form

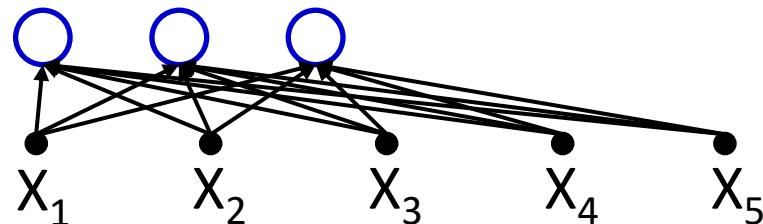
How many layers for a Boolean MLP?

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

$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \bar{X}_1 X_2 \bar{X}_3 X_4 X_5 + \textcircled{\bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5} + \\ X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 X_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + X_1 X_2 \bar{X}_3 X_4 X_5$$



- Expressed in disjunctive normal form

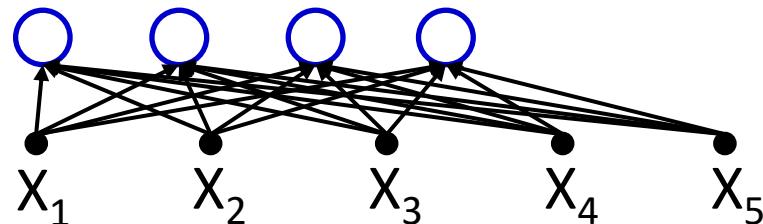
How many layers for a Boolean MLP?

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

$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \bar{X}_1 X_2 \bar{X}_3 X_4 X_5 + \bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5 + \text{ (circled term)} \\ X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 X_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + X_1 X_2 \bar{X}_3 \bar{X}_4 X_5$$



- Expressed in disjunctive normal form

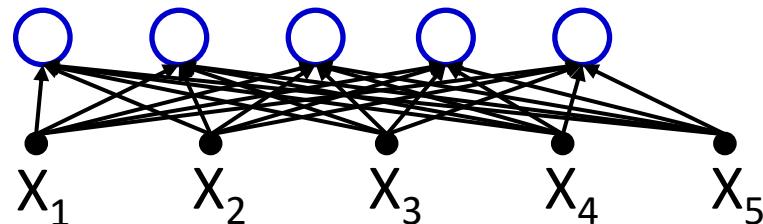
How many layers for a Boolean MLP?

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

$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \bar{X}_1 X_2 \bar{X}_3 X_4 X_5 + \bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5 + X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 X_5 + \textcircled{X}_1 \bar{X}_2 X_3 X_4 X_5 + X_1 X_2 \bar{X}_3 \bar{X}_4 X_5$$



- Expressed in disjunctive normal form

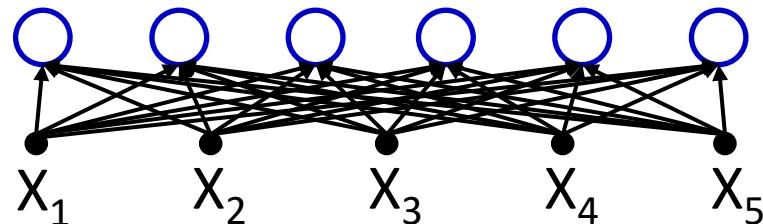
How many layers for a Boolean MLP?

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

$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \bar{X}_1 X_2 \bar{X}_3 X_4 X_5 + \bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5 + X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 X_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + \textcircled{X}_1 X_2 \bar{X}_3 \bar{X}_4 X_5$$



- Expressed in disjunctive normal form

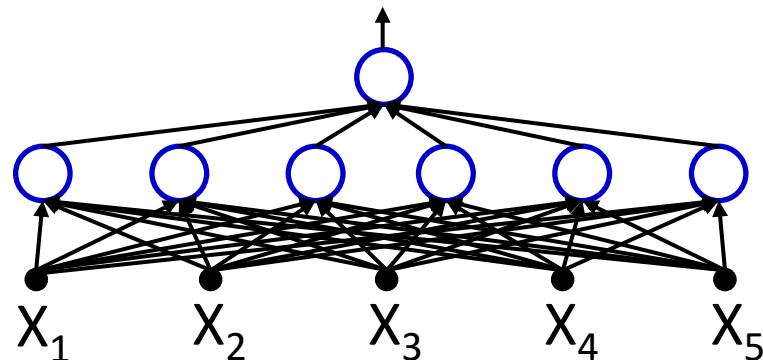
How many layers for a Boolean MLP?

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

$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \bar{X}_1 X_2 \bar{X}_3 X_4 X_5 + \bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5 + X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 X_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + X_1 X_2 \bar{X}_3 \bar{X}_4 X_5$$



- Expressed in disjunctive normal form

+ → OR
- → AND

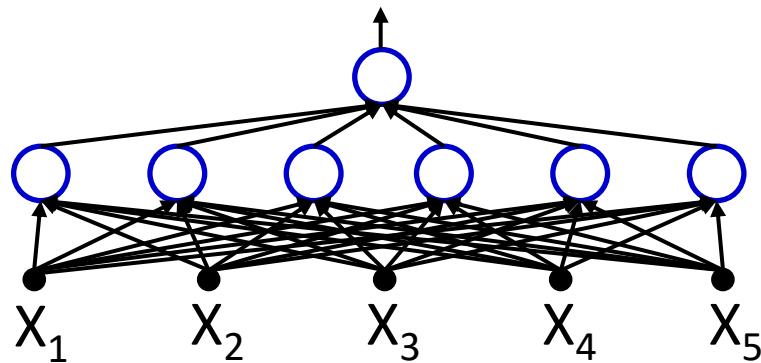
How many layers for a Boolean MLP?

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

$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \bar{X}_1 X_2 \bar{X}_3 X_4 X_5 + \bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5 + X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 X_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + X_1 X_2 \bar{X}_3 \bar{X}_4 X_5$$



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

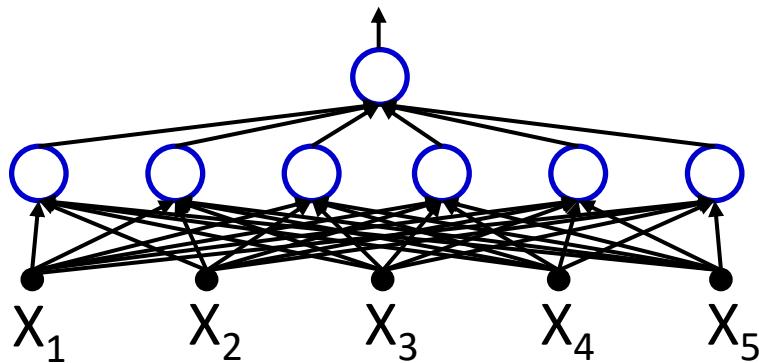
How many layers for a Boolean MLP?

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

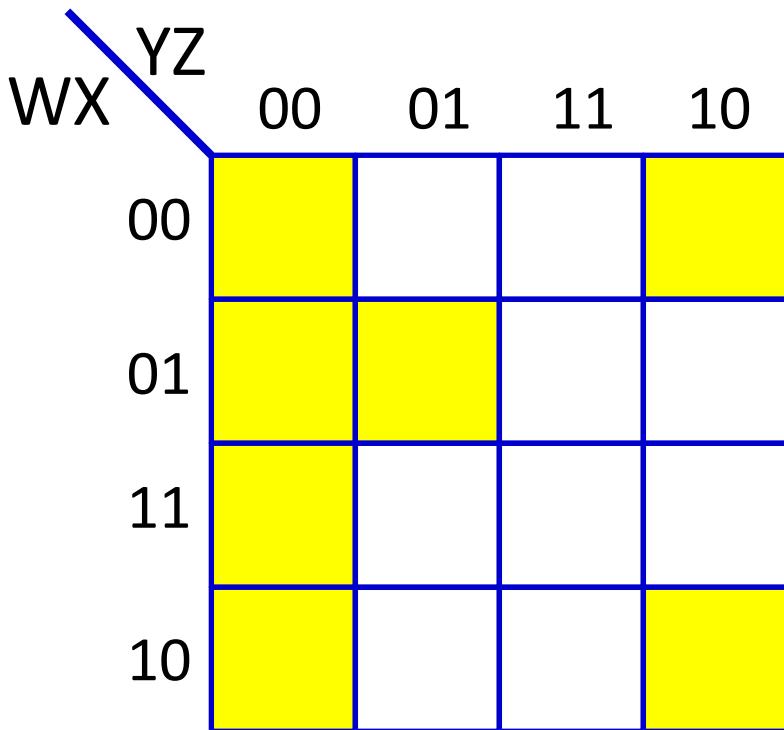
$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \bar{X}_1 X_2 \bar{X}_3 X_4 X_5 + \bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5 + X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 X_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + X_1 X_2 \bar{X}_3 \bar{X}_4 X_5$$



- 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



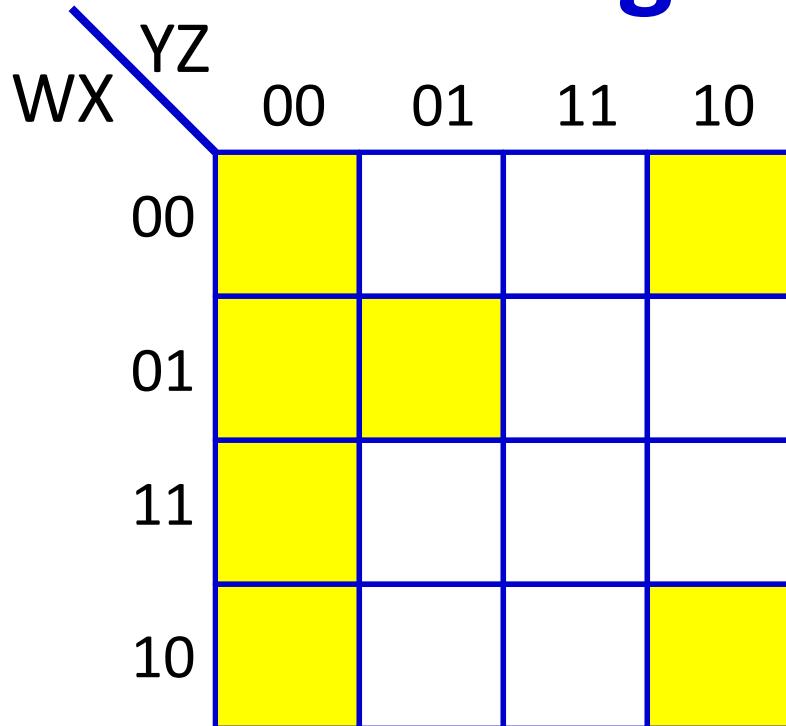
Topological way to arrange i/p
This is a "Karnaugh Map"

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

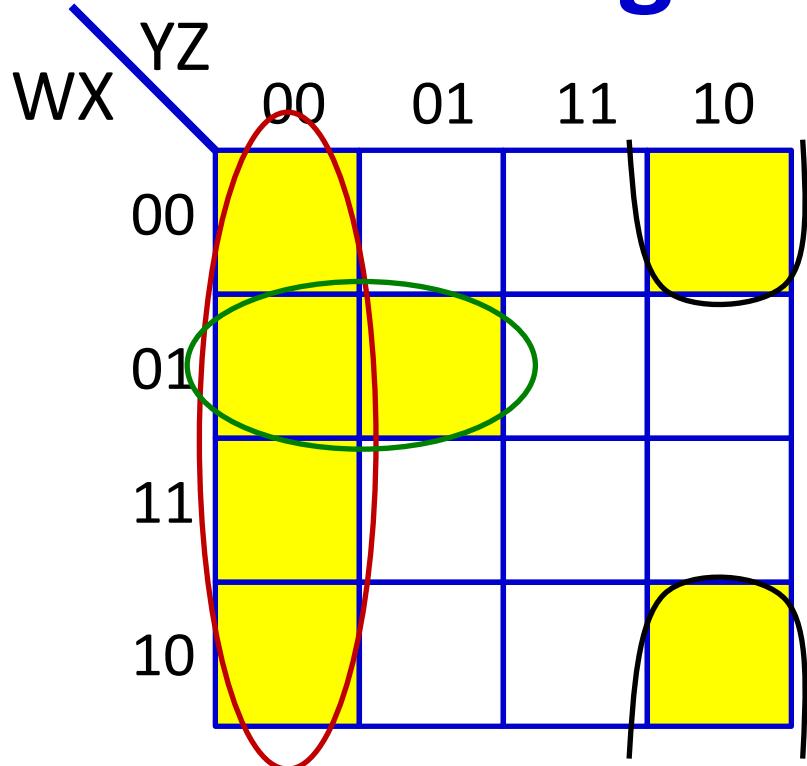
- DNF form:
 - Find groups
 - Express as reduced DNF

Reducing a Boolean Function



Basic DNF formula will require 7 terms

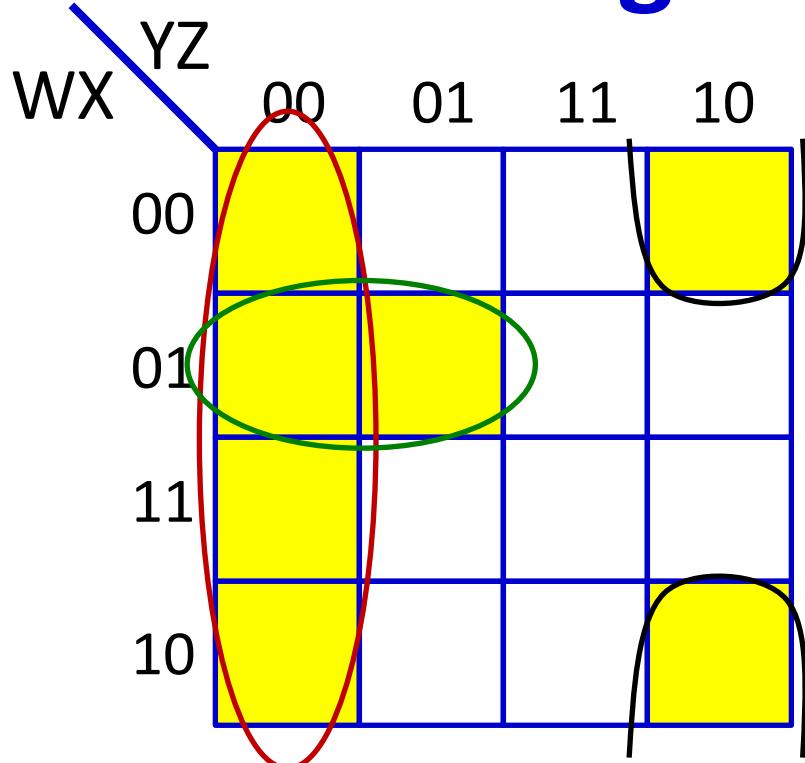
Reducing a Boolean Function



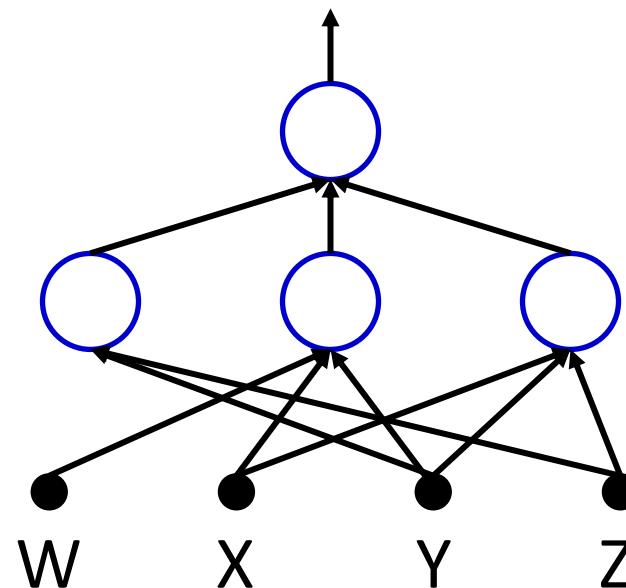
$$O = \bar{Y}\bar{Z} + \bar{W}X\bar{Y} + \bar{X}YZ$$

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

Reducing a Boolean Function



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



- *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

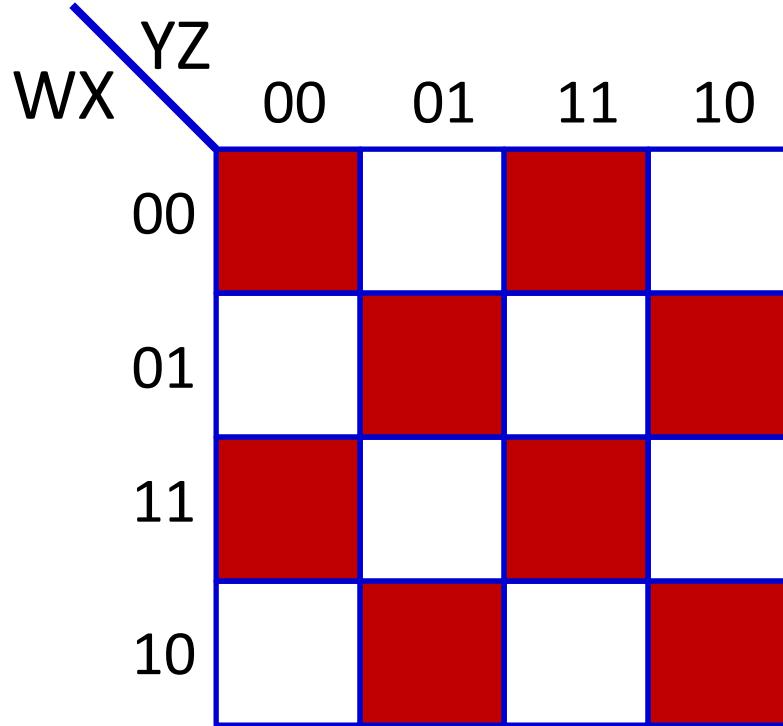
Largest irreducible DNF?

	00	01	11	10
00				
01				
11				
10				

WX \ YZ

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

Largest irreducible DNF?

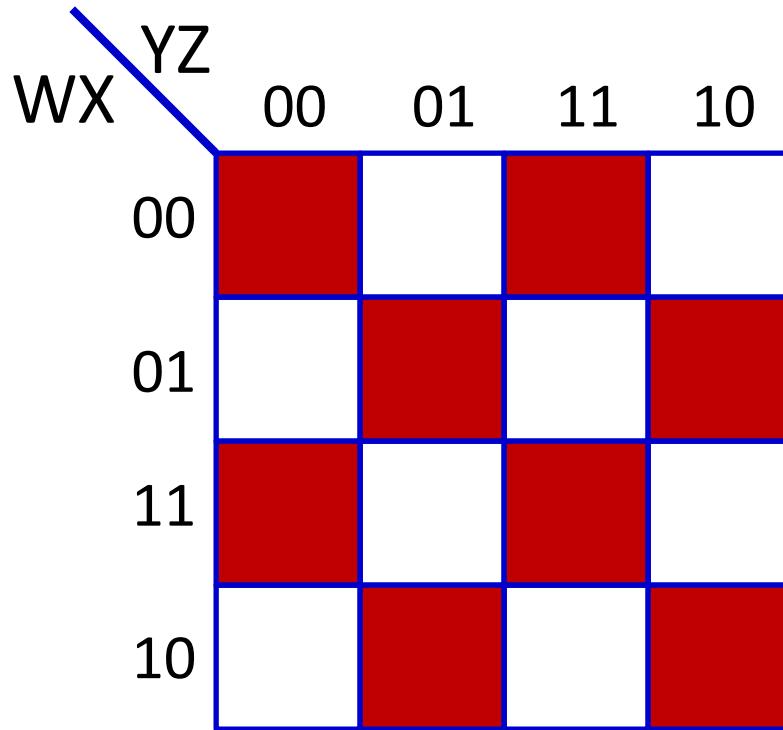


Red=0, white=1

↳ ORS of ANTS

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

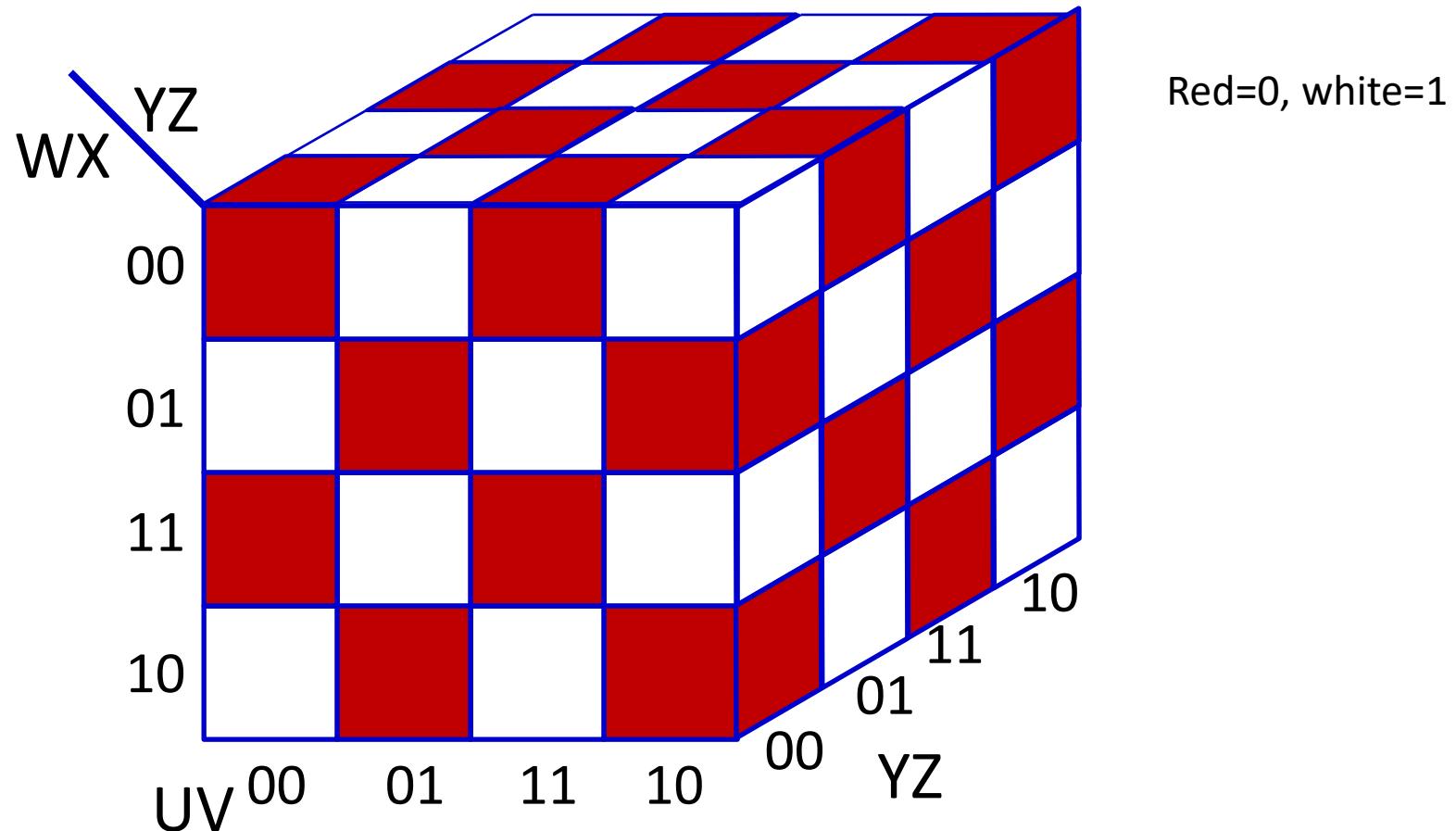
Largest irreducible DNF?



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

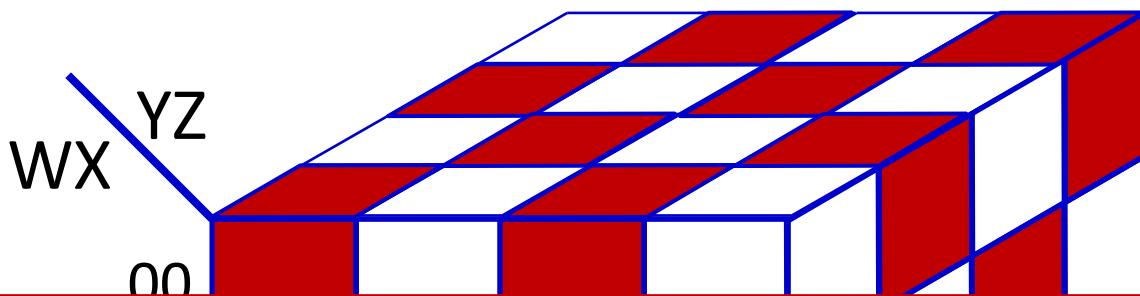
- What arrangement of ones and zeros simply cannot be reduced further?

Width of a one-hidden-layer Boolean MLP

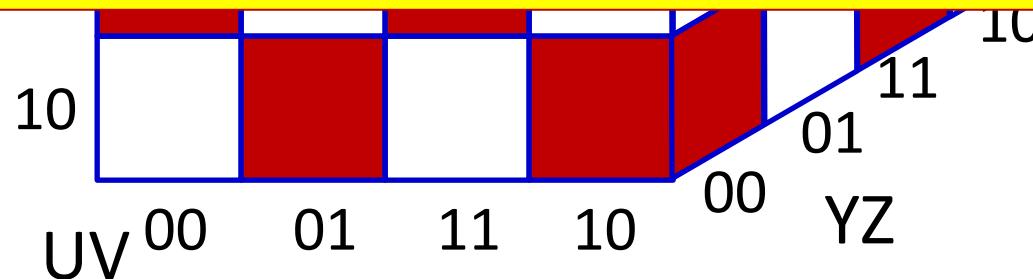


- How many neurons in a DNF (one-hidden-layer) MLP for this Boolean function of 6 variables?

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

Poll 2

How many neurons will be required in the hidden layer of a one-hidden-layer network that models a Boolean function over 10 inputs, where the output for two input bit patterns that differ in only one bit is always different? (I.e. the checkerboard Karnaugh map)

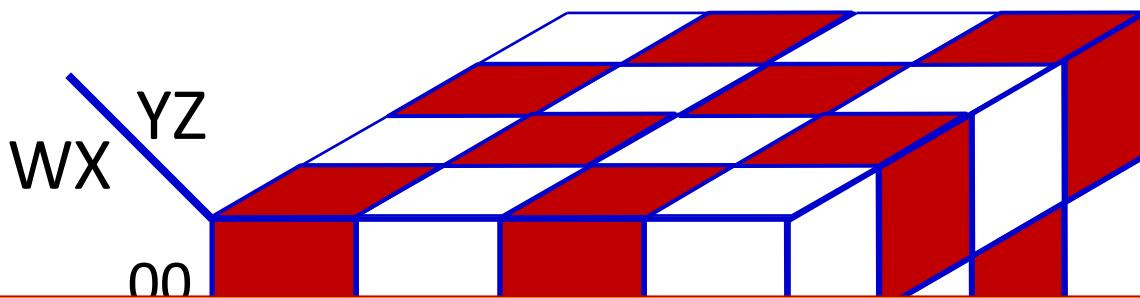
- 20
- 256
- 512
- 1024

Poll 2

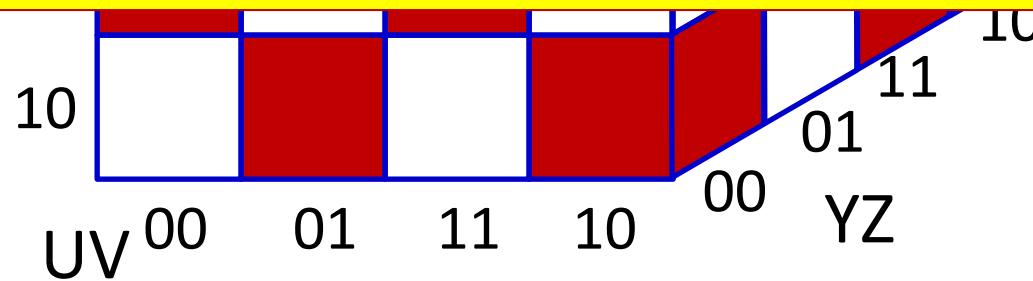
How many neurons will be required in the hidden layer of a one-hidden-layer network that models a Boolean function over 10 inputs, where the output for two input bit patterns that differ in only one bit is always different? (I.e. the checkerboard Karnaugh map)

- 20
- 256
- **512**
- 1024

Width of a one-hidden-layer Boolean MLP

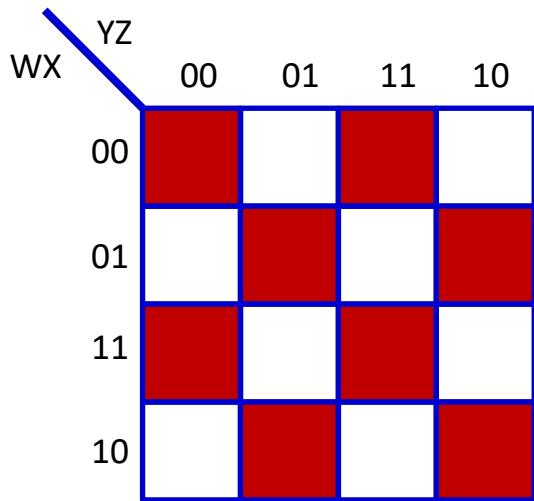


Can be generalized: Will require 2^{N-1} perceptrons in hidden layer
Exponential in N

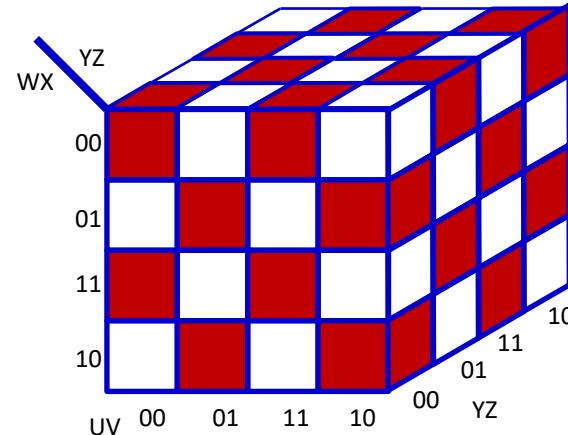


How many units if we use *multiple hidden layers*?

Size of a deep MLP

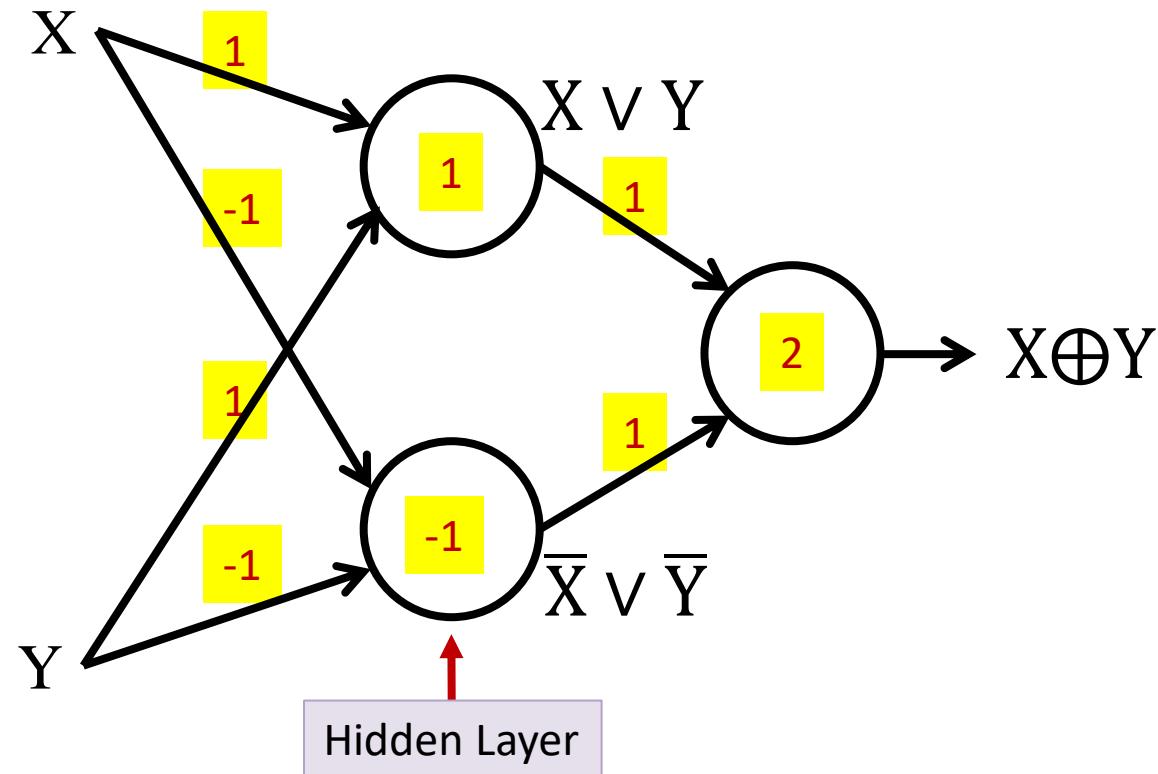


$$O = W \oplus X \oplus Y \oplus Z$$



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

Multi-layer perceptron XOR

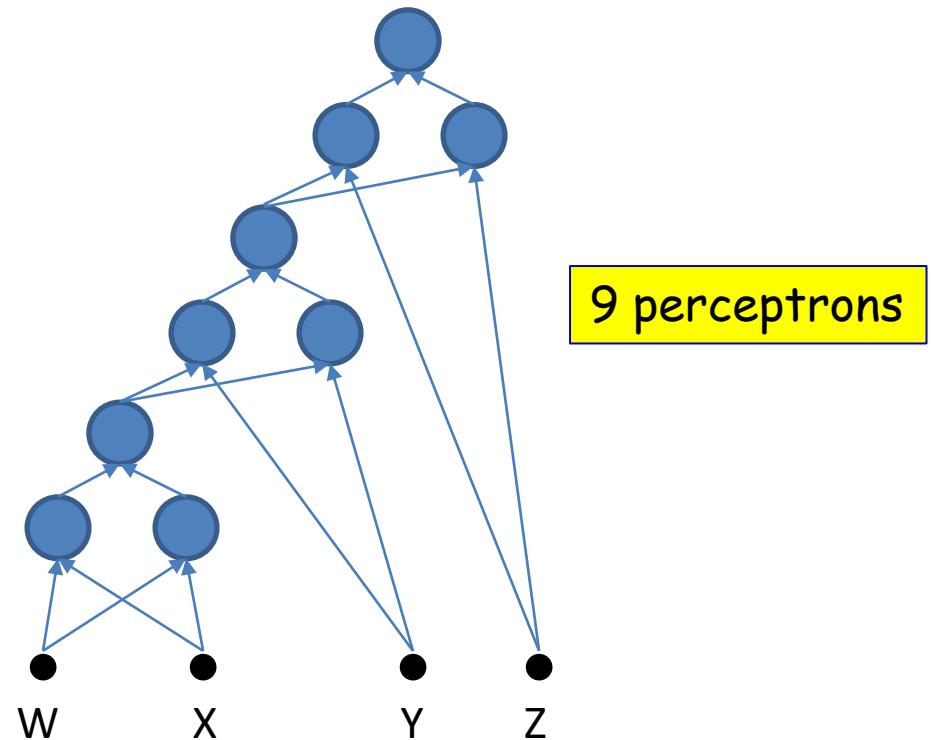


- An XOR takes three perceptrons

Size of a deep MLP

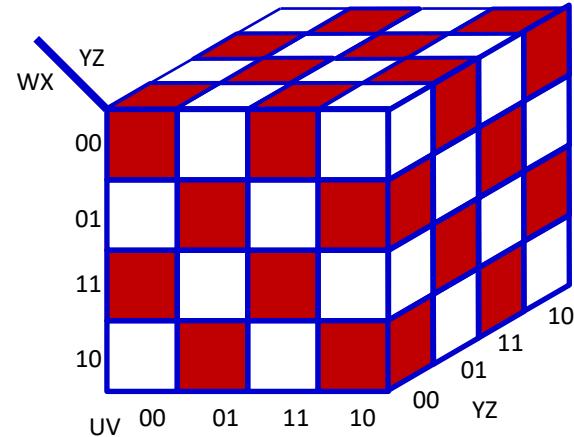
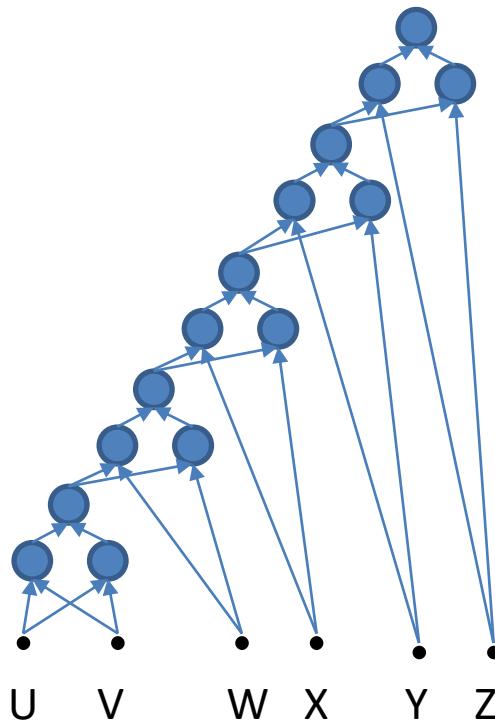
	WX	YZ	
00	00	01	11
01	01	11	00
11	11	00	01
10	00	11	10

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



- An XOR needs 3 perceptrons
- This network will require $3 \times 3 = 9$ perceptrons

Size of a deep MLP

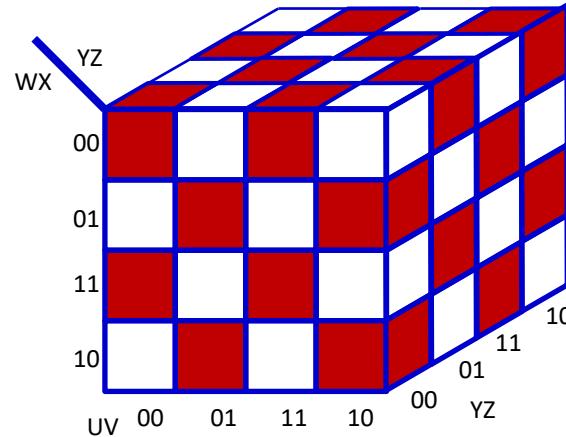
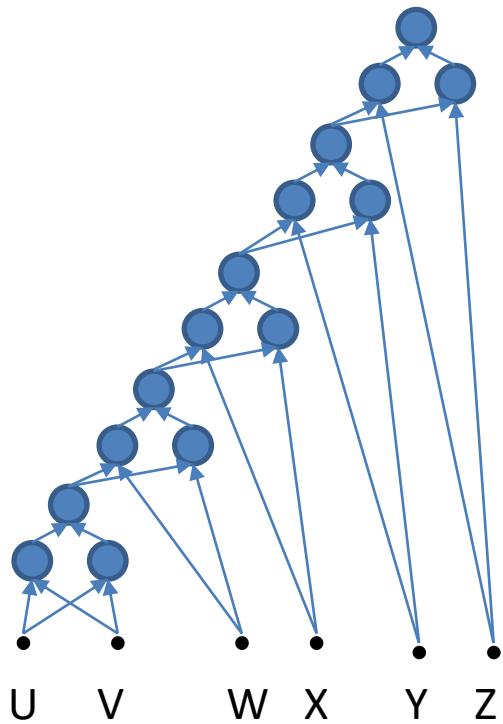


$$O = U \oplus V \oplus W \oplus X \oplus Y \oplus Z$$

15 perceptrons

- An XOR needs 3 perceptrons
- This network will require $3 \times 5 = 15$ perceptrons

Size of a deep MLP

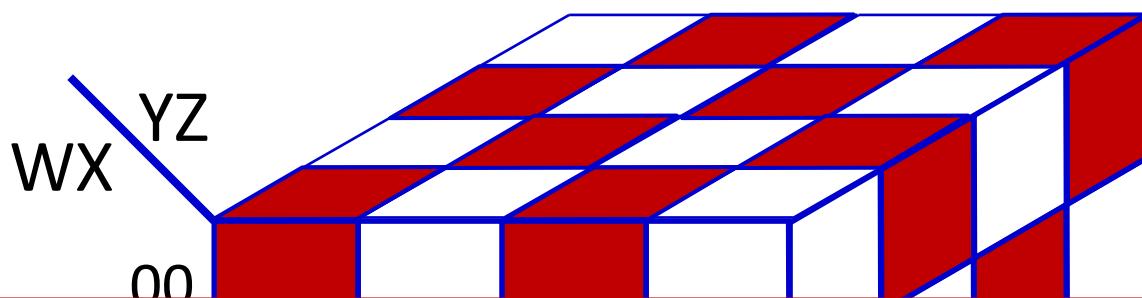


$$O = U \oplus V \oplus W \oplus X \oplus Y \oplus Z$$

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

- An XOR needs 3 perceptrons
- This network will require $3 \times 5 = 15$ 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



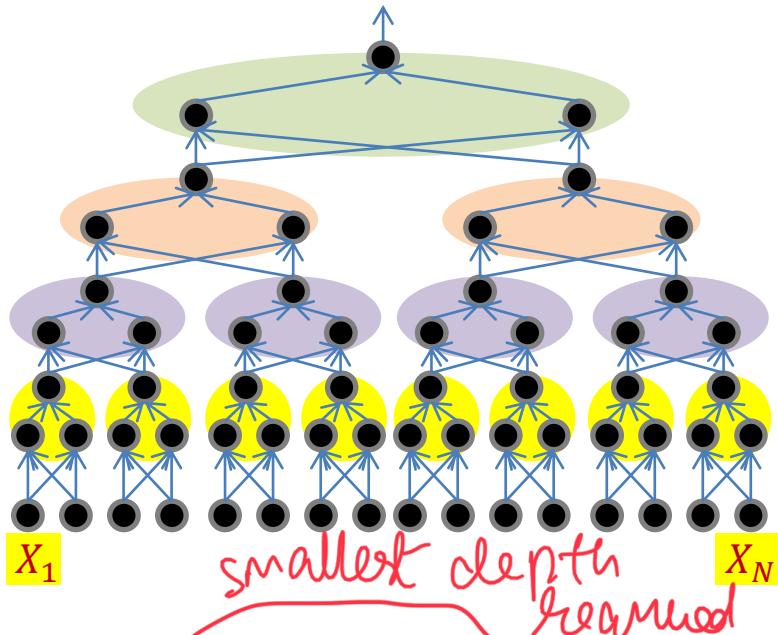
Will require $3(N-1)$ perceptrons in a deep network

Linear in N !!!

Deep MN!

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

A better representation

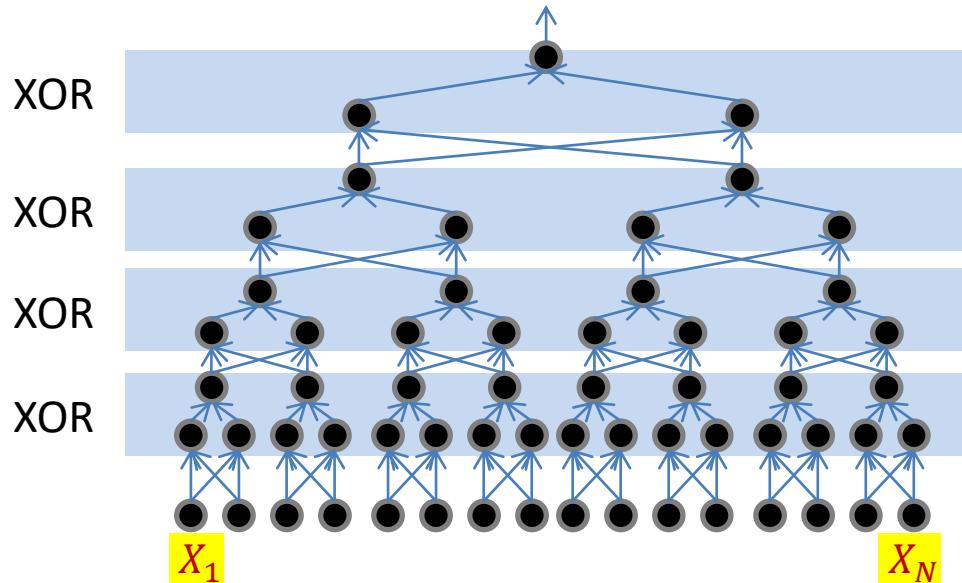


- Only $2 \log_2 N$ layers
 - By pairing terms
 - 2 layers per XOR

Each state has 2 layers,
each state reduces IP by
a factor of 2 \Rightarrow total
layers = $2 \log_2 N$

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

A better representation



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

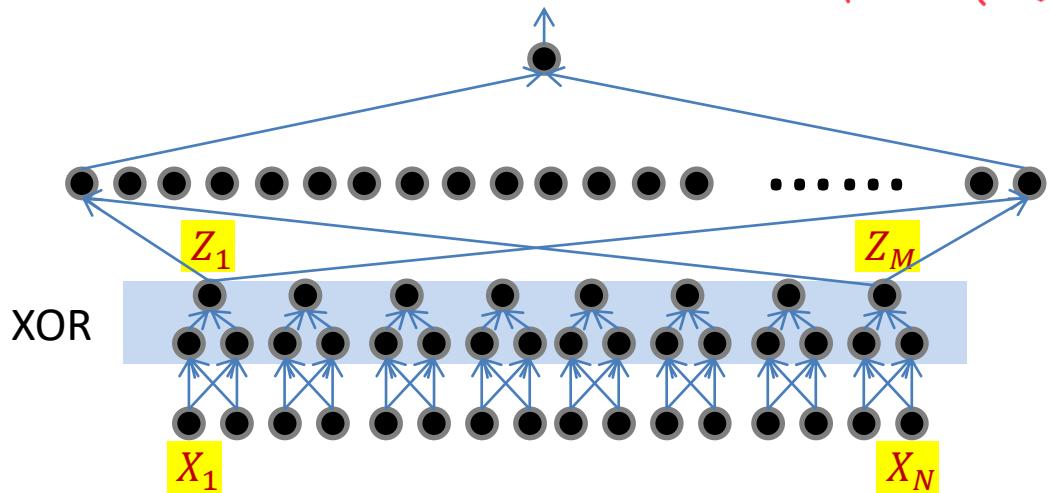
- Only $2 \log_2 N$ layers
 - By pairing terms
 - 2 layers per XOR

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

If you have a max depth to use (R) \Rightarrow Use $R-1$ layers using XOR via pairing, k for last layer

The challenge of depth

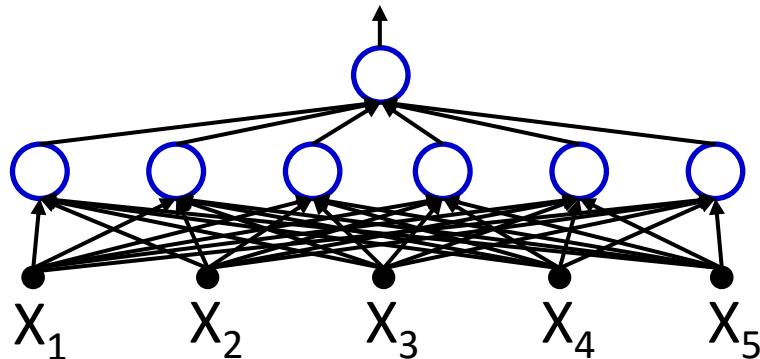
use exponential layer w/ remaining inputs -



$$\begin{aligned} O &= X_1 \oplus X_2 \oplus \cdots \oplus X_N \\ &= Z_1 \oplus Z_2 \oplus \cdots \oplus Z_M \end{aligned}$$

- Using only K hidden layers will require $O(2^{CN})$ neurons in the K th layer, where $C = 2^{-(K-1)/2}$
 - Because the output is the XOR of all the $2^{N-K/2}$ values output by the $K-1$ th 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



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

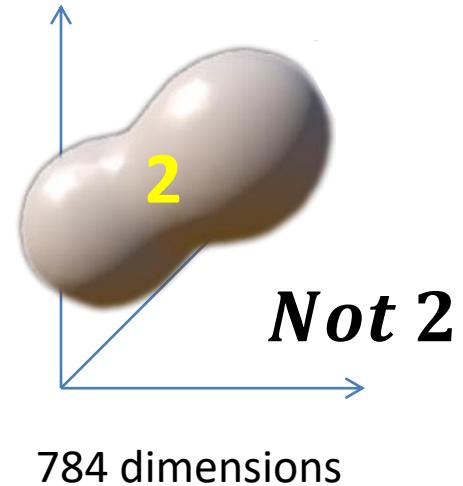
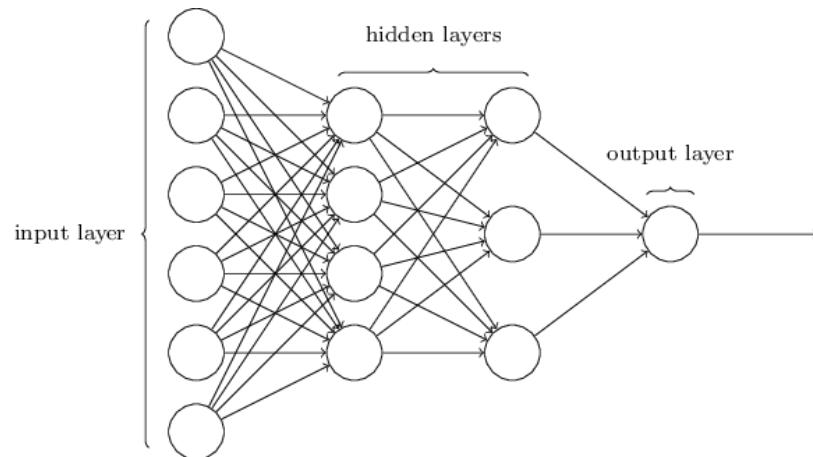
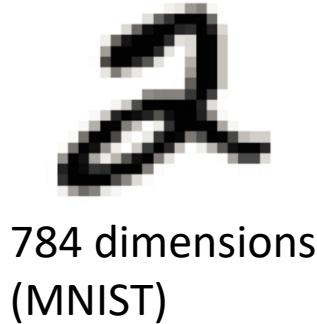
Story so far

- 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

Today

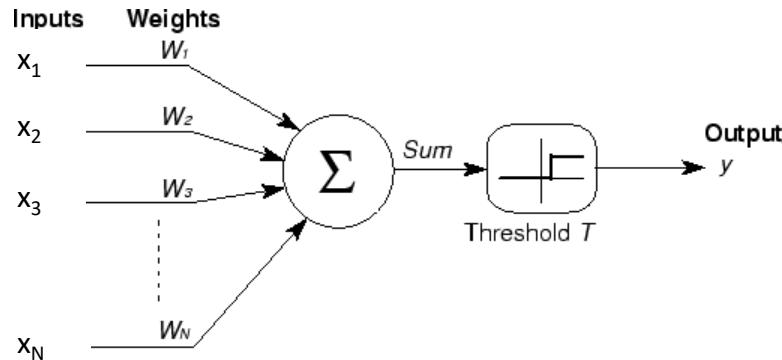
- Multi-layer Perceptrons as universal Boolean functions
 - The need for depth
- MLPs as universal classifiers
 - The need for depth
- MLPs as universal approximators
- A discussion of optimal depth and width
- Brief segue: RBF networks

Recap: The MLP as a classifier

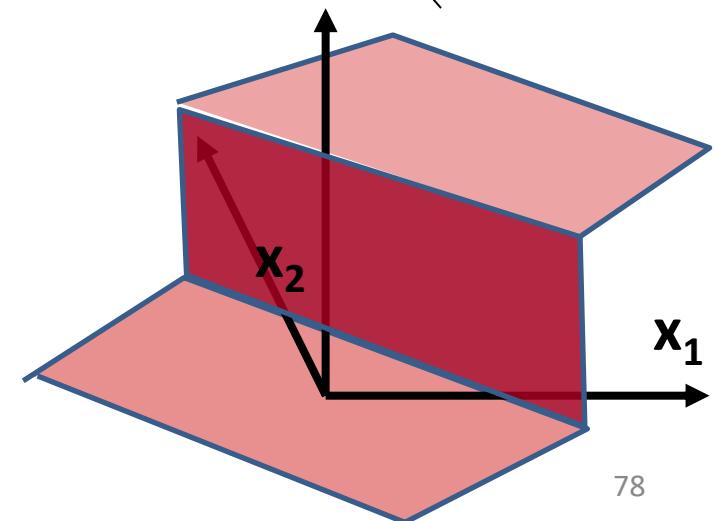
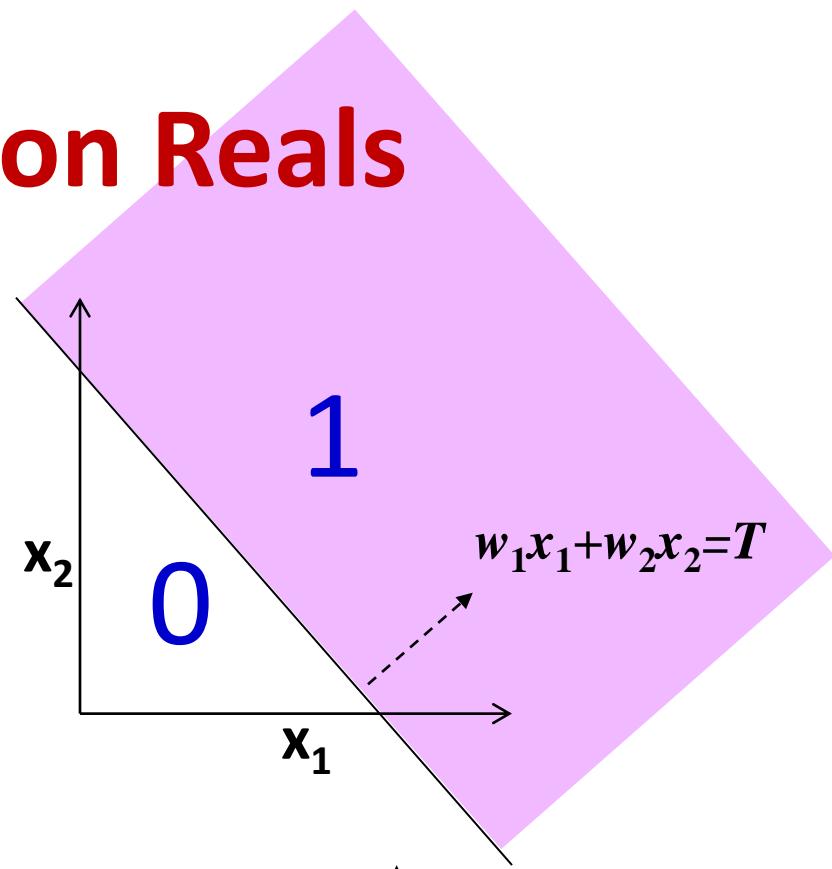


- MLP as a function over real inputs
- MLP as a function that finds a complex “decision boundary” over a space of *reals*

A Perceptron on Reals



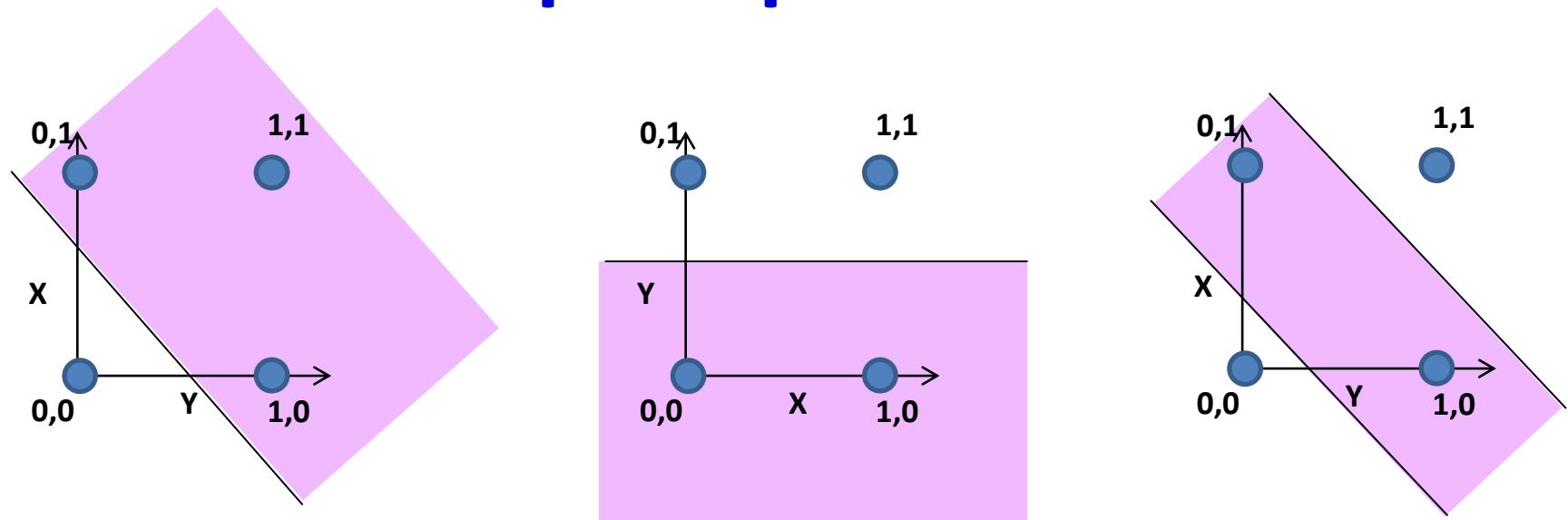
$$y = \begin{cases} 1 & \text{if } \sum_i w_i x_i \geq T \\ 0 & \text{else} \end{cases}$$



- 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

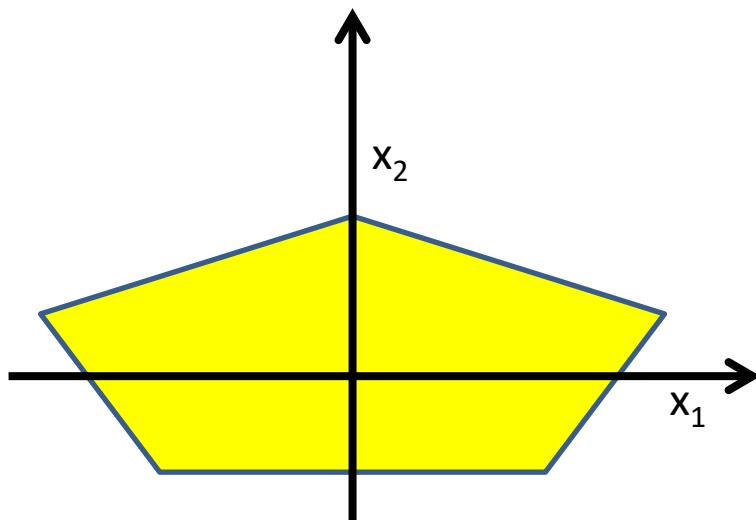
Poll 3

- An XOR network needs two hidden neurons and one output neuron, because we need one hidden neuron for each of the two boundaries of the XOR region, and an output neuron to AND them. True or false?
 - True
 - False

Poll 3

- An XOR network needs two hidden neurons and one output neuron, because we need one hidden neuron for each of the two boundaries of the XOR region, and an output neuron to AND them. True or false?
 - **True**
 - False

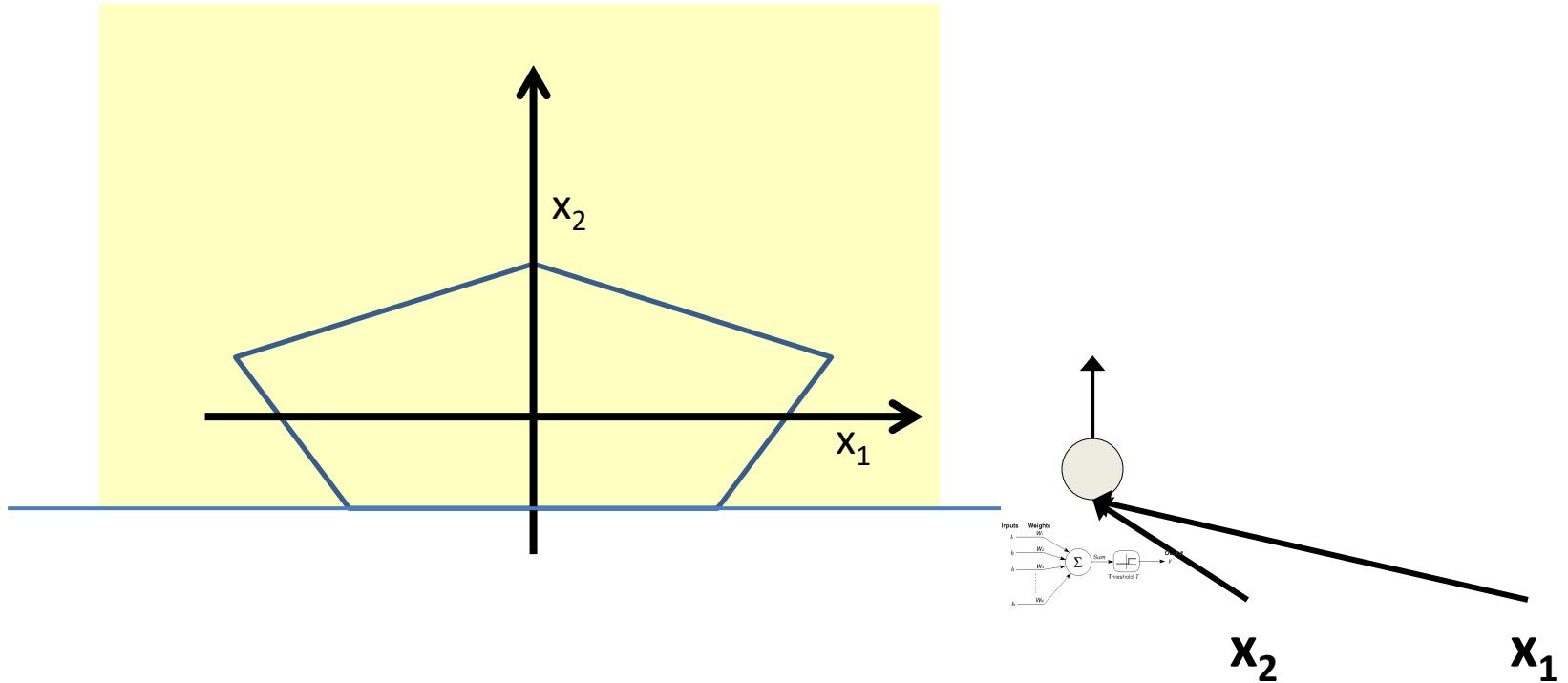
Composing complicated “decision” boundaries



Can now be composed into “networks” to compute arbitrary classification “boundaries”

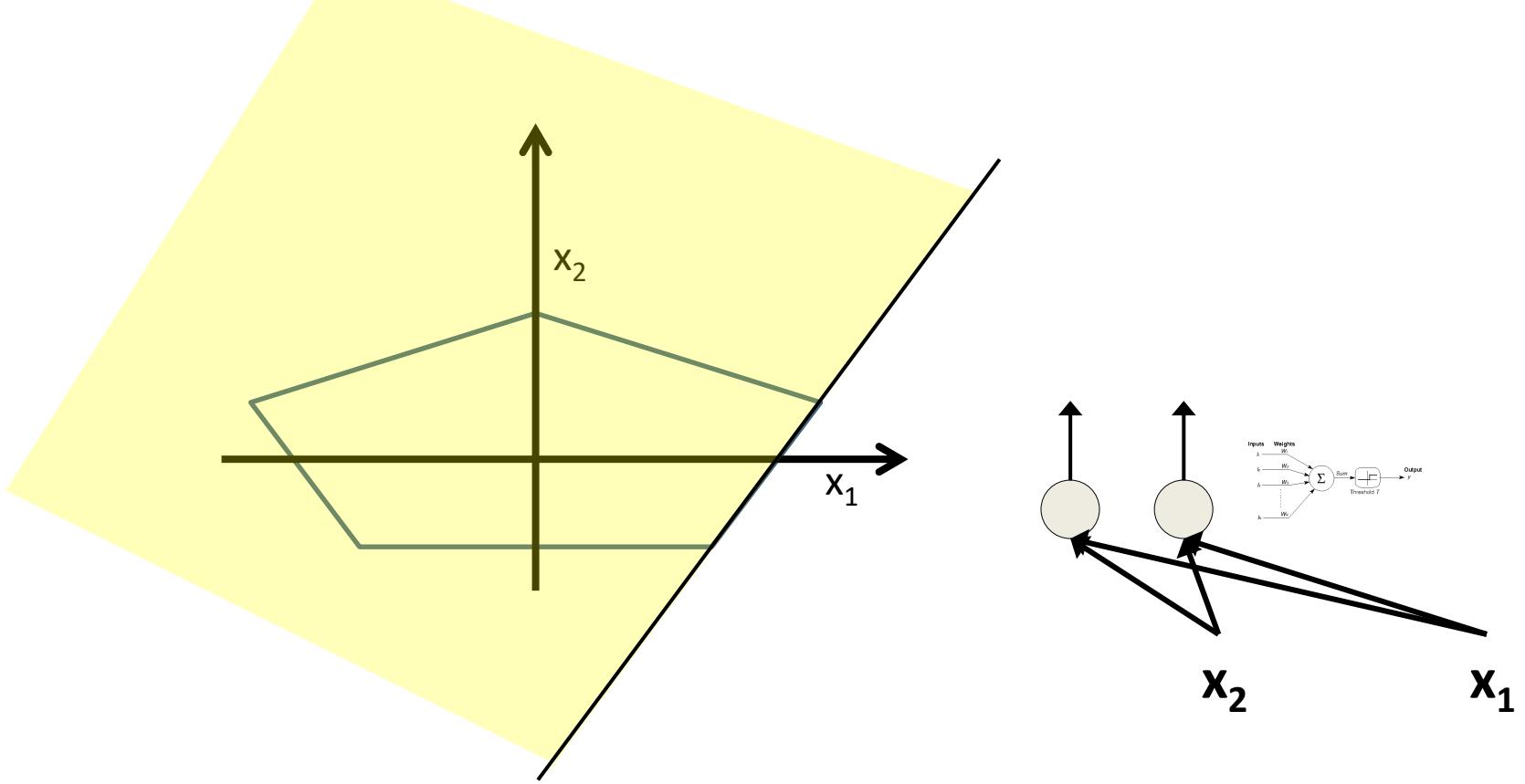
- Build a network of units with a single output that fires if the input is in the coloured area

Booleans over the reals



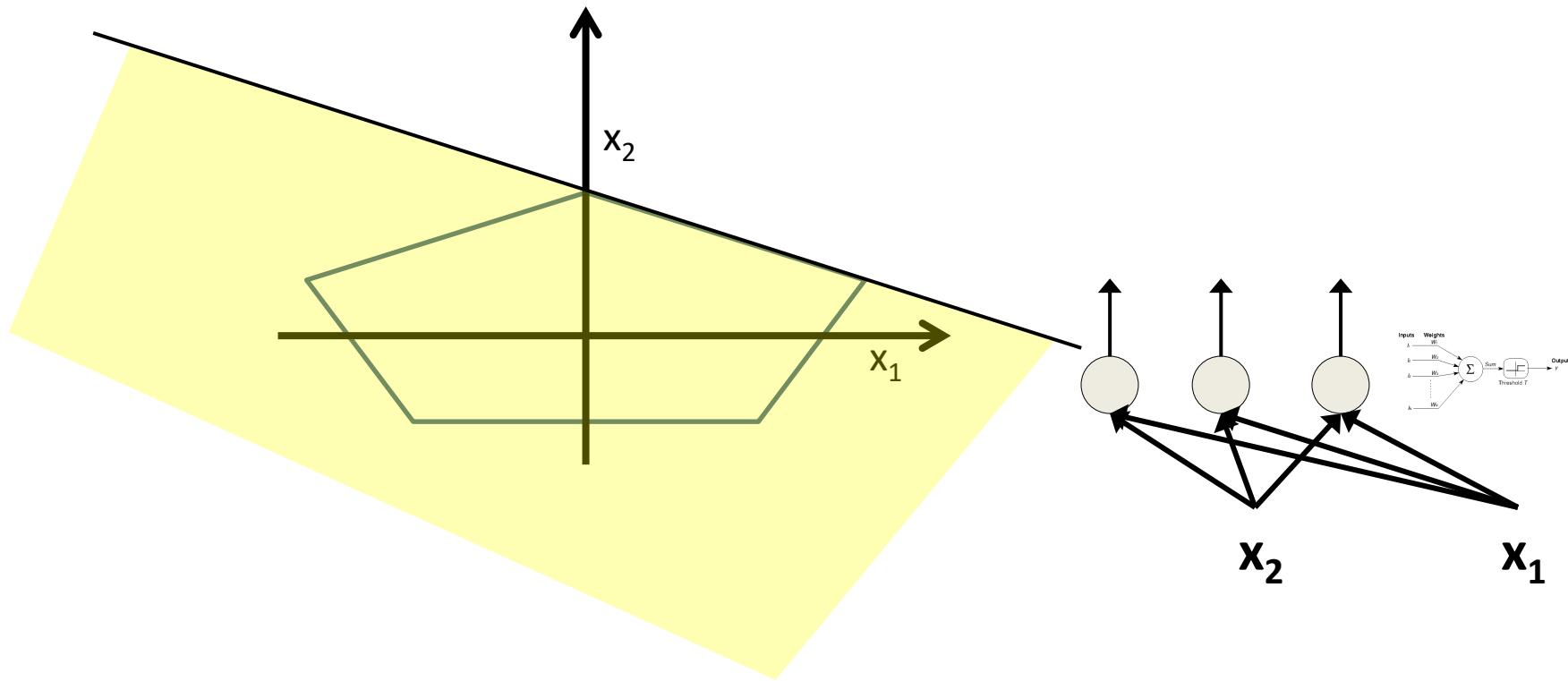
- The network must fire if the input is in the coloured area

Booleans over the reals



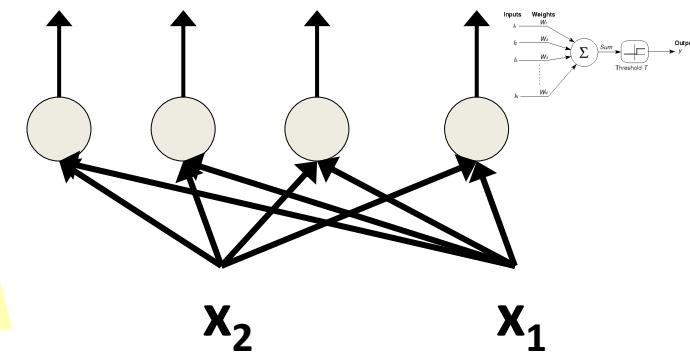
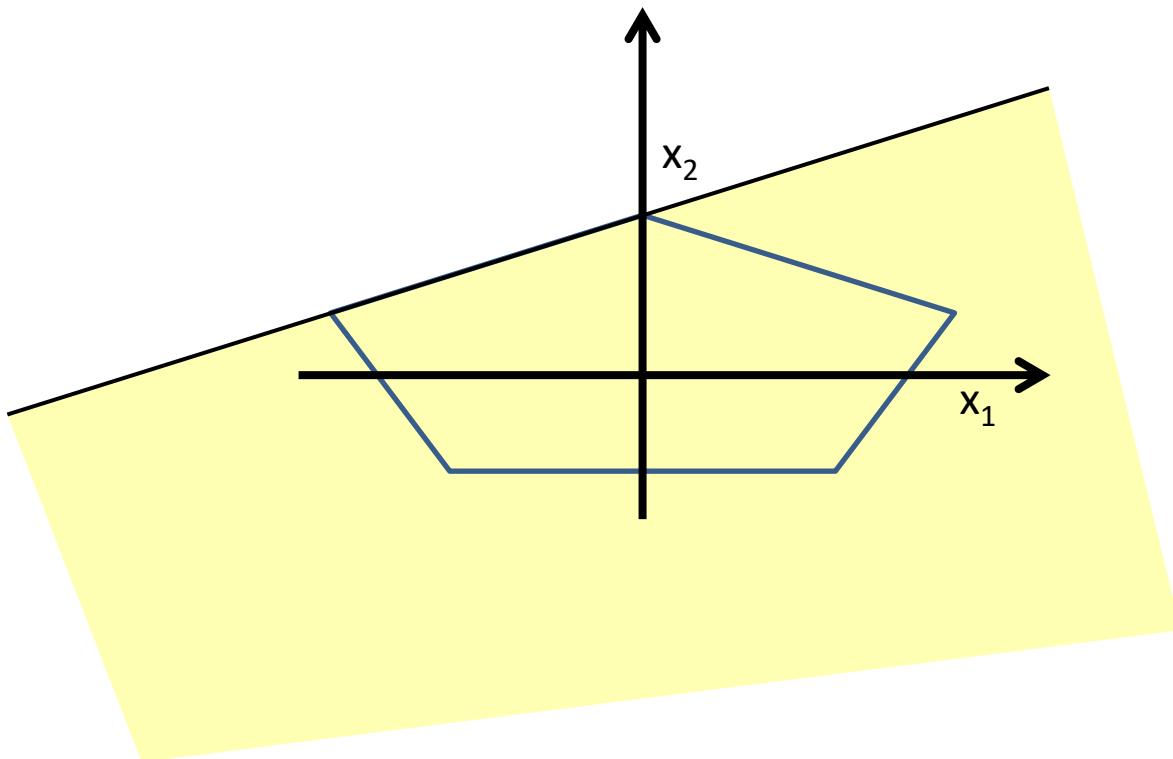
- The network must fire if the input is in the coloured area

Booleans over the reals



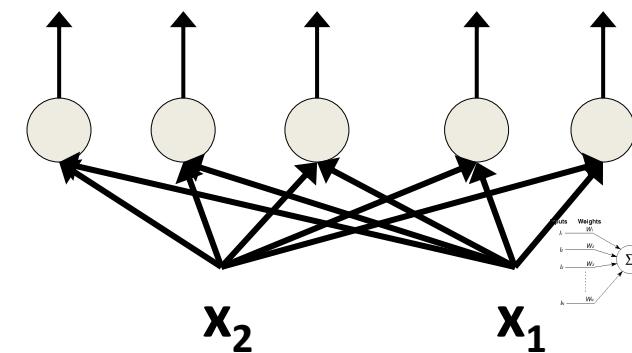
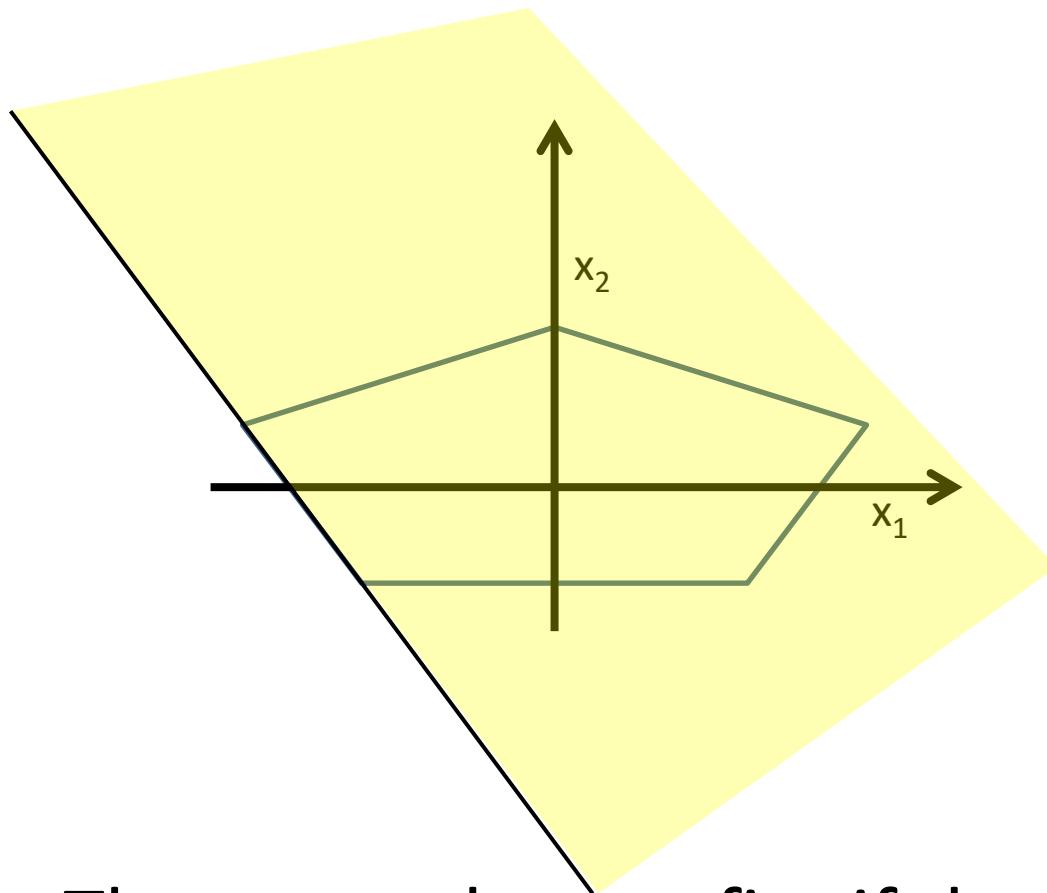
- The network must fire if the input is in the coloured area

Booleans over the reals



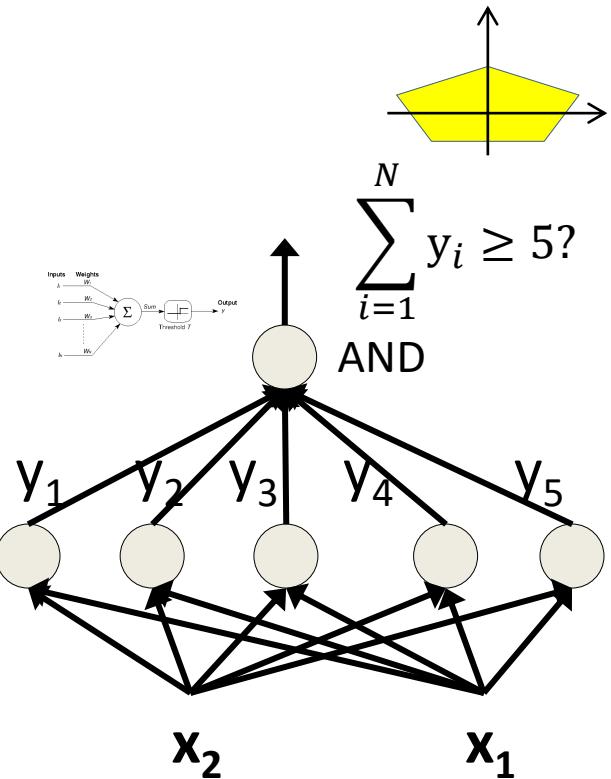
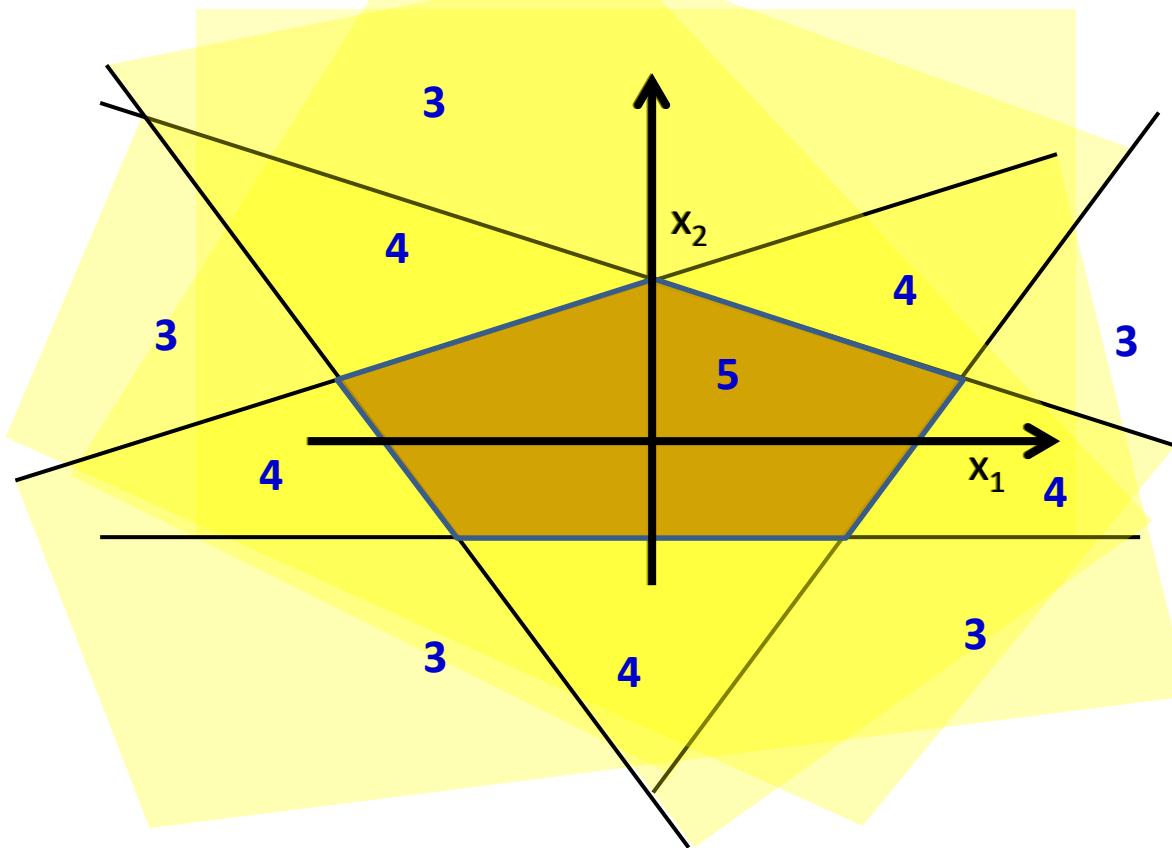
- The network must fire if the input is in the coloured area

Booleans over the reals



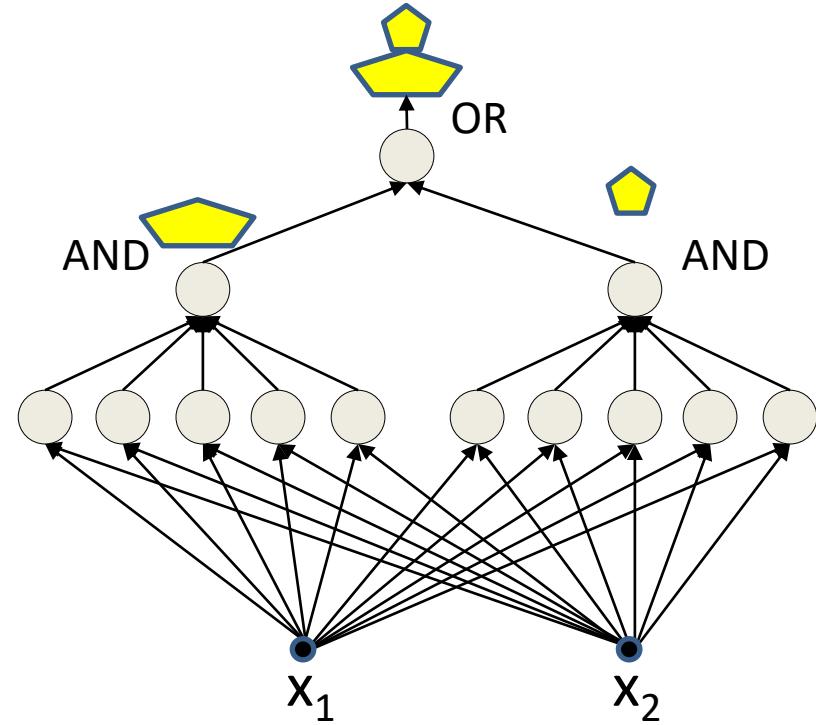
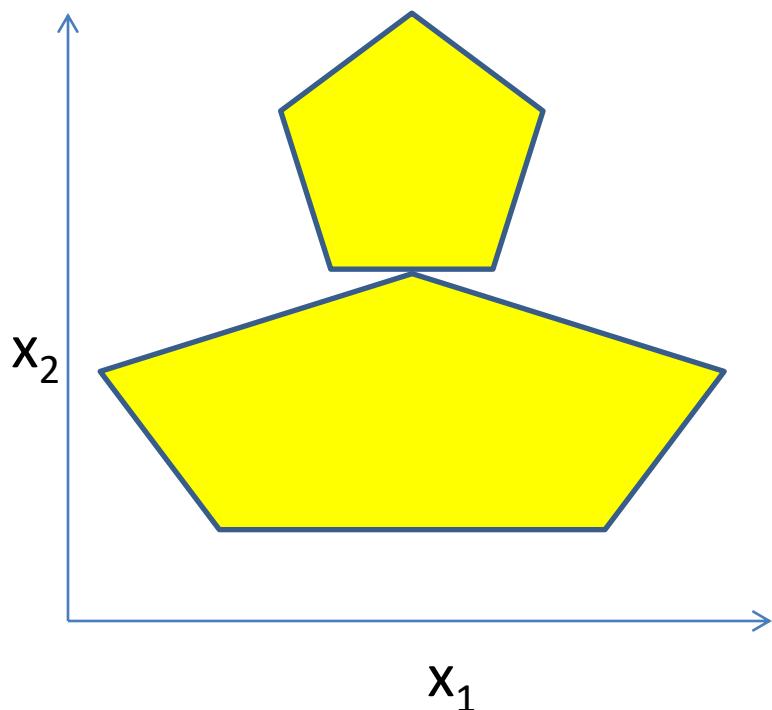
- The network must fire if the input is in the coloured area

Booleans over the reals



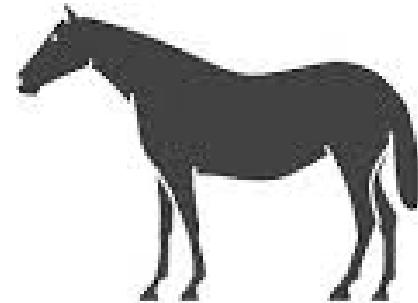
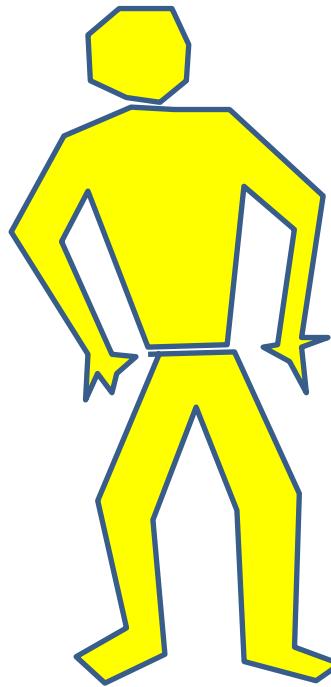
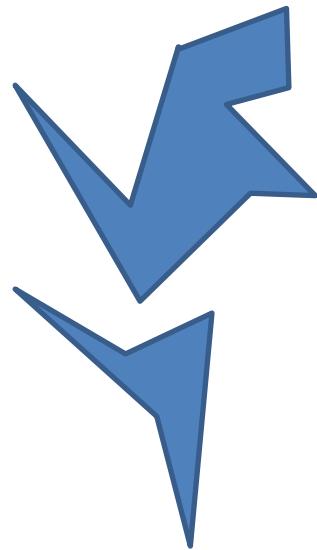
- The network must fire if the input is in the coloured area
 - The AND compares the sum of the hidden outputs to 5
 - NB: What would the pattern be if it compared it to 4?

More complex decision boundaries



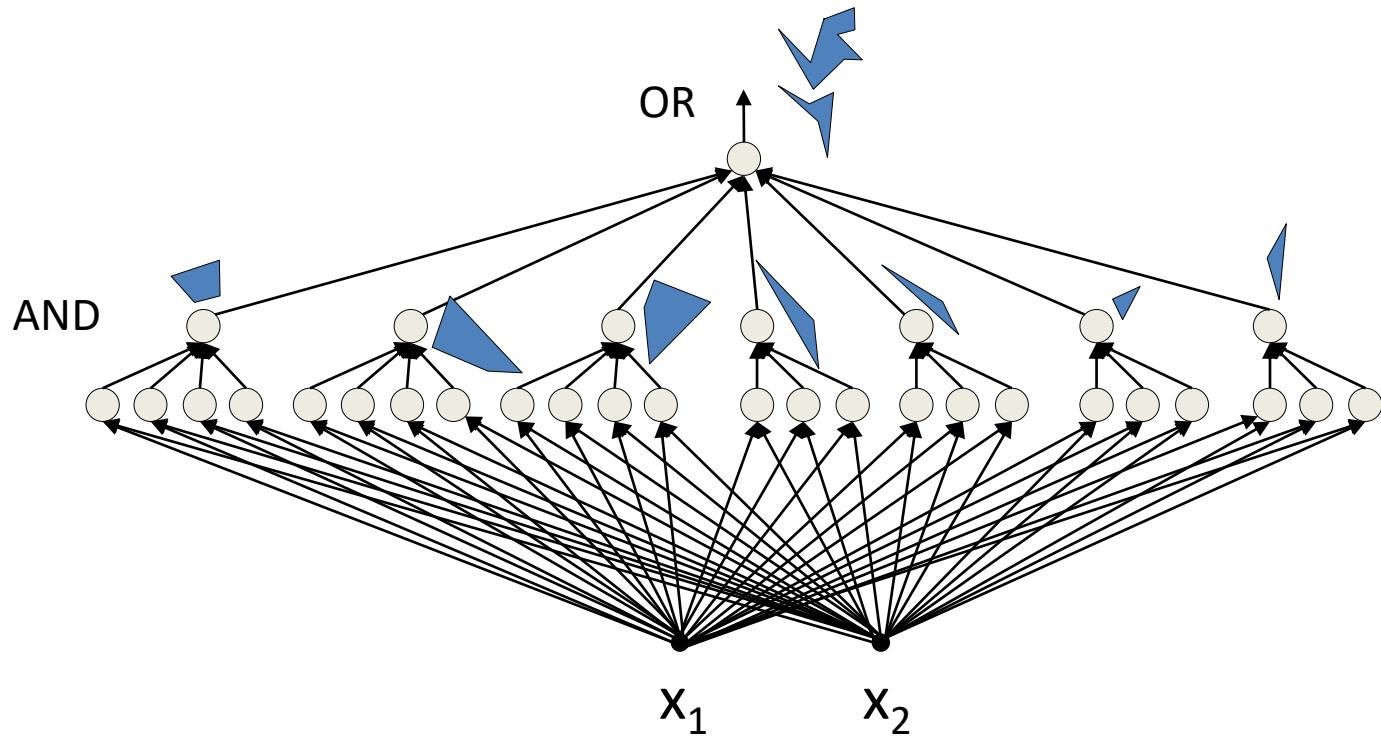
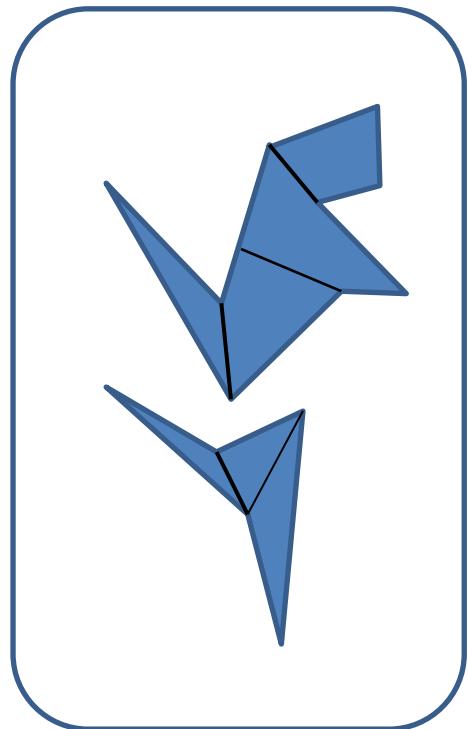
- Network to fire if the input is in the yellow area
 - “OR” two polygons
 - A third layer is required

Complex decision boundaries



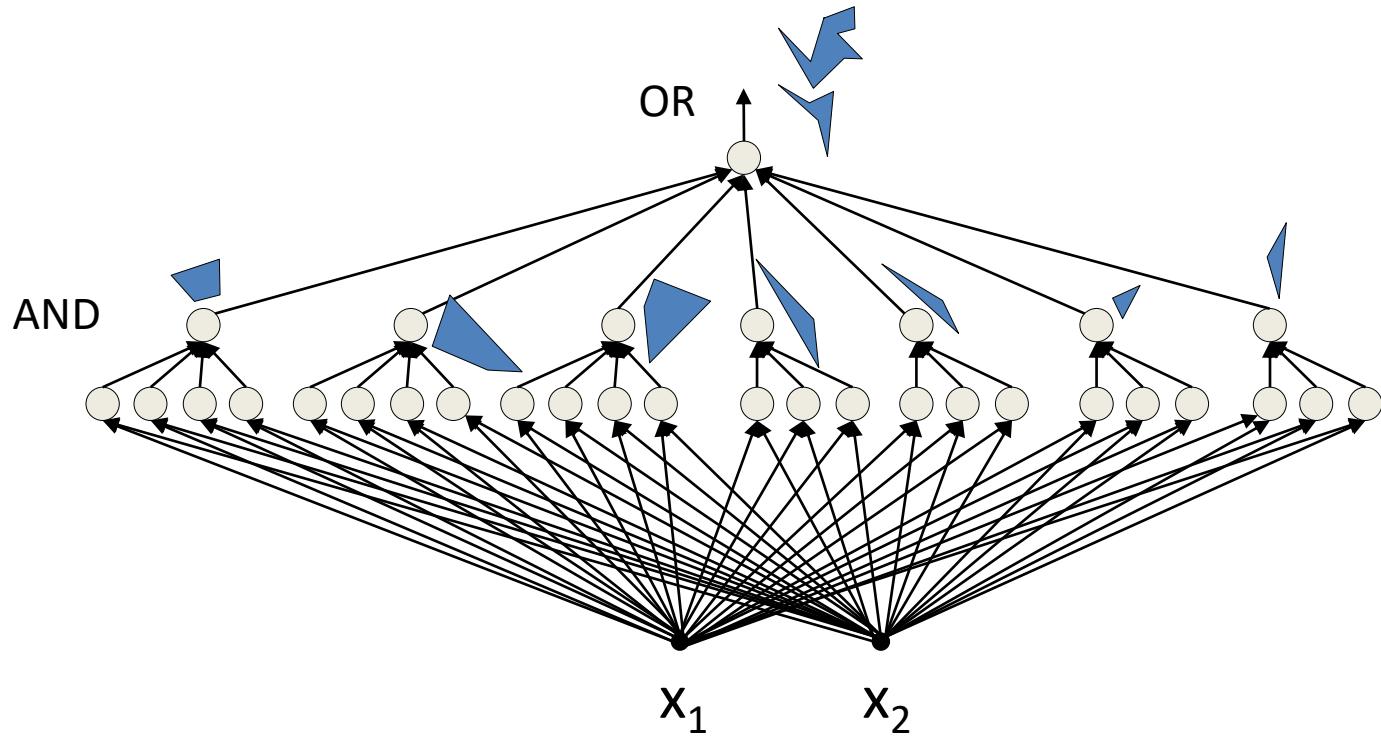
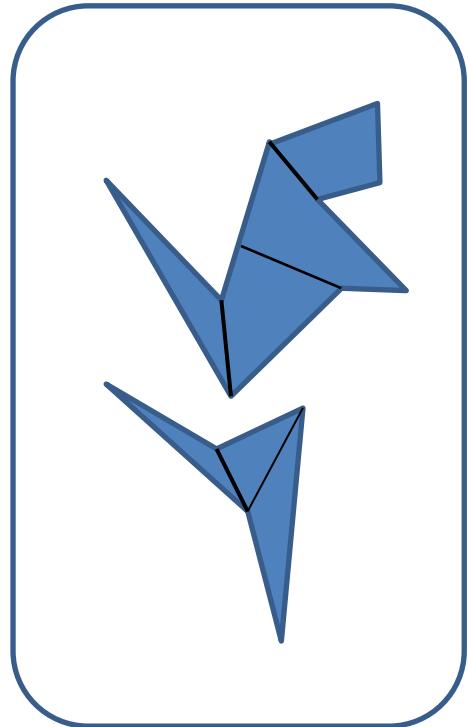
- Can compose *arbitrarily* complex decision boundaries

Complex decision boundaries



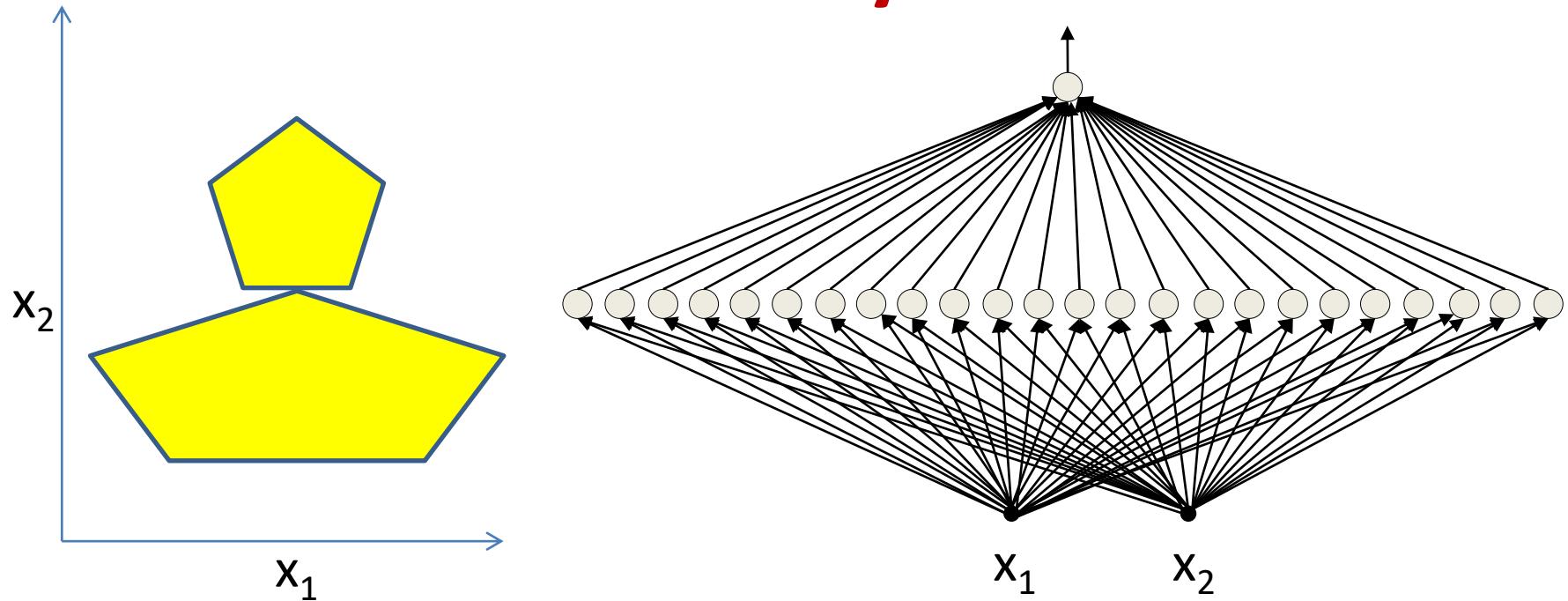
- Can compose *arbitrarily* complex decision boundaries

Complex decision boundaries



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

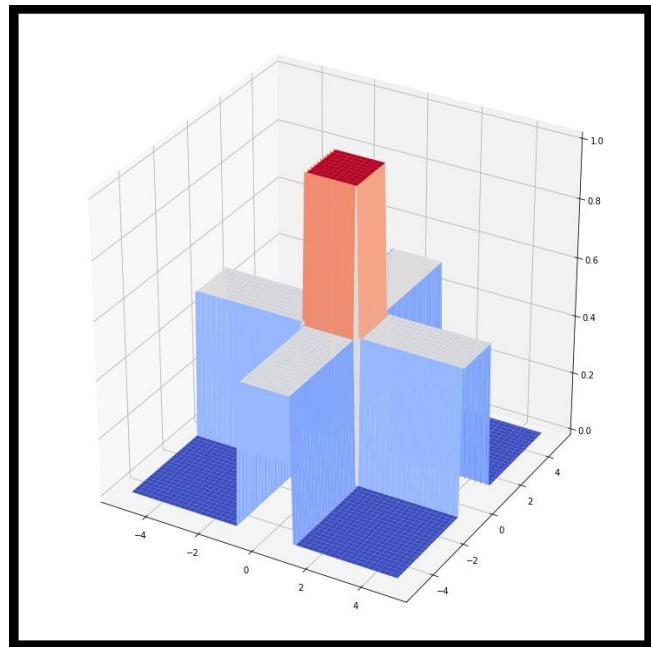
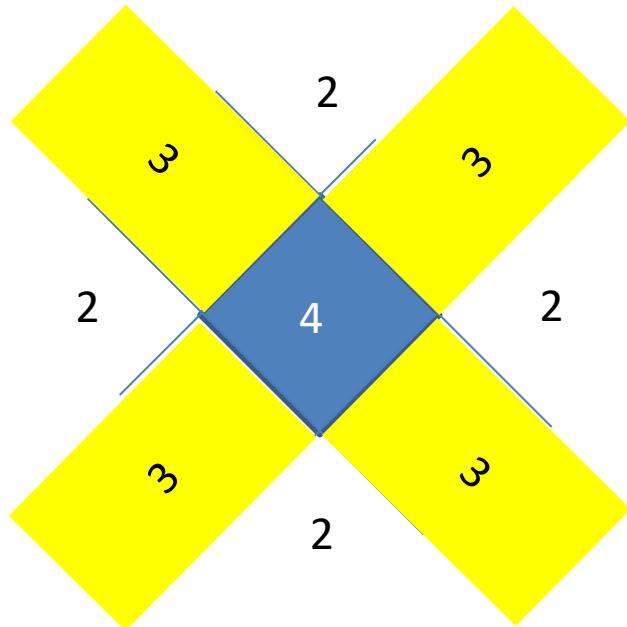
Exercise: compose this with one hidden layer



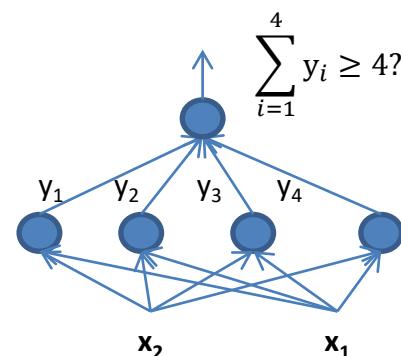
- How would you compose the decision boundary to the left with only one hidden layer?

$$(a \& b) \sqcup (c \& d) =$$

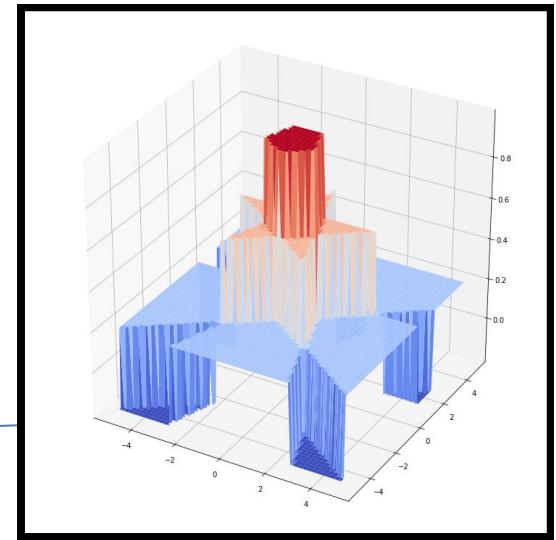
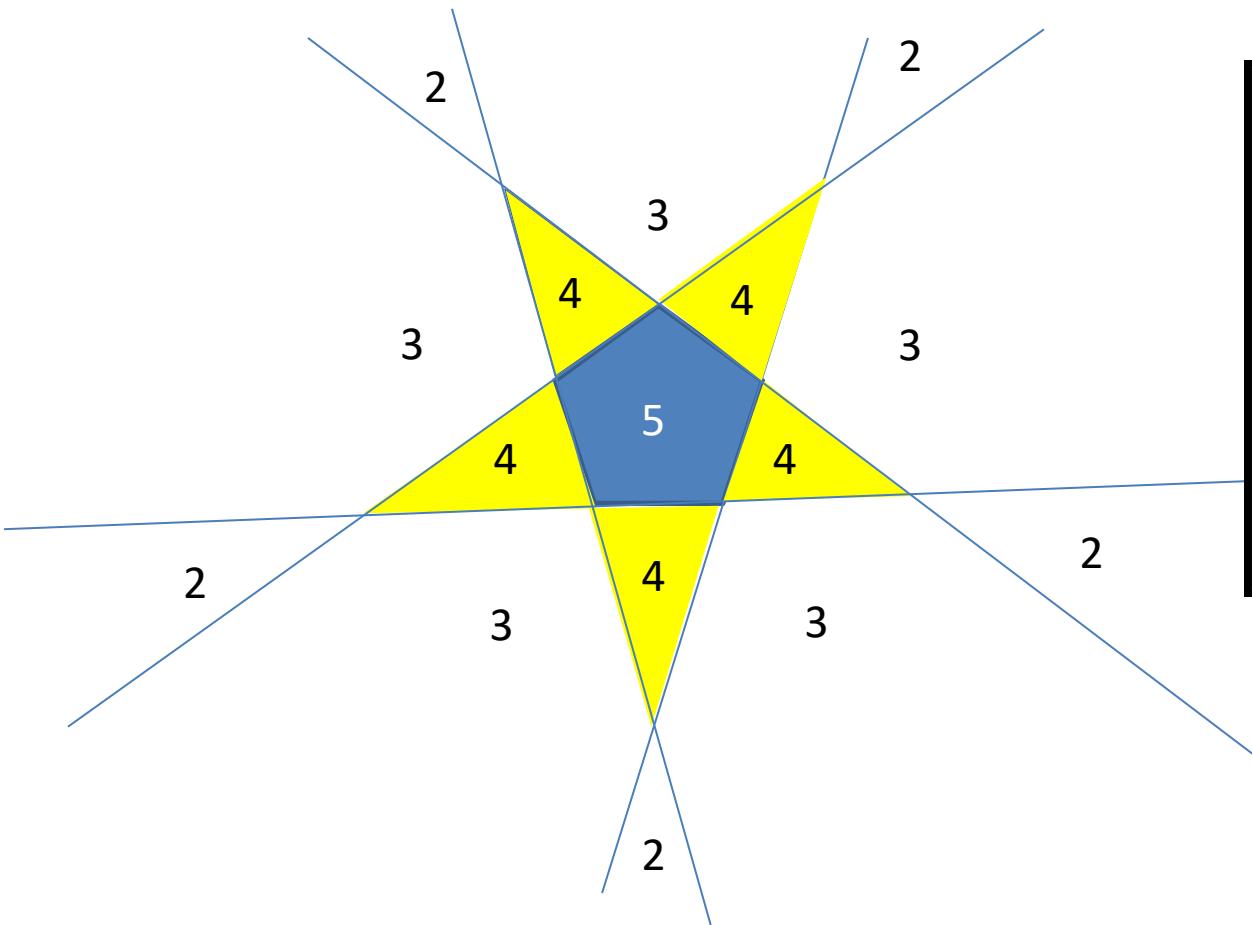
Composing a Square decision boundary



- The polygon net

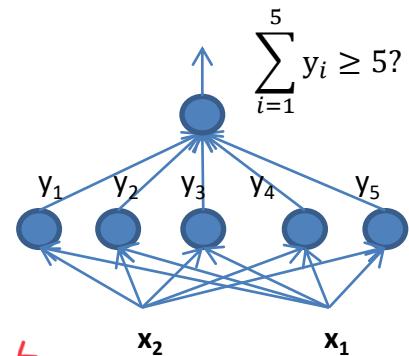


Composing a pentagon

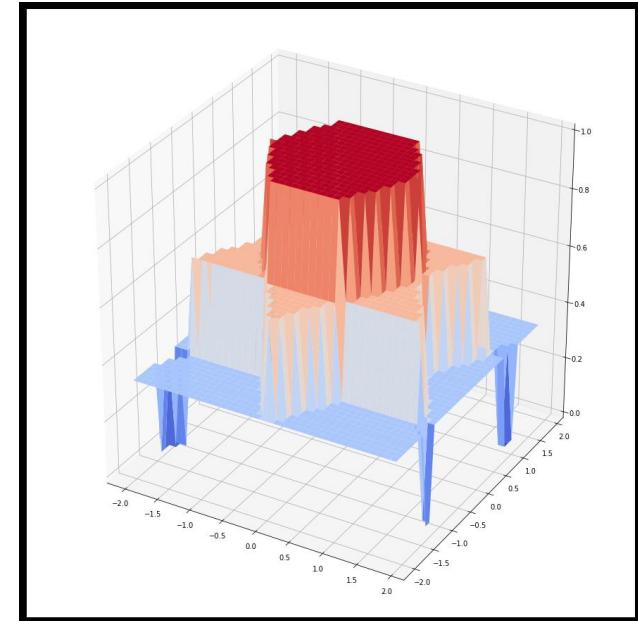
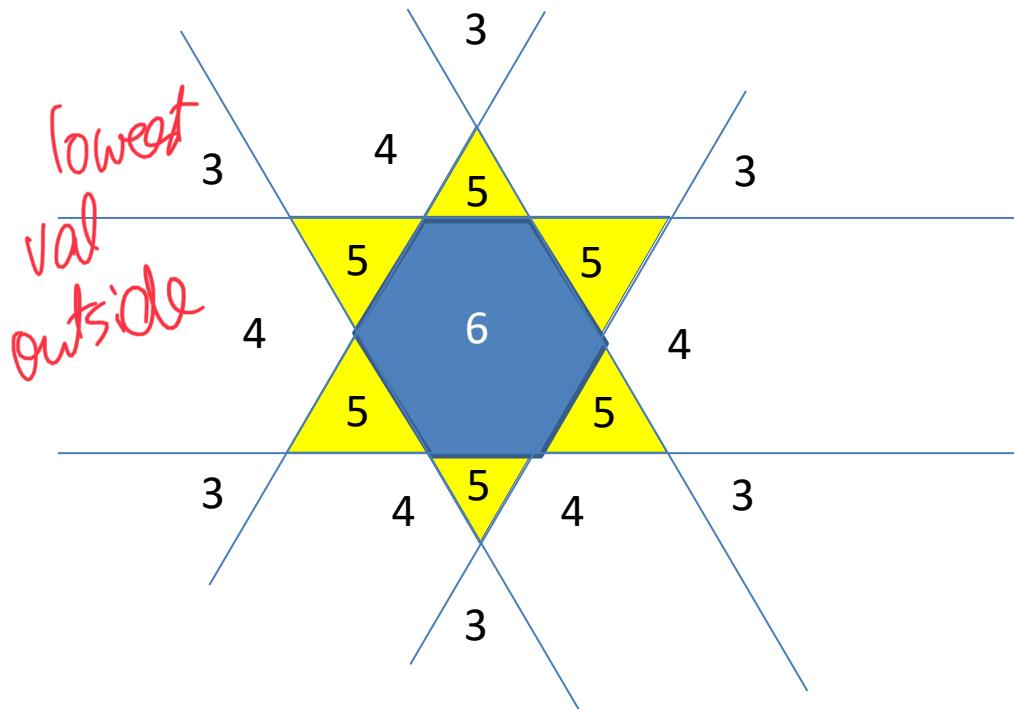


- The polygon net

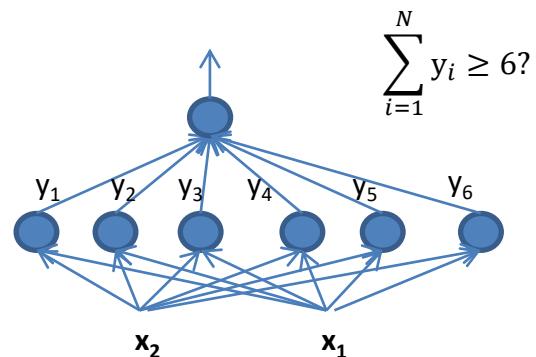
If you use $th=5 \Rightarrow$ pentagon ; $th=4 \Rightarrow$ star
 $; th=3 \Rightarrow$ infinite



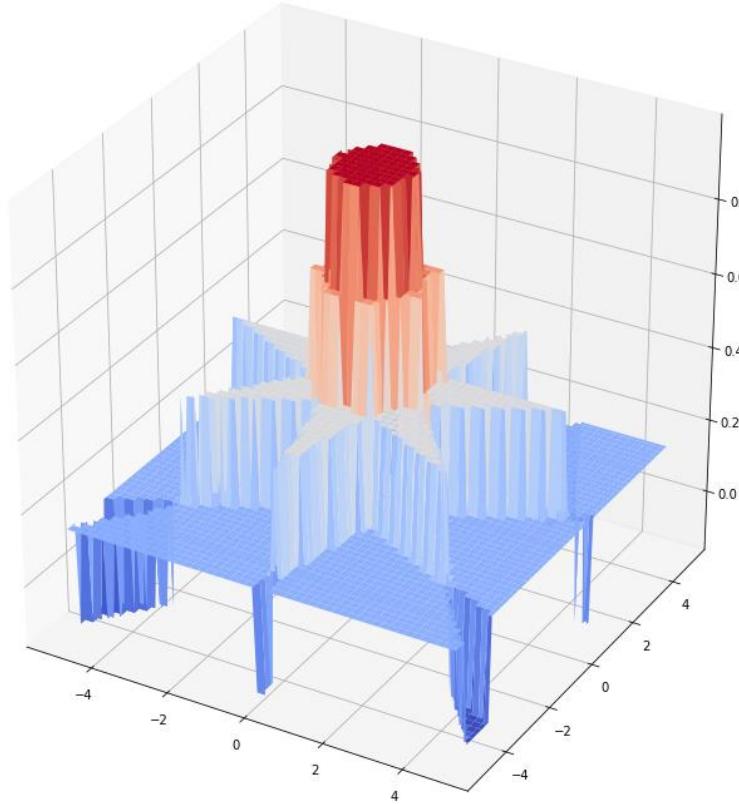
Composing a hexagon



- The polygon net

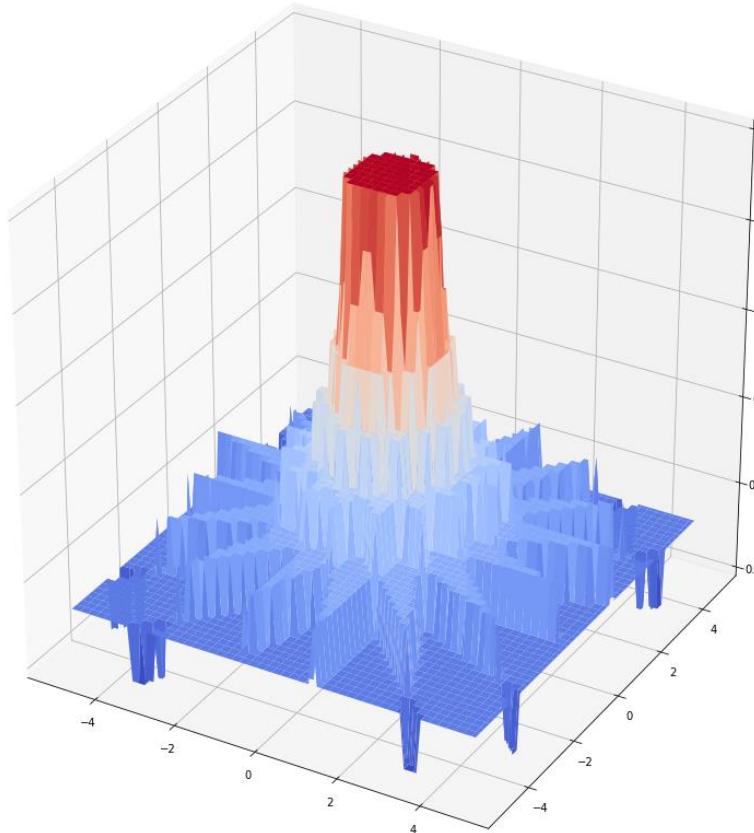


How about a heptagon



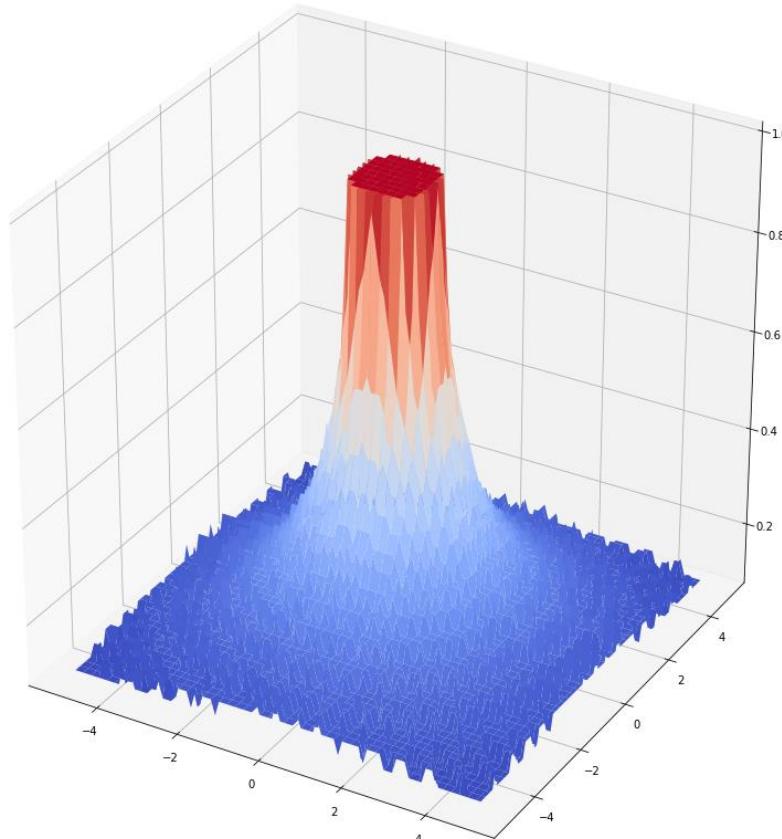
- What are the sums in the different regions?
 - A pattern emerges as we consider $N > 6..$
 - N is the number of sides of the polygon

16 sides



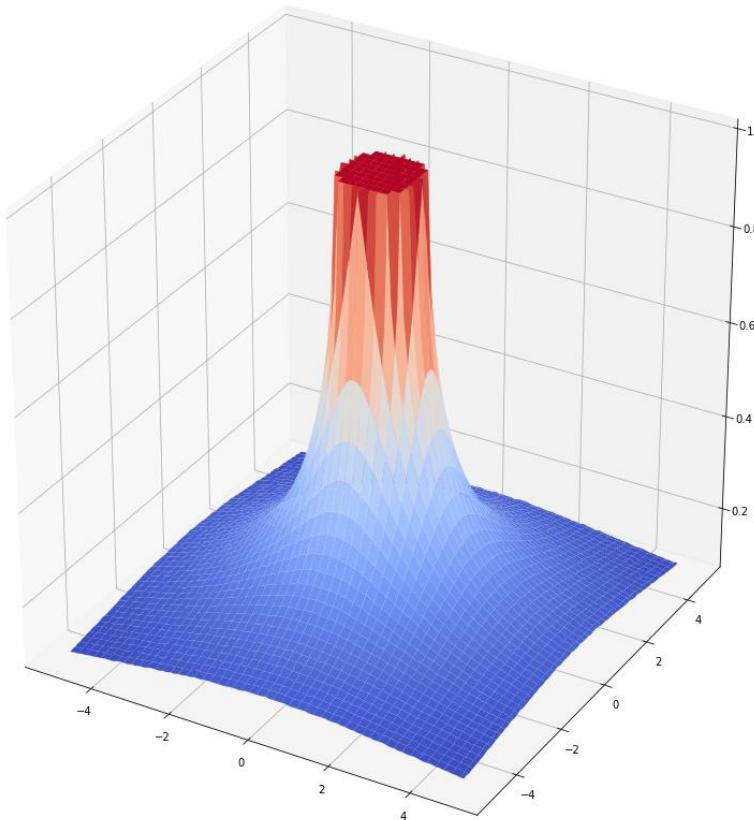
- What are the sums in the different regions?
 - A pattern emerges as we consider $N > 6..$

64 sides



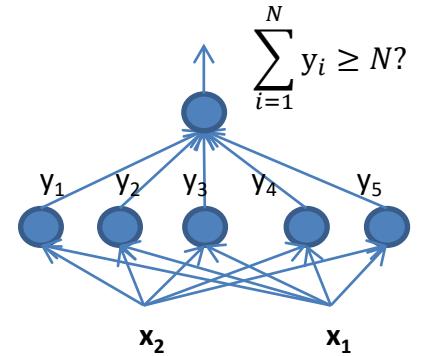
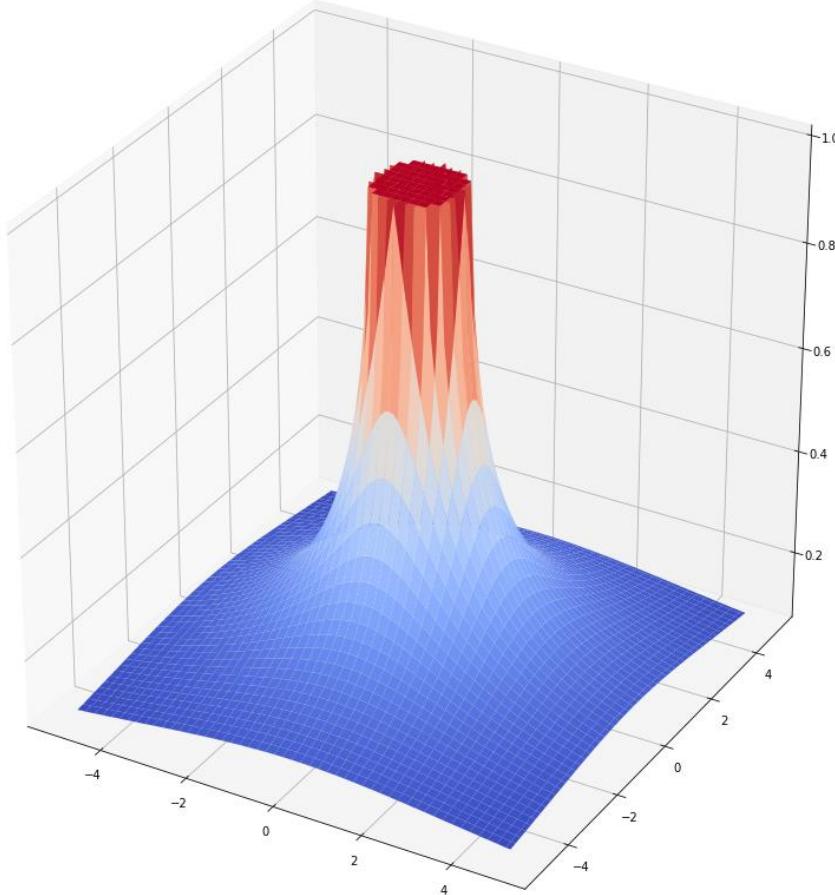
- What are the sums in the different regions?
 - A pattern emerges as we consider $N > 6..$

1000 sides



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

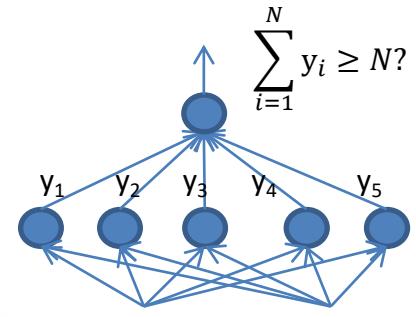
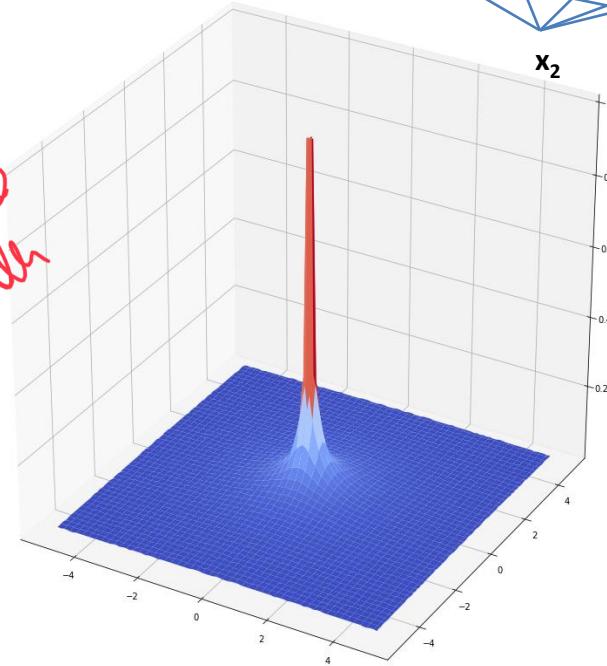
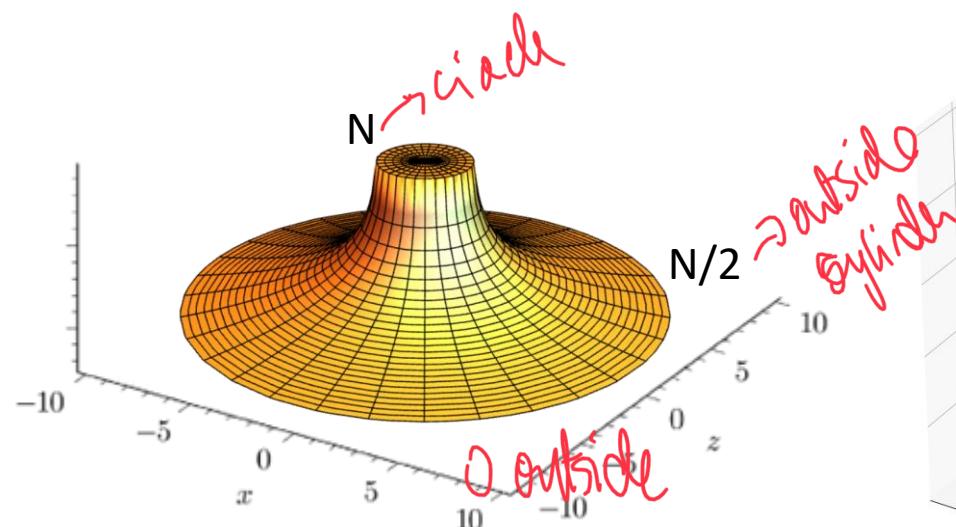
Polygon net



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

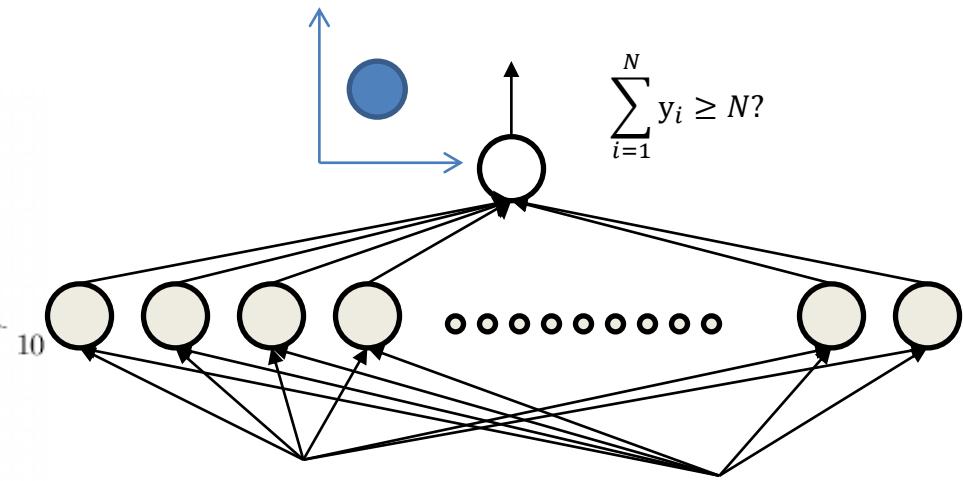
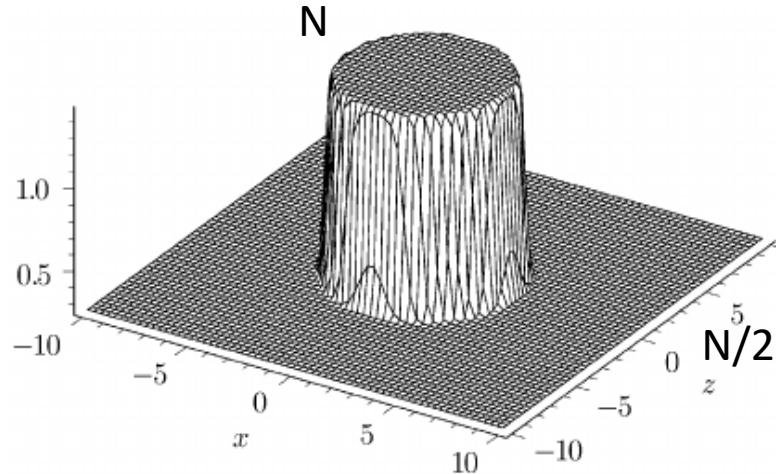
In the limit

If you keep adding



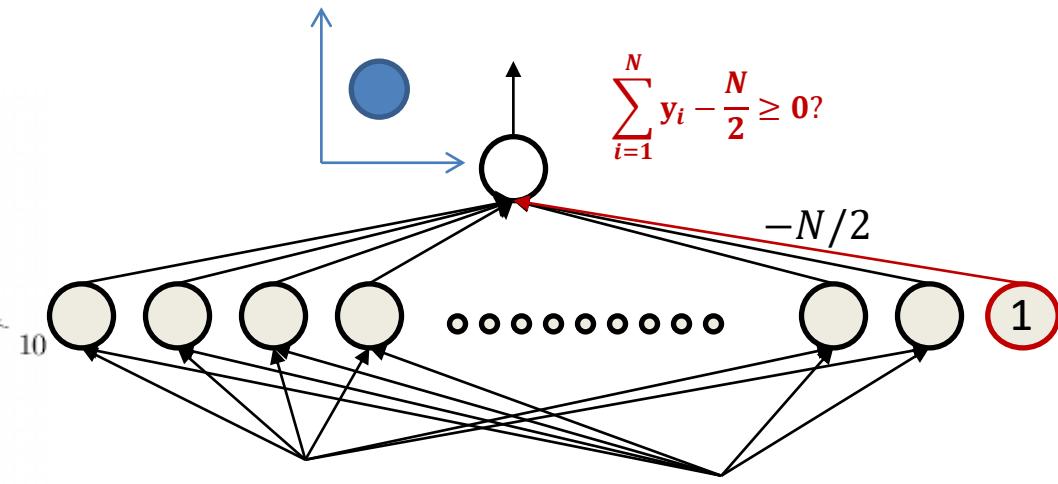
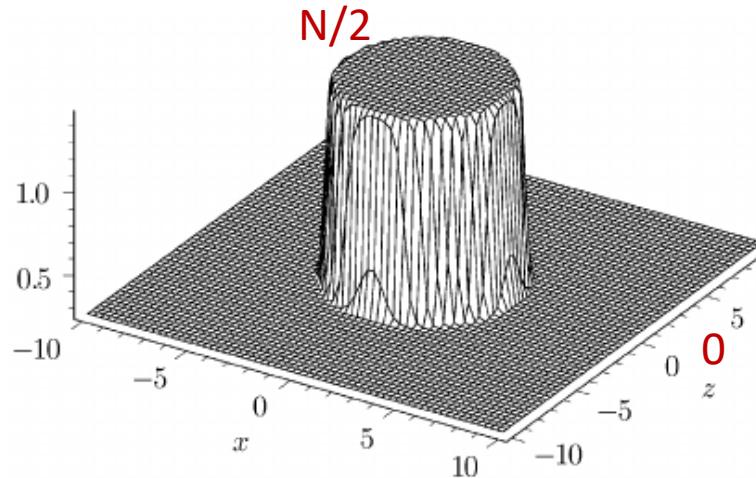
- $\sum_i y_i = N \left(1 - \frac{1}{\pi} \arccos \left(\min \left(1, \frac{\text{radius}}{|\mathbf{x}-\text{center}|} \right) \right) \right)$
 - Value of the sum at the output unit, as a function of distance from center, as N increases
- For small radius, it's a near perfect cylinder
 - N in the cylinder,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

Composing a circle



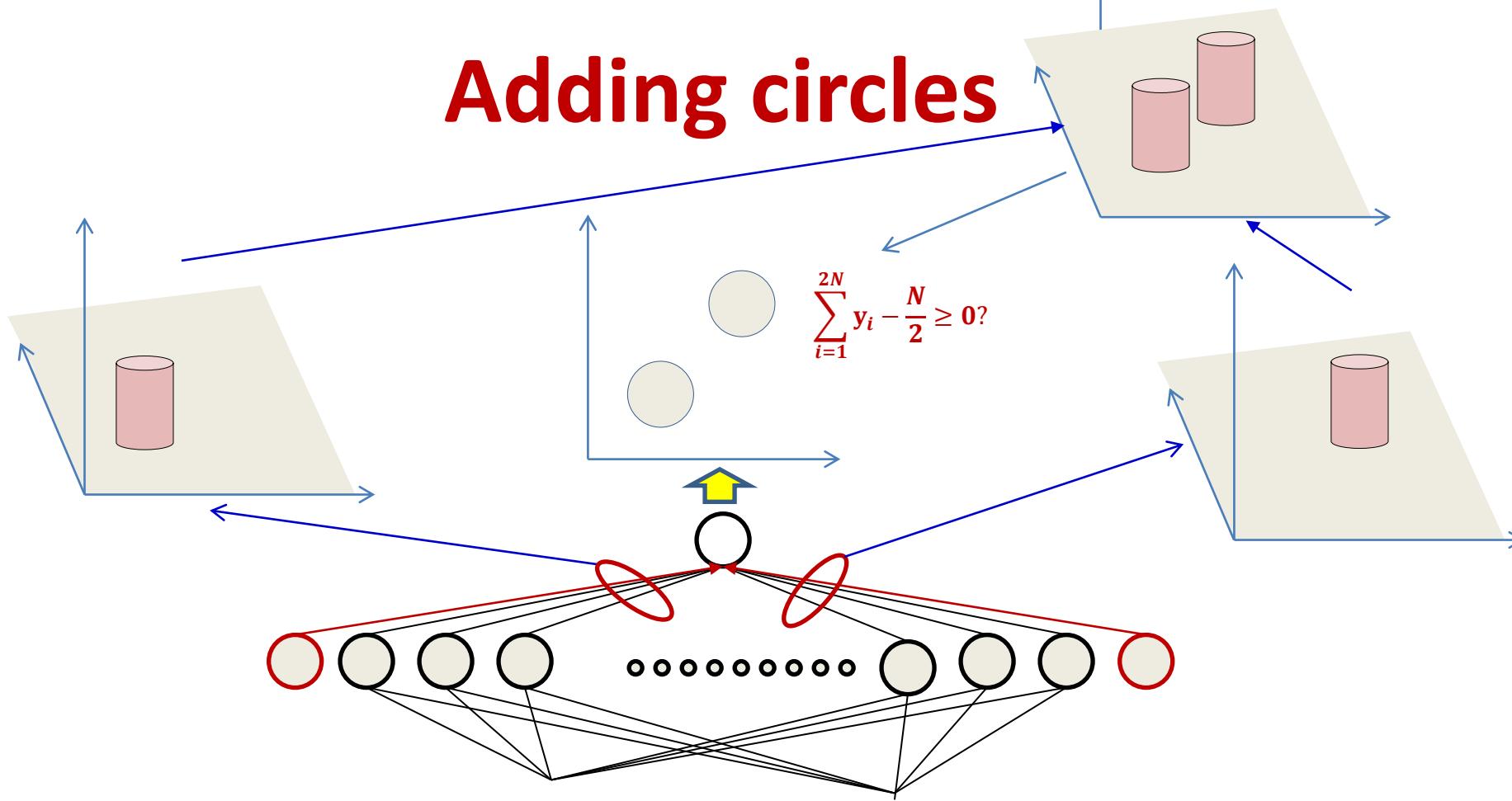
- The circle net
 - Very large number of neurons
 - *Sum is N inside the circle, $N/2$ outside almost everywhere*
 - Circle can be at any location

Composing a circle



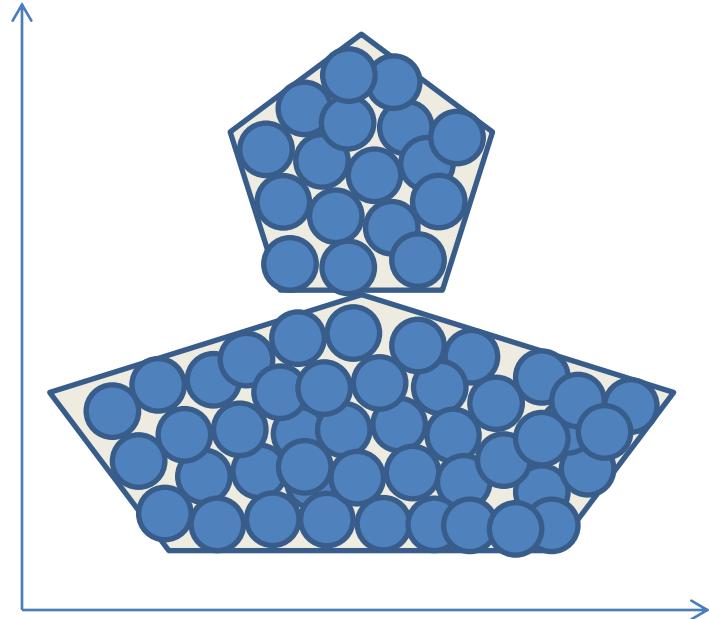
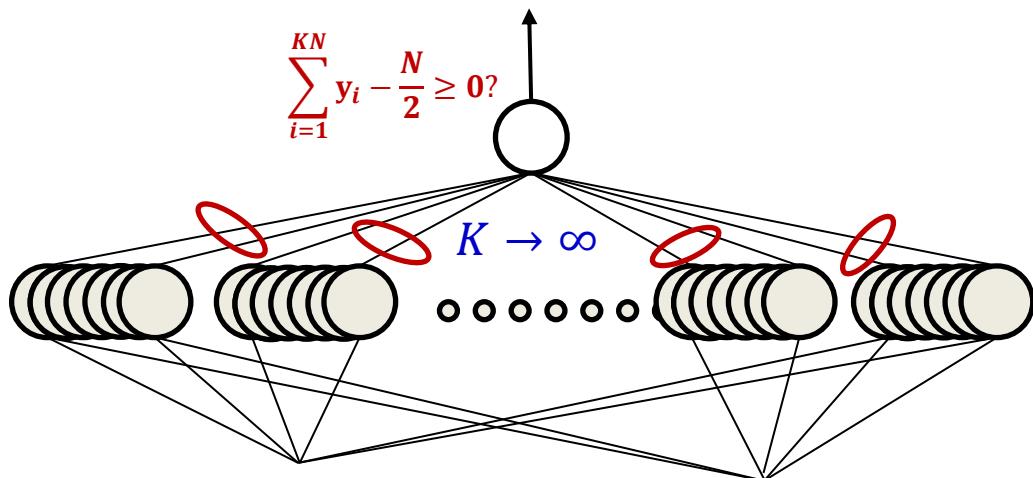
- The circle net
 - Very large number of neurons
 - *Sum is $N/2$ inside the circle, 0 outside almost everywhere*
 - Circle can be at any location

Adding circles



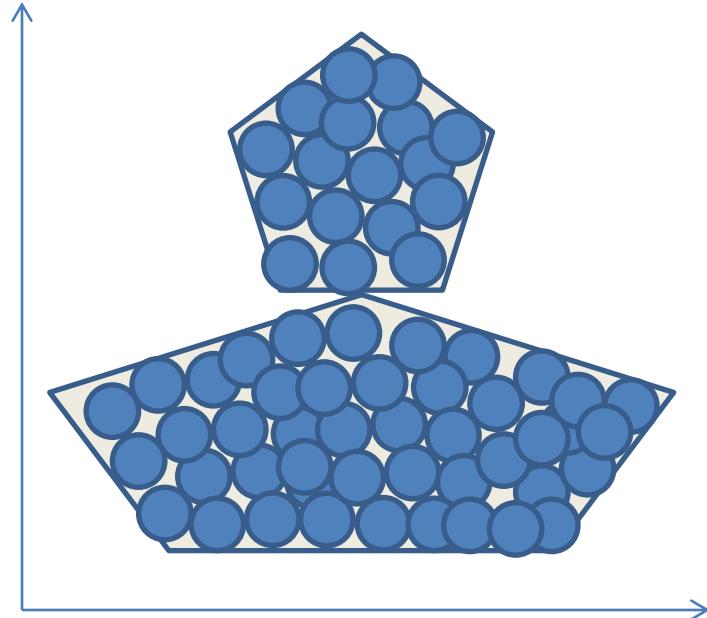
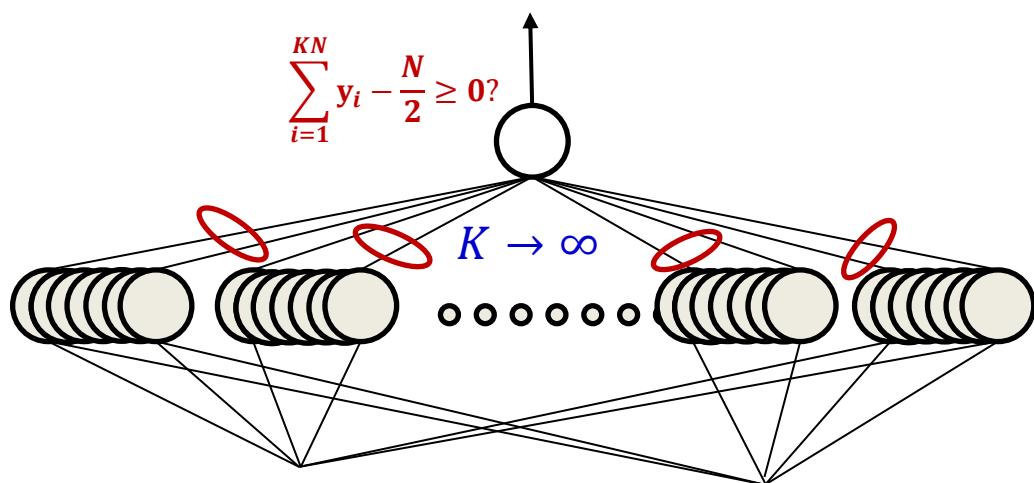
- The “sum” of two circles sub nets is exactly $N/2$ inside either circle, and 0 almost everywhere outside

Composing an arbitrary figure



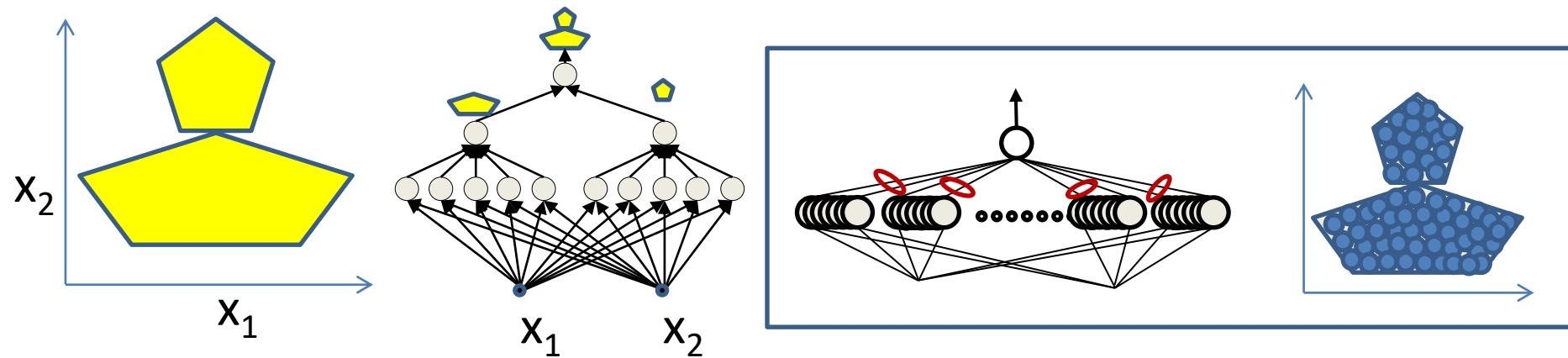
- Just fit in an arbitrary number of circles
 - More accurate approximation with greater number of smaller circles
 - Can achieve arbitrary precision

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



Adding depth makes a diff!

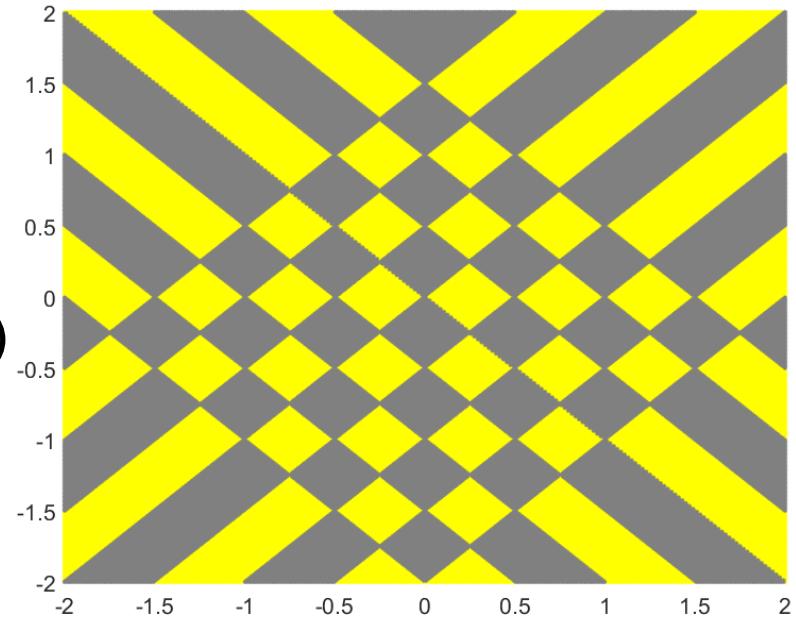
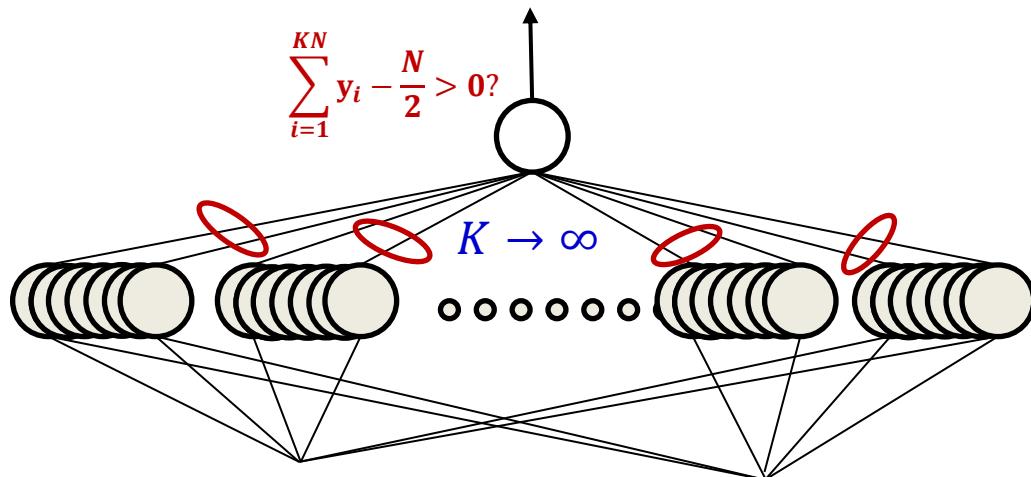
- Deeper networks can require far fewer neurons
 - 12 vs. ~infinite hidden neurons in this example

Optimal depth in *generic* nets

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

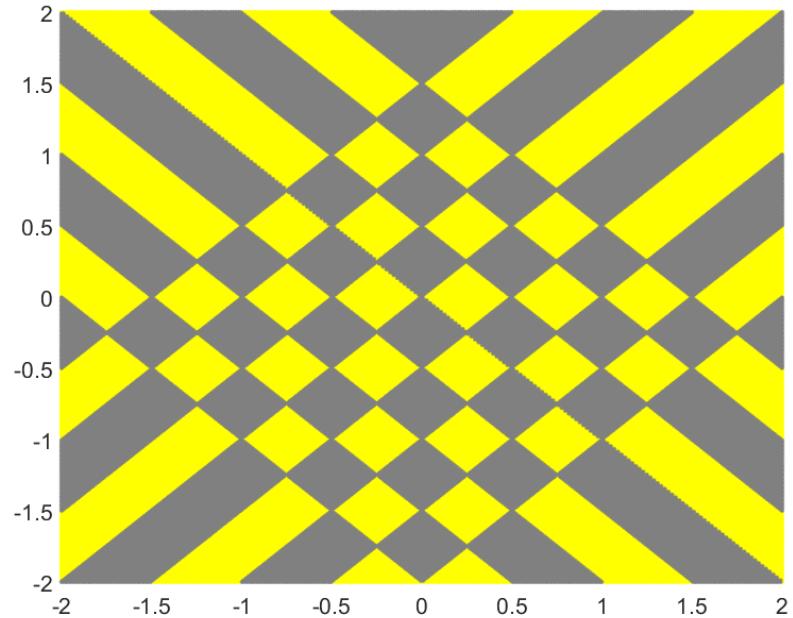
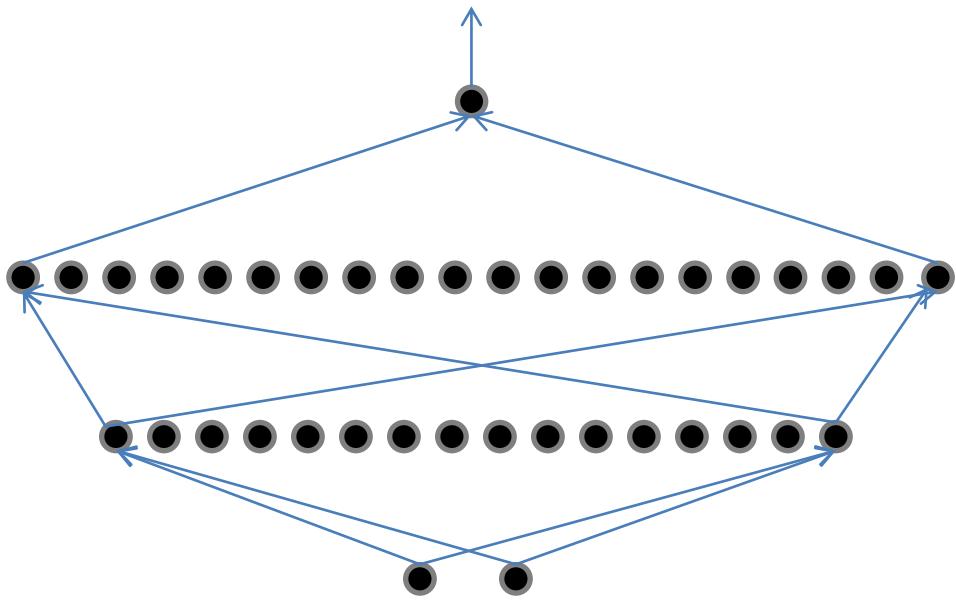
Optimal depth

Yellow Outside



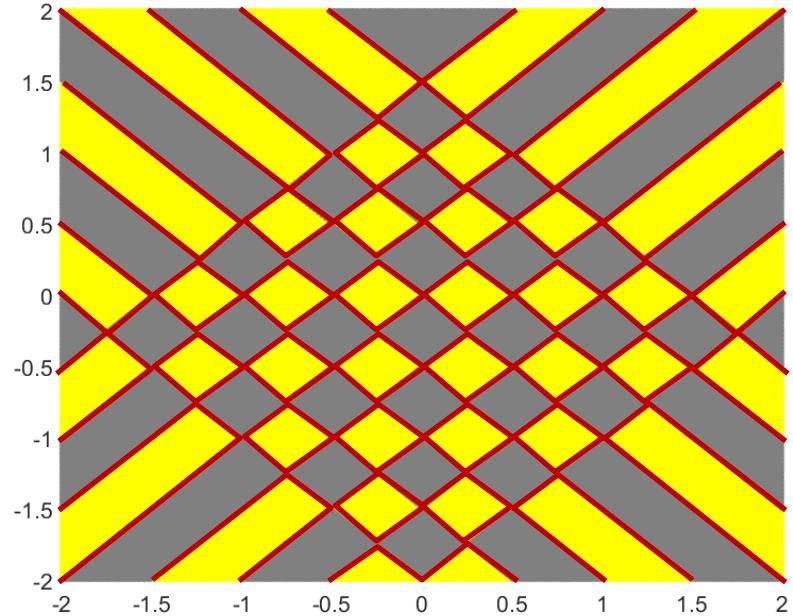
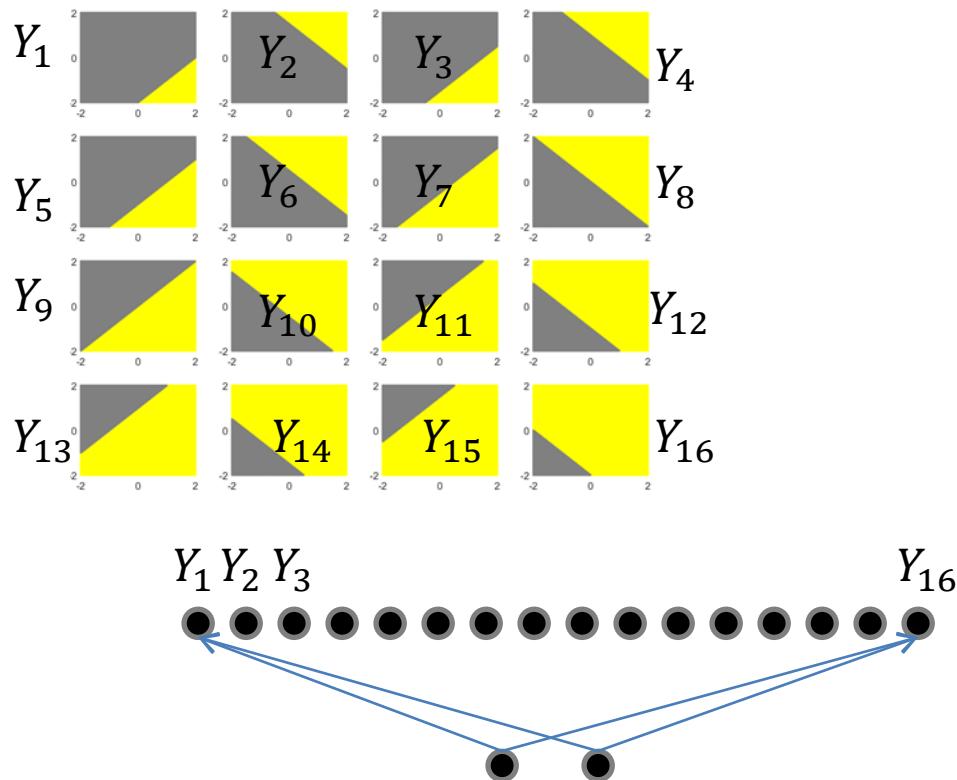
- A naïve one-hidden-layer neural network will require infinite hidden neurons

Optimal depth



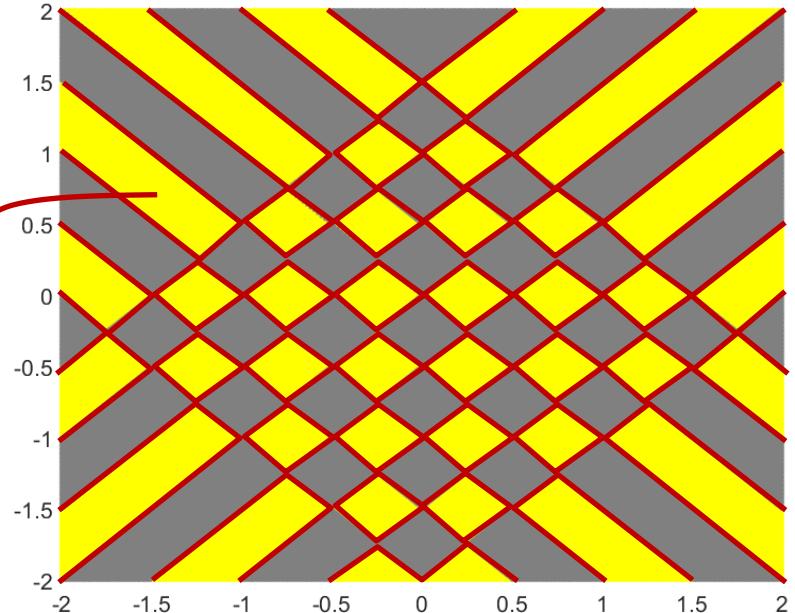
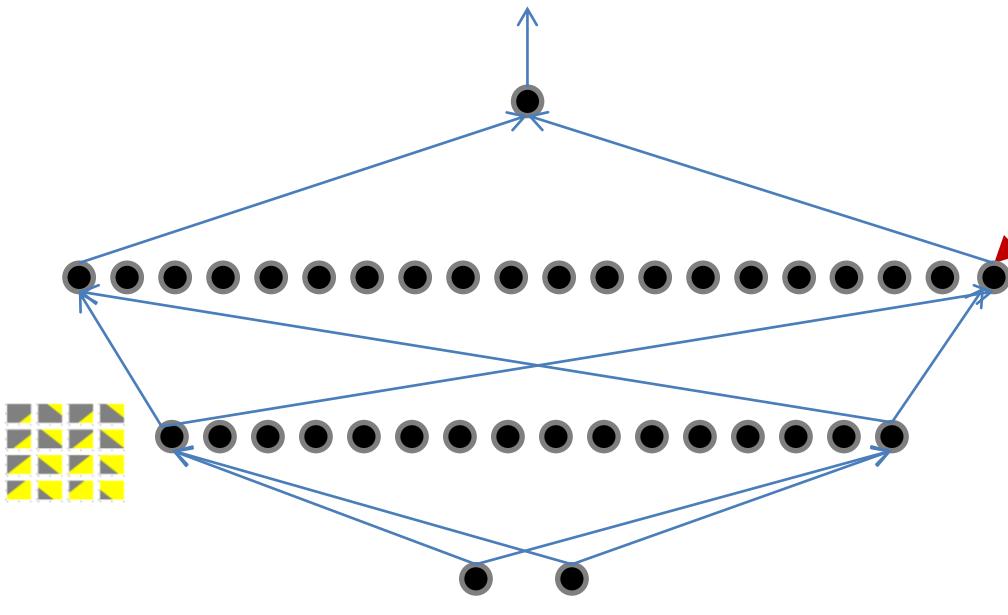
- Two hidden-layer network: 56 hidden neurons

Optimal depth



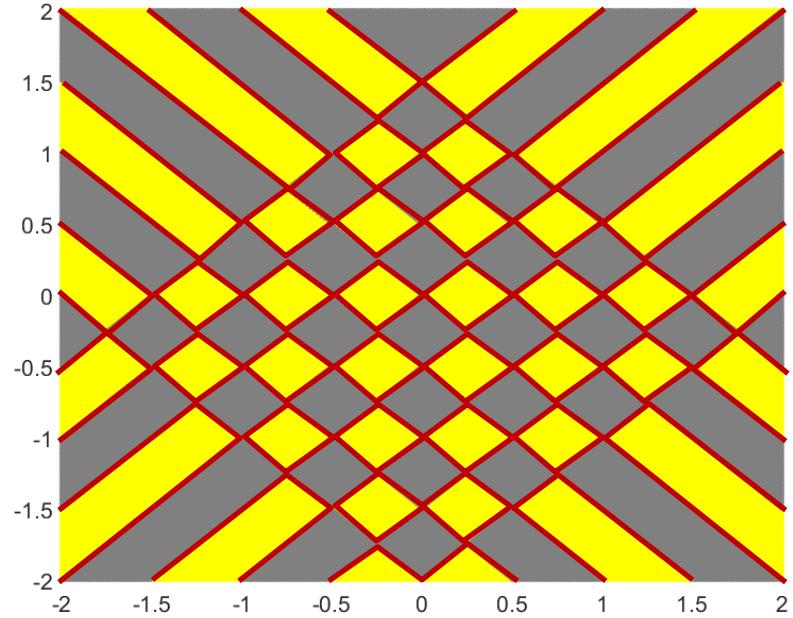
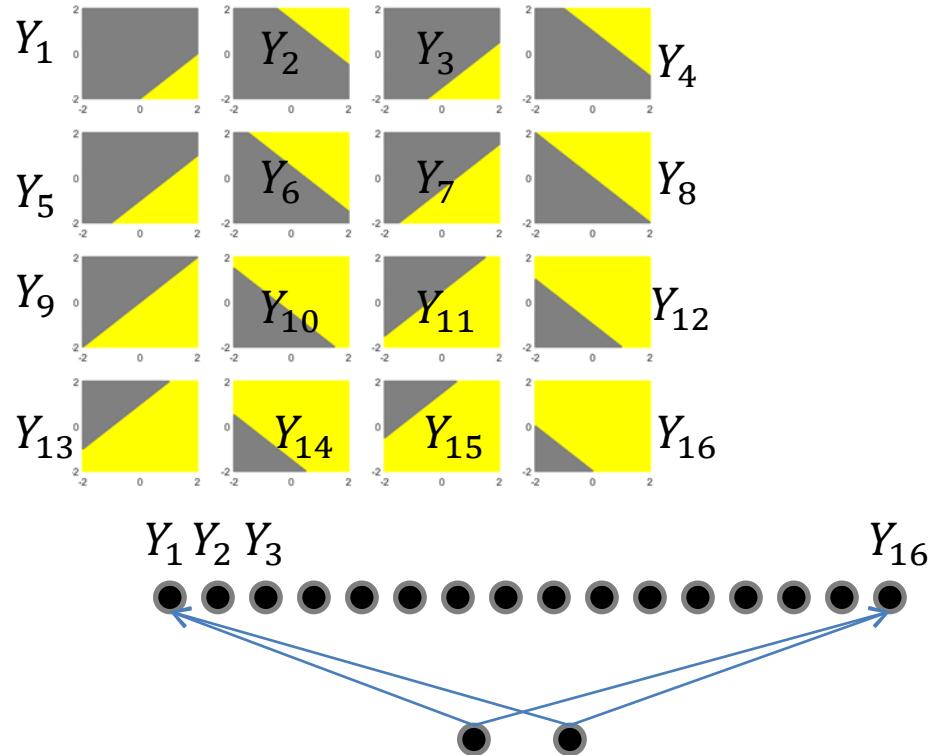
- Two-hidden-layer network: 56 hidden neurons
 - 16 neurons in hidden layer 1

Optimal depth



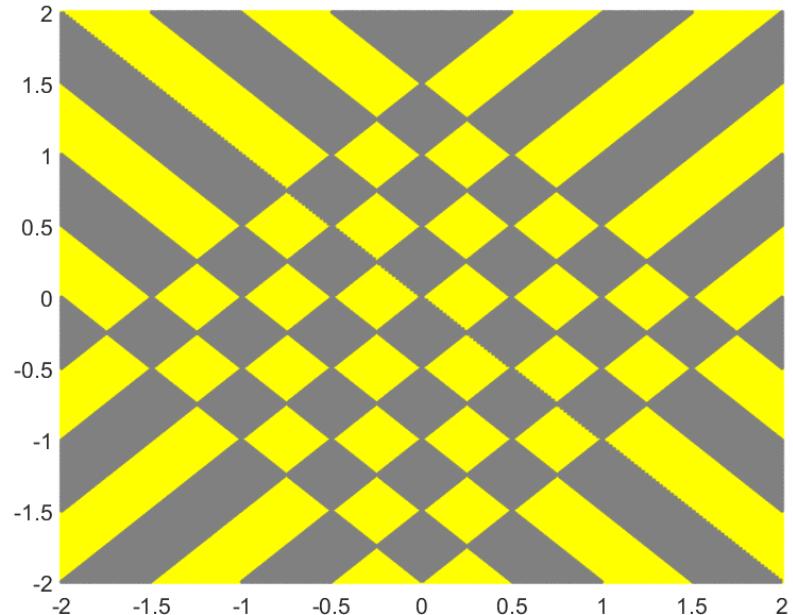
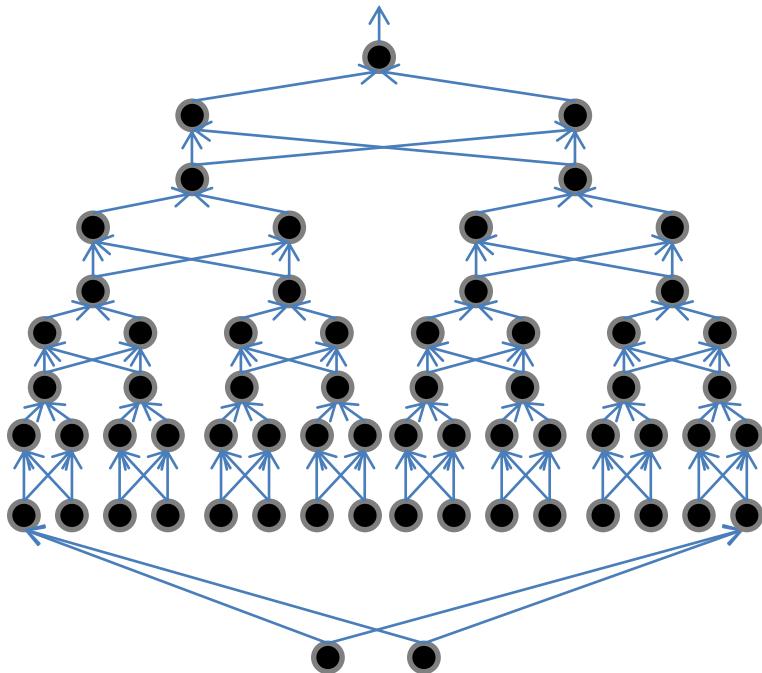
- Two-hidden-layer network: 56 hidden neurons
 - 16 in hidden layer 1
 - 40 in hidden layer 2
 - 57 total neurons, including output neuron

Optimal depth



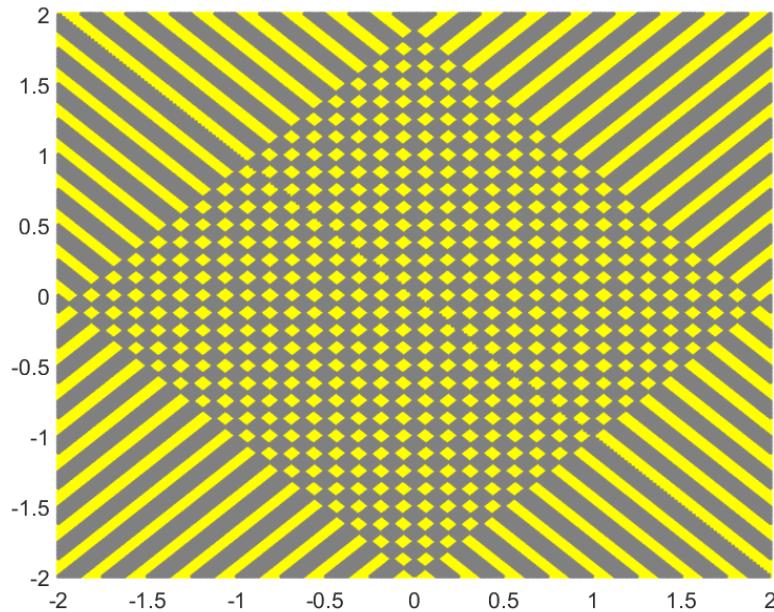
- But this is just $Y_1 \oplus Y_2 \oplus \dots \oplus Y_{16}$

Optimal depth



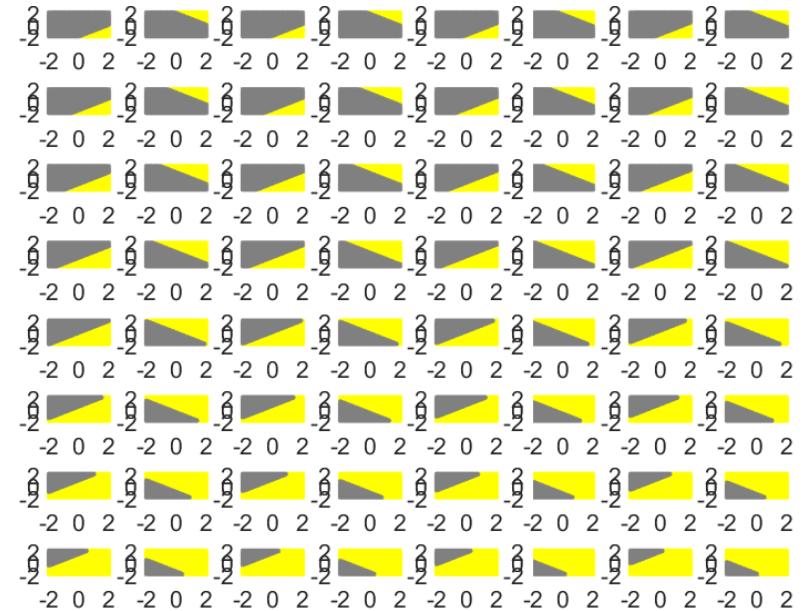
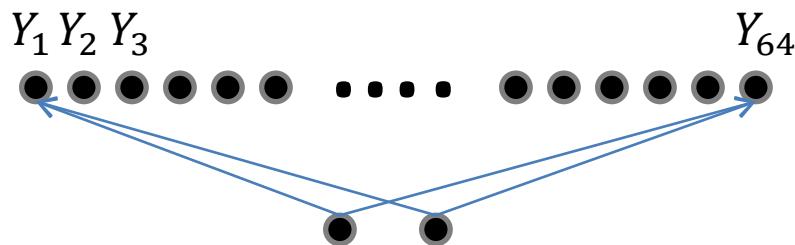
- But this is just $Y_1 \oplus Y_2 \oplus \dots \oplus Y_{16}$
 - The XOR net will require $16 + 15 \times 3 = 61$ neurons
 - 46 neurons if we use a two-neuron XOR model

Optimal depth



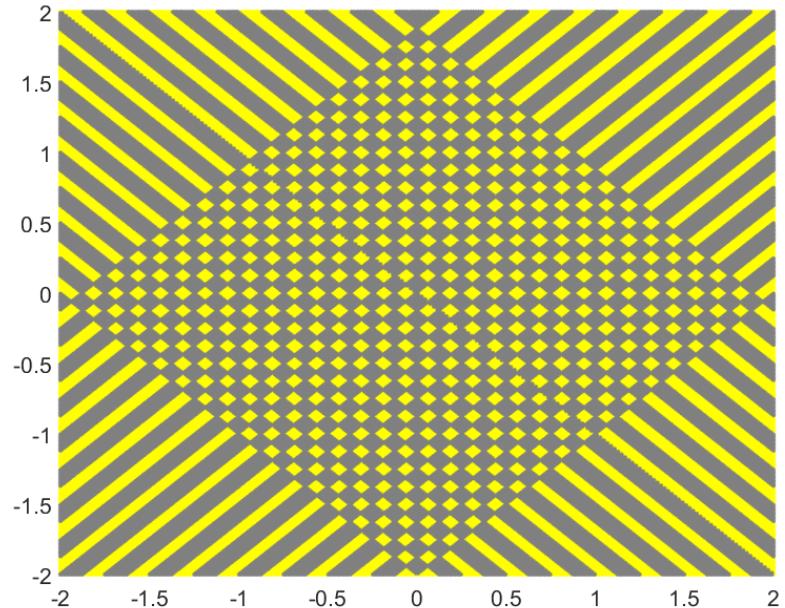
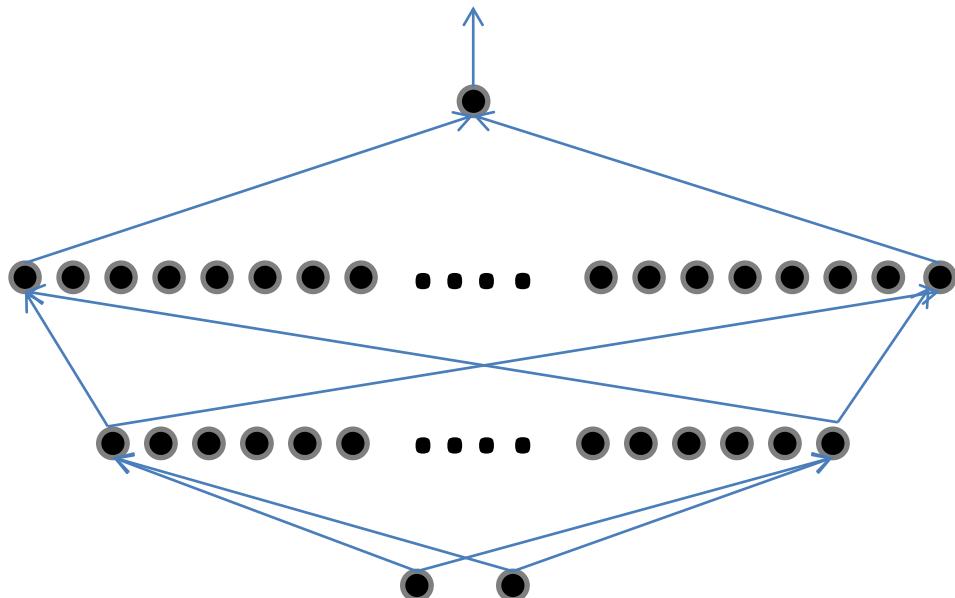
- Grid formed from 64 lines
 - Network must output 1 for inputs in the yellow regions

Actual linear units



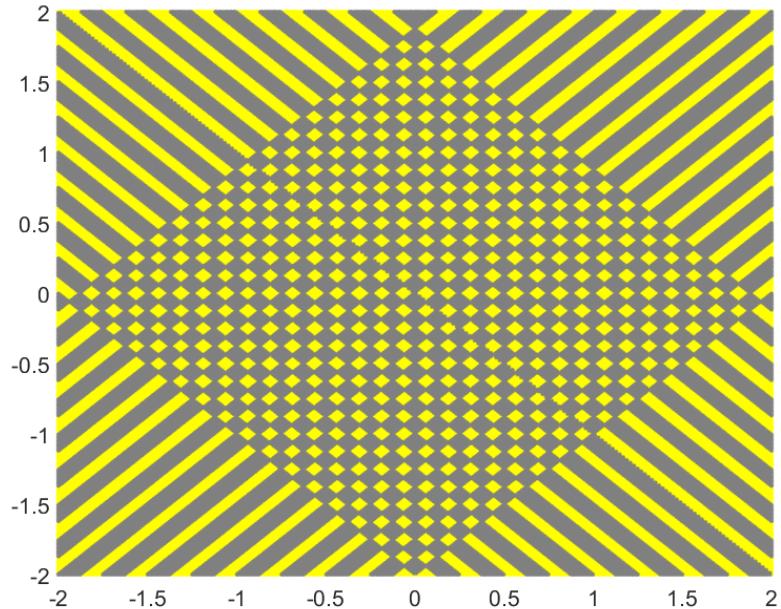
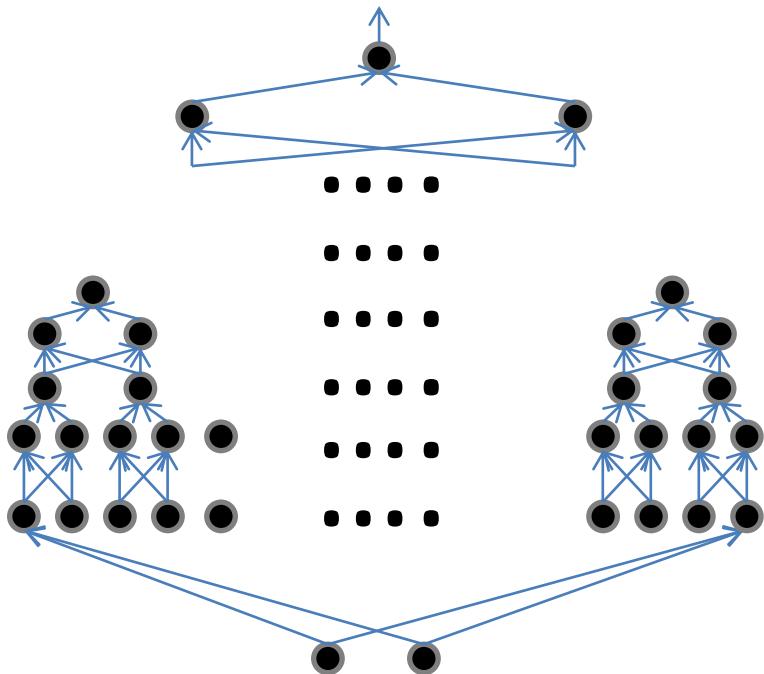
- 64 basic linear feature detectors

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

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

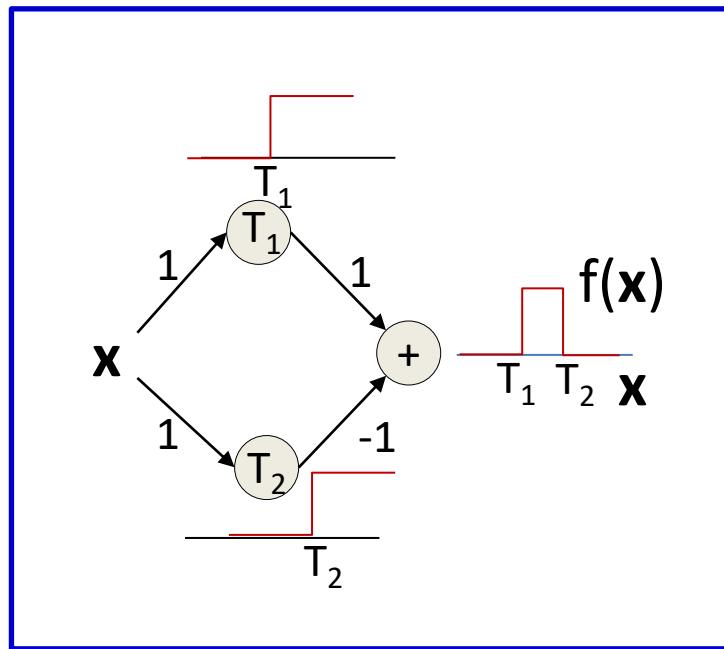
Story so far

- 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

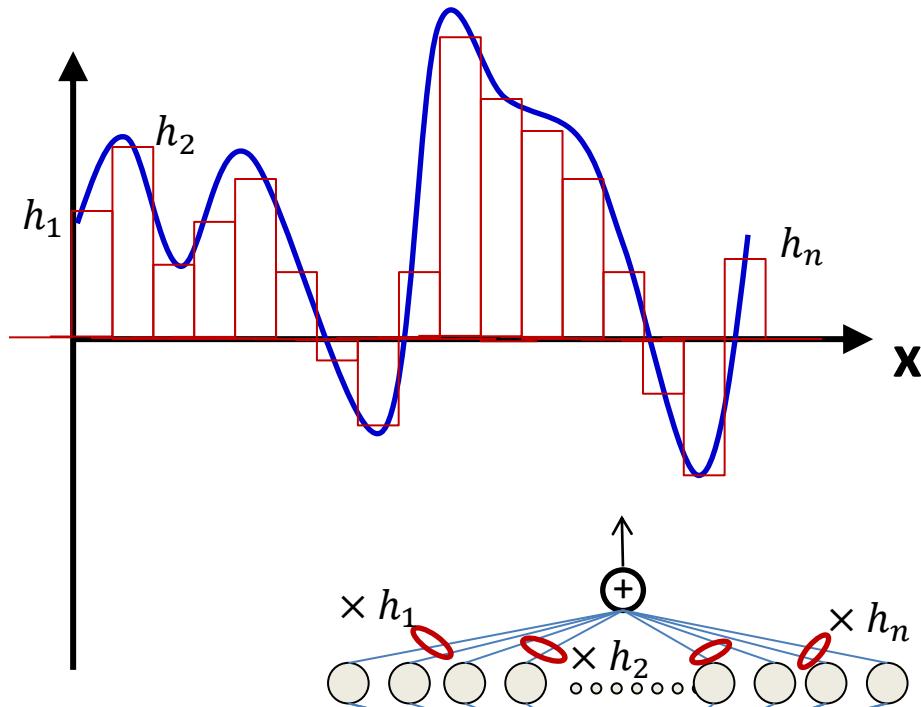
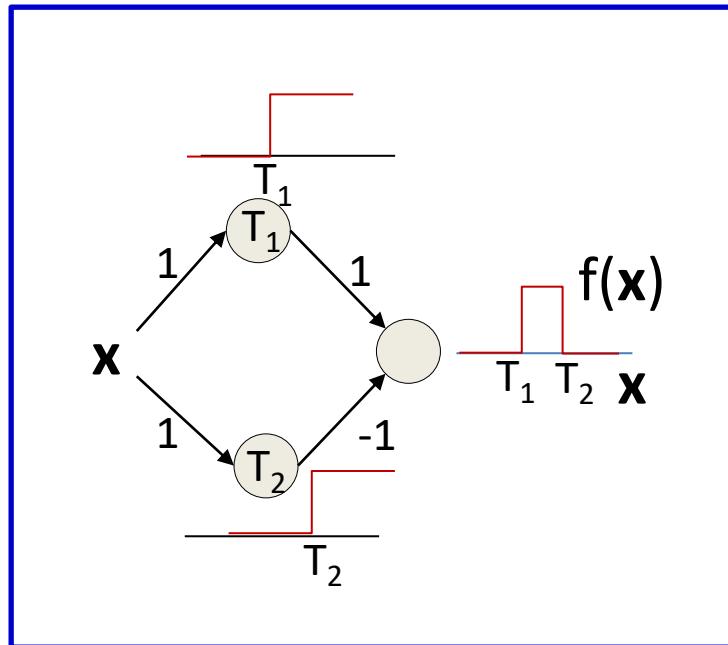
- Multi-layer Perceptrons as universal Boolean functions
 - The need for depth
- MLPs as universal classifiers
 - The need for depth
- MLPs as universal approximators
- A discussion of optimal depth and width
- Brief segue: RBF networks

MLP as a continuous-valued regression



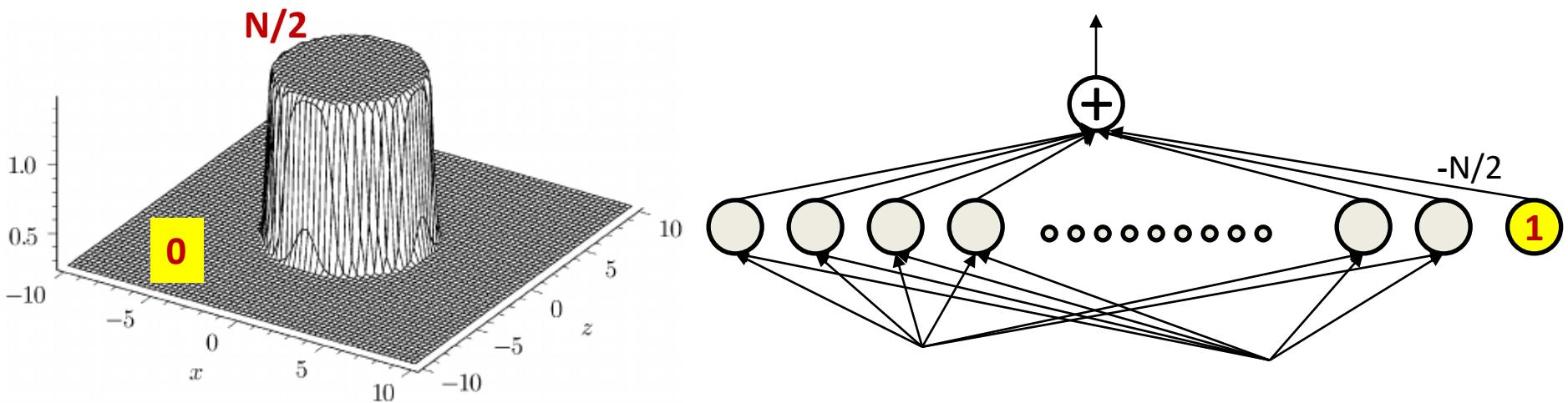
- A simple 3-unit MLP with a “summing” output unit can generate a “square pulse” over an input
 - Output is 1 only if the input lies between T_1 and T_2
 - T_1 and T_2 can be arbitrarily specified

MLP as a continuous-valued regression



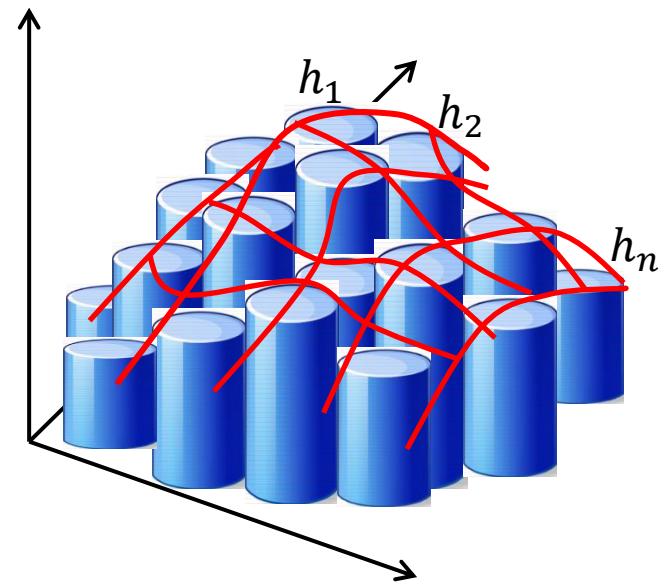
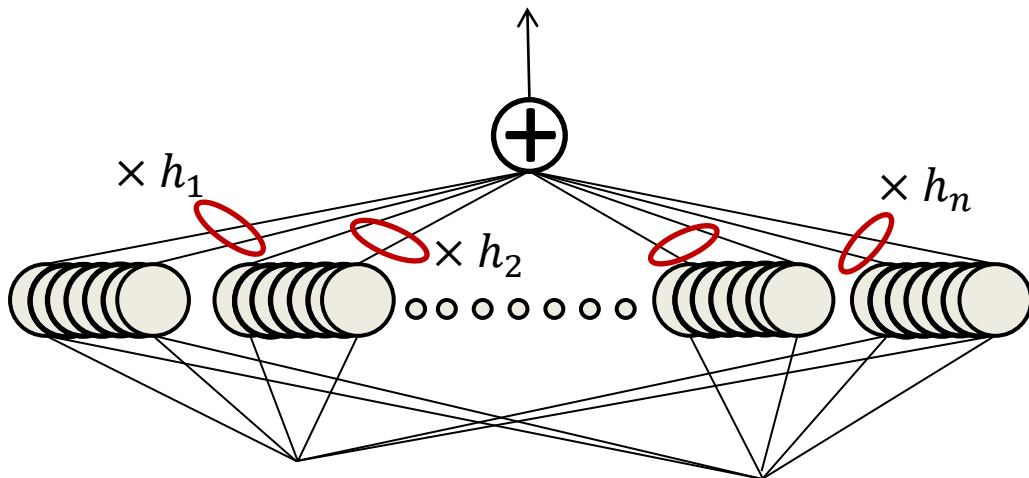
- A simple 3-unit MLP can generate a “square pulse” over an input
- An MLP with many units can model an arbitrary function over an input
 - To arbitrary precision
 - Simply make the individual pulses narrower
- A one-hidden-layer MLP can model an arbitrary function of a single input

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!**

1 hidden layer

Poll 4

Any real valued function can be modelled exactly by a one-hidden layer network with infinite neurons in the hidden layer, true or false?

- False
- True

Poll 4

Any real valued function can be modelled exactly by a one-hidden layer network with infinite neurons in the hidden layer, true or false?

- **False**
- True

Explanation: (it can only be approximated)

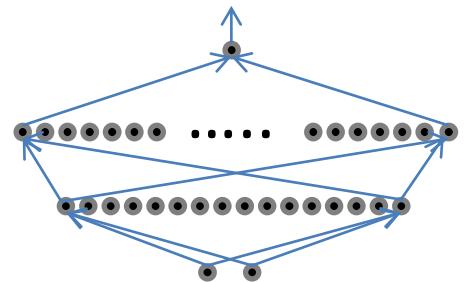
Today

- 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 *sufficient* depth and width
- Brief segue: RBF networks

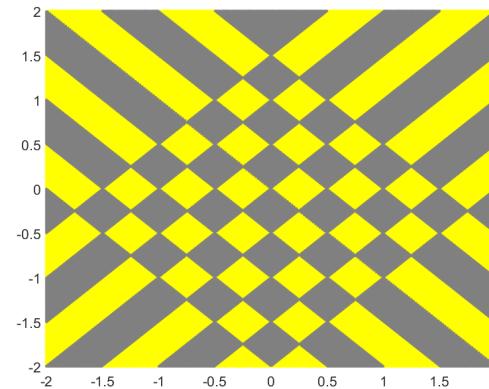
The issue of depth

- 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

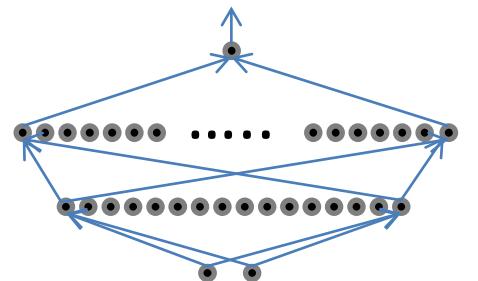


A network with 16 or more neurons in the first layer is capable of representing the figure to the right perfectly

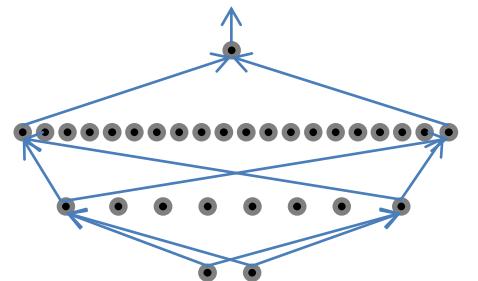


- A neural network *can* represent any function provided it has sufficient *capacity*
 - i.e. sufficiently broad and deep to represent the function
- Not all architectures can represent any function

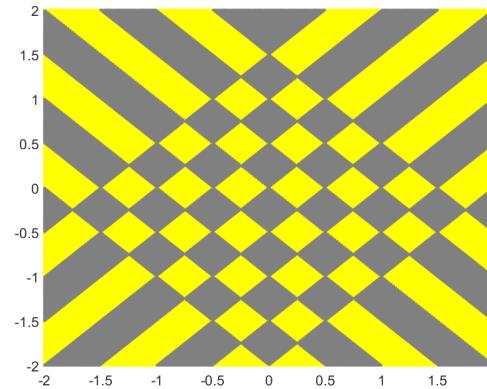
Sufficiency of architecture



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



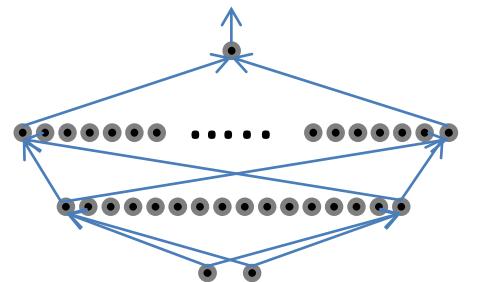
A network with less than 16 threshold-activation neurons in the first layer cannot represent this pattern exactly



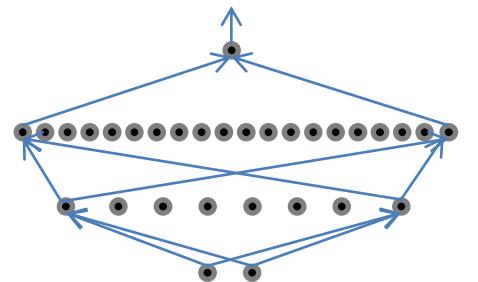
Why?

- 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

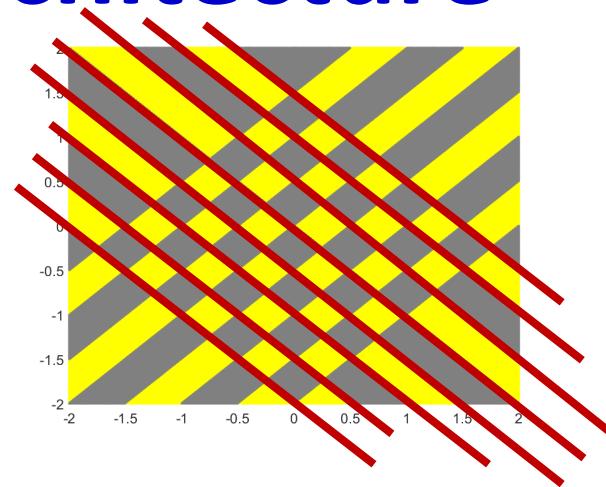
Sufficiency of architecture



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



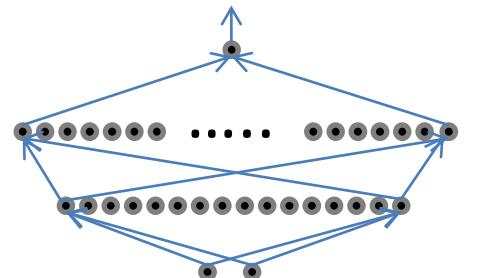
A network with less than 16 threshold-activation neurons in the first layer cannot represent this pattern exactly



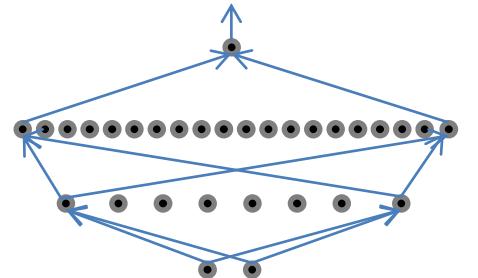
Why?

- A network with only 8 threshold neurons in the first layer may capture these 8 boundaries

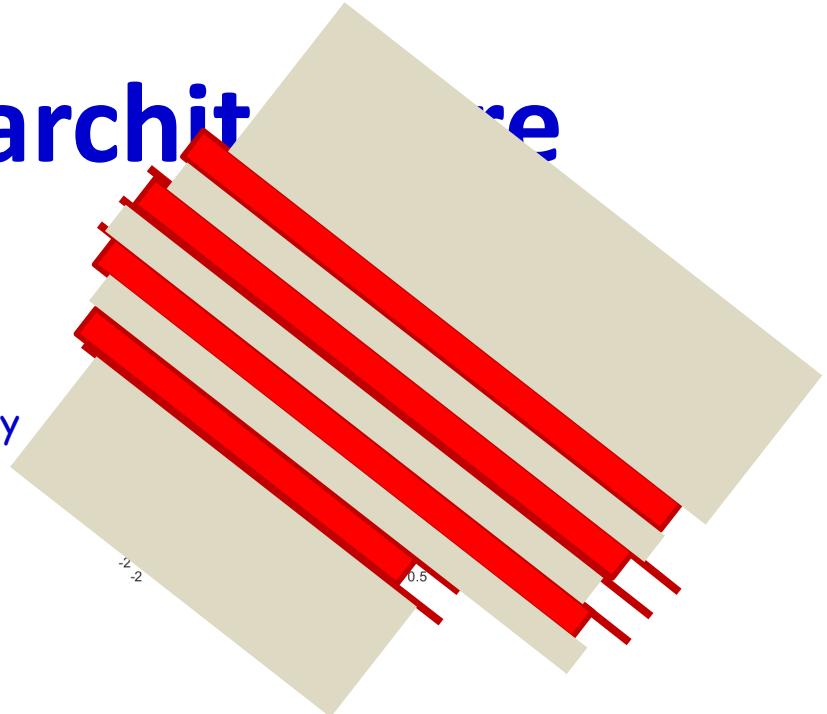
Sufficiency of architecture



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

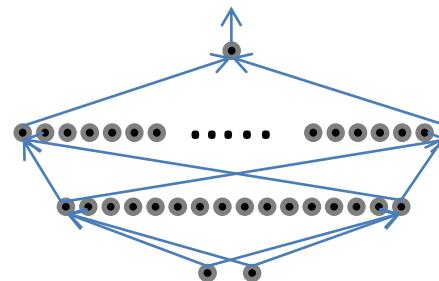


A network with less than 16 threshold-activation neurons in the first layer cannot represent this pattern exactly

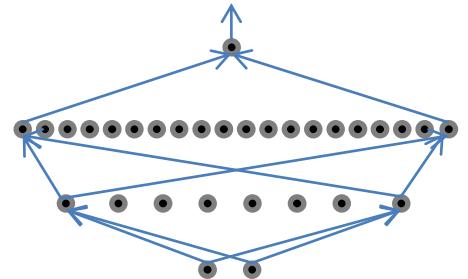


- A network with only 8 threshold neurons in the first layer may capture these 8 boundaries
- That can only give you information about which of these strips the input is in, but not *where* in the strip

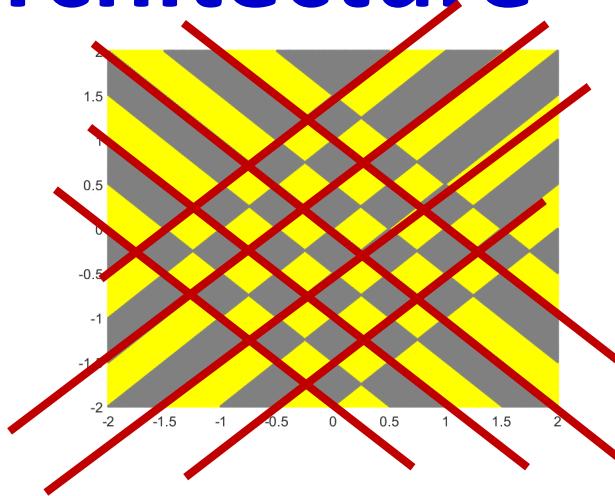
Sufficiency of architecture



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

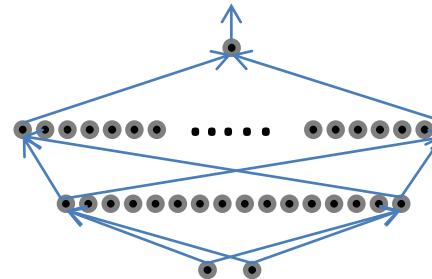


A network with less than 16 threshold-activation neurons in the first layer cannot represent this pattern exactly

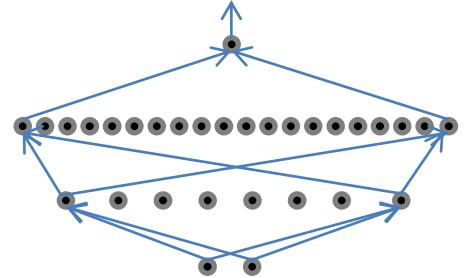


- Even if the 8 first-layer neurons capture *these* boundaries...

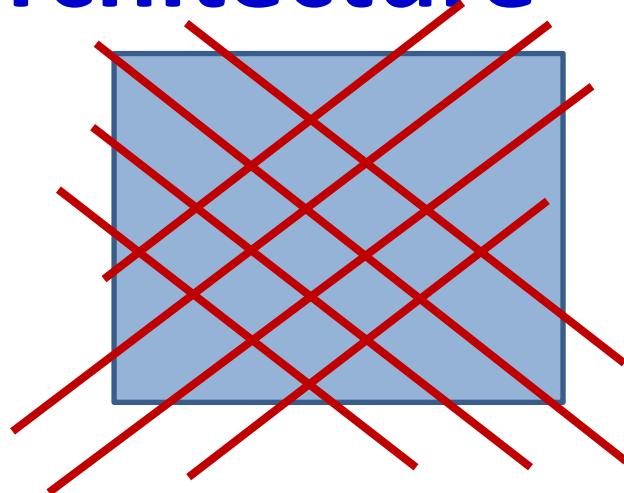
Sufficiency of architecture



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

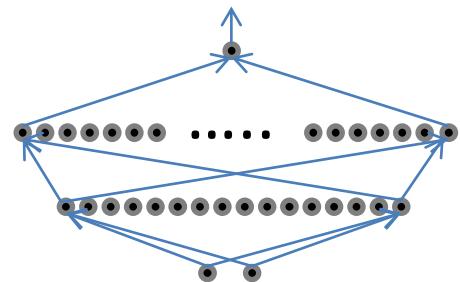


A network with less than 16 threshold-activation neurons in the first layer cannot represent this pattern exactly

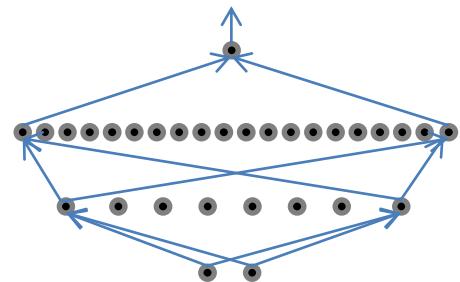
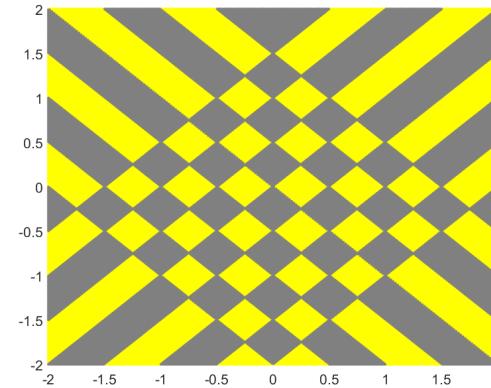


- Even if the 8 first-layer neurons capture *these* boundaries...
- ... they can only place you in one of these 25 cells, but cannot inform you of *where* in the cell

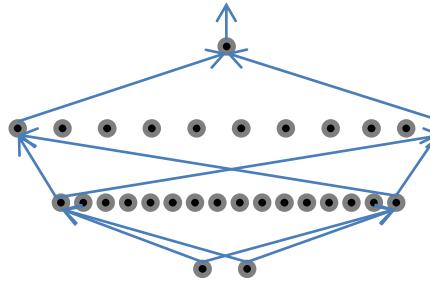
Sufficiency of architecture



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



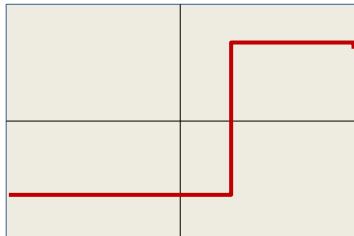
A network with less than 16 threshold-activation neurons in the first layer cannot represent this pattern exactly



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

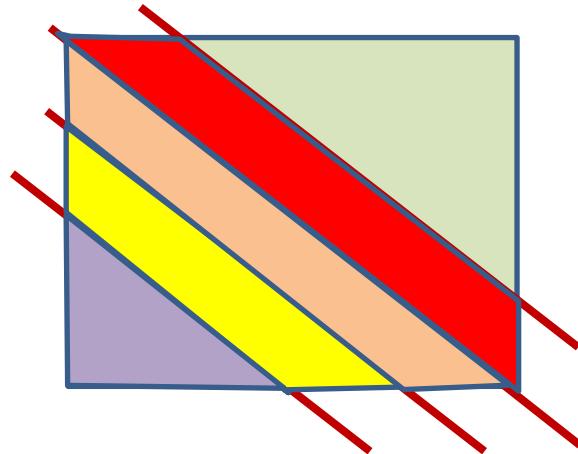
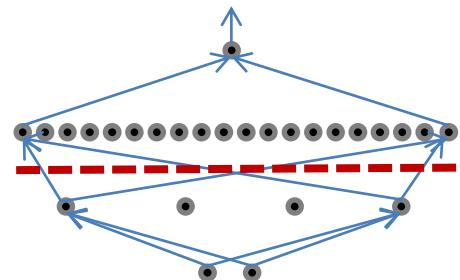
- Similar restrictions apply to higher layers
- Regardless of depth, every layer must be sufficiently wide in order to capture the function
- Not all architectures can represent any function

Sufficiency of architecture



This effect is because we use the threshold activation

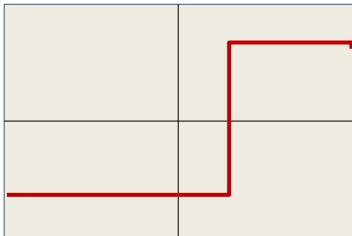
It *gates* information in the input from later layers



The pattern of outputs within any colored region is identical

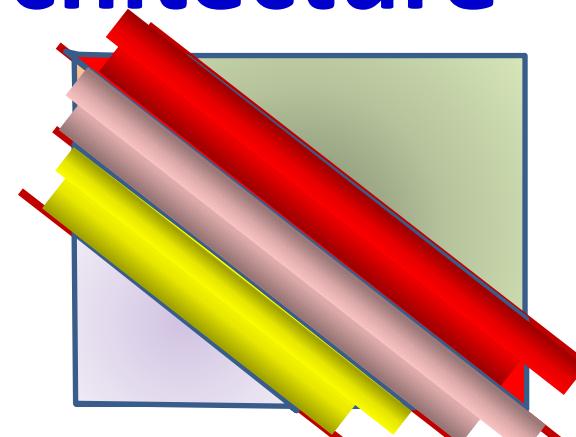
Subsequent layers do not obtain enough information to partition them

Sufficiency of architecture



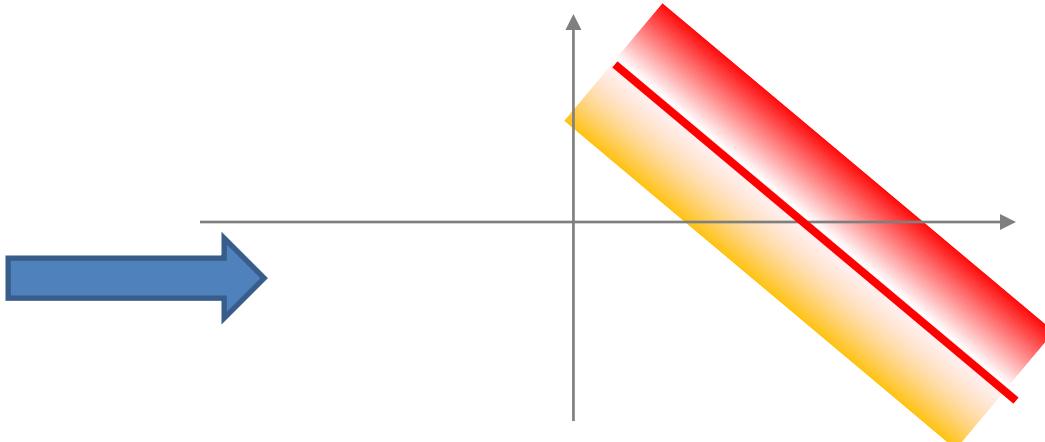
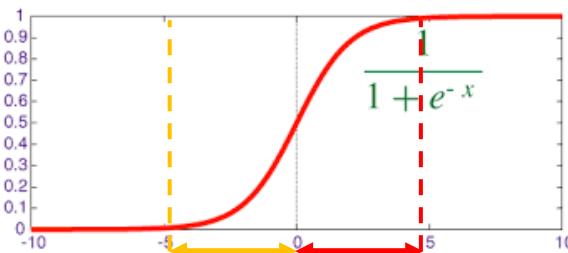
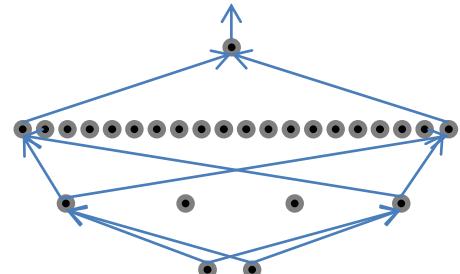
This effect is because we use the threshold activation

It *gates* information in the input from later layers



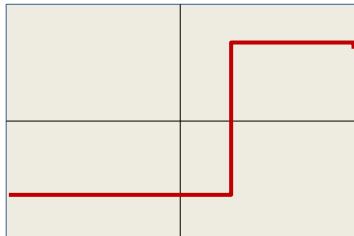
Continuous activation functions result in graded output at the layer

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



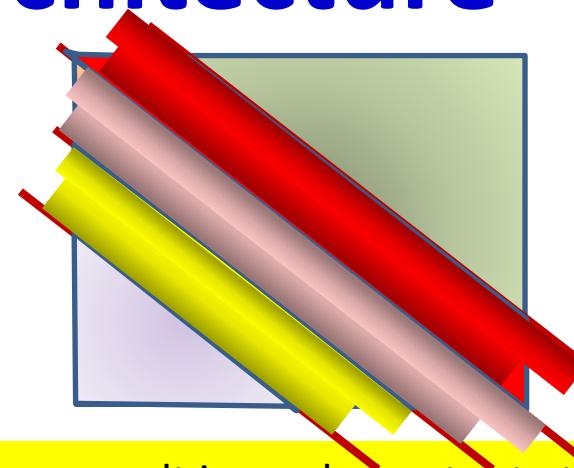
Which is why threshold act func is BAD.

Sufficiency of architecture



This effect is because we use the threshold activation

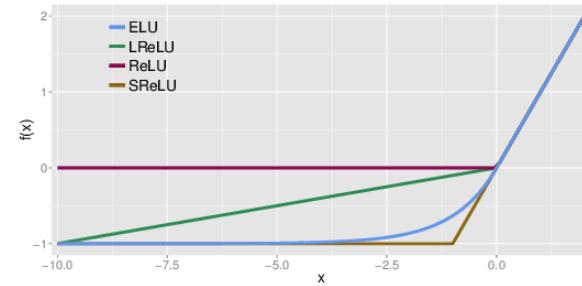
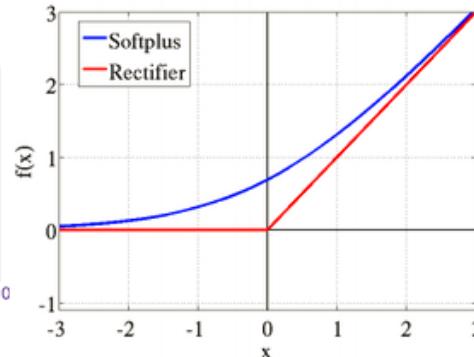
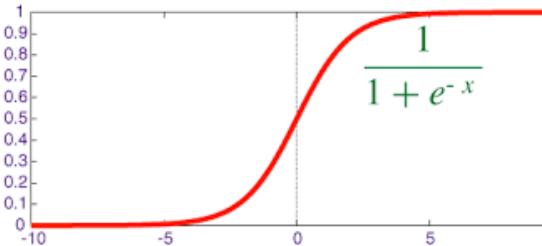
It *gates* information in the input from later layers



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

Lessons so far

- MLPs are universal function approximators
 - Can model any Boolean function, classification function, or regression
- Deeper MLPs can achieve the same precision with far fewer neurons, but must still have sufficient *capacity*
 - The activations must pass information through
 - Each layer must still be sufficiently wide to convey all relevant information to subsequent layers

Poll 5

Mark all true statements

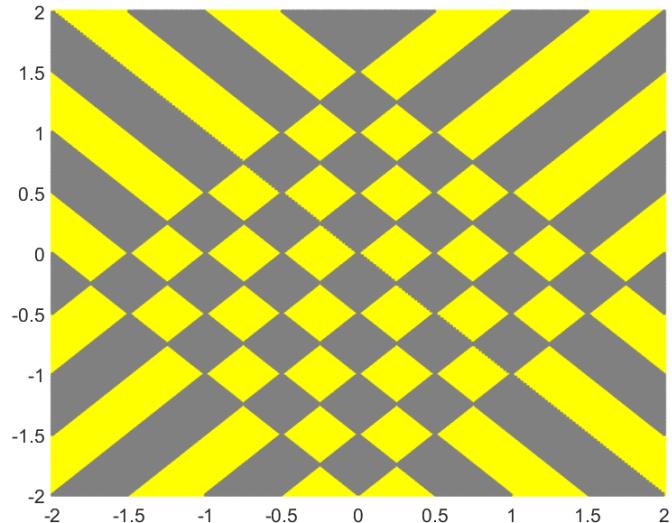
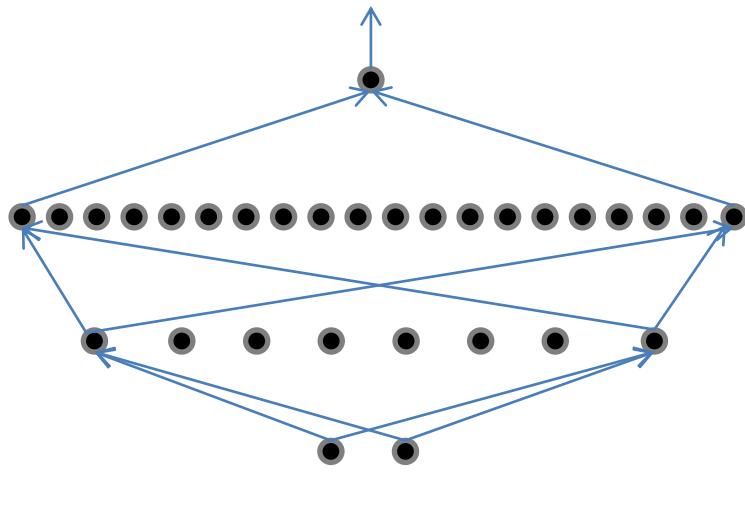
- A network with an upper bound on layer width (no. of neurons in a layer) can nevertheless model any function by making it sufficiently deep.
- Networks with "graded" activation functions are more able to compensate for insufficient width through depth, than those with threshold or saturating activations.
- We can always compensate for limits in the width and depth of the network by using more graded activations.
- For a given accuracy of modelling a function, networks with more graded activations will generally be smaller than those with less graded (i.e saturating or thresholding) activations.

Poll 5

Mark all true statements

- A network with an upper bound on layer width (no. of neurons in a layer) can nevertheless model any function by making it sufficiently deep.
- **Networks with "graded" activation functions are more able to compensate for insufficient width through depth, than those with threshold or saturating activations.**
- We can always compensate for limits in the width and depth of the network by using more graded activations.
- **For a given accuracy of modelling a function, networks with more graded activations will generally be smaller than those with less graded (i.e saturating or thresholding) activations.**

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
 - Battlott, 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 exists 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

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*