

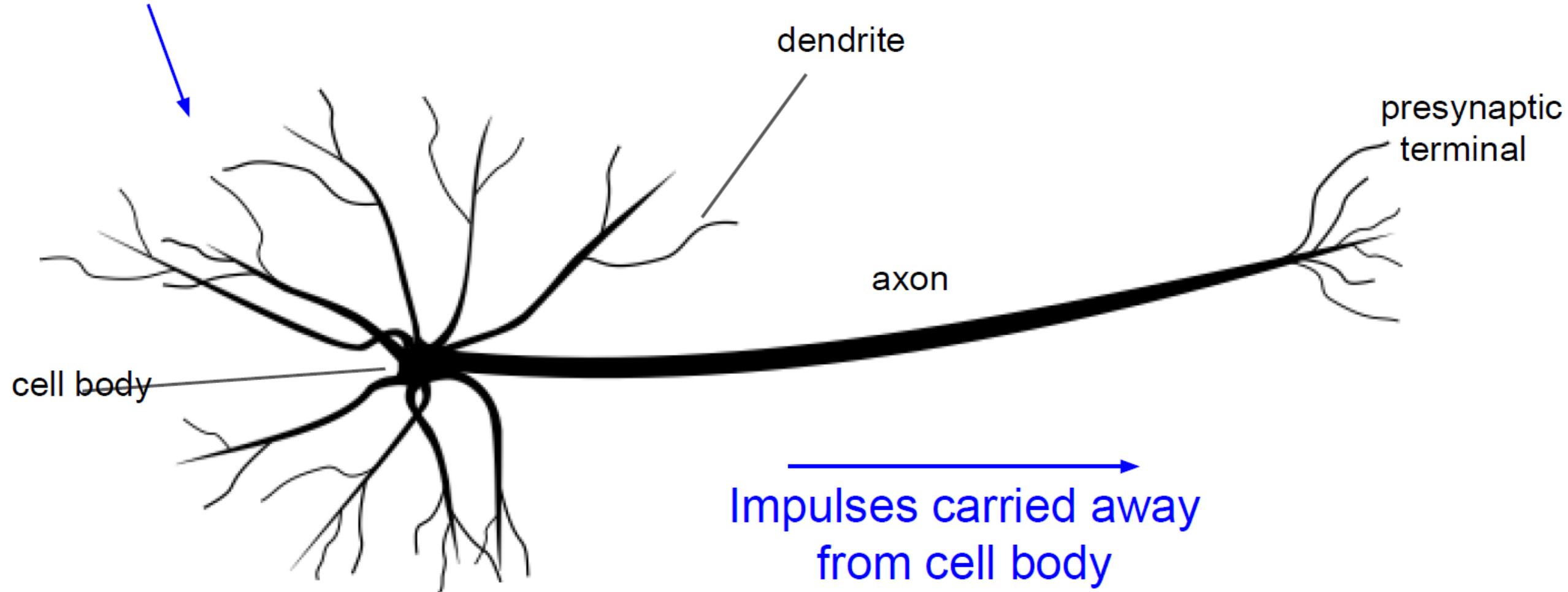
# Multi-Layer Networks

M. Soleymani  
Deep Learning

Sharif University of Technology  
Spring 2024

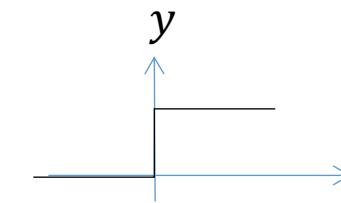
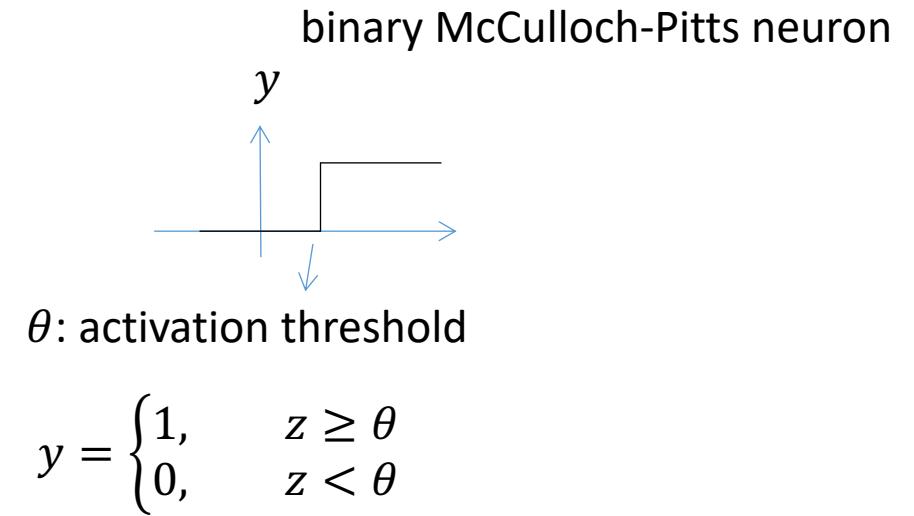
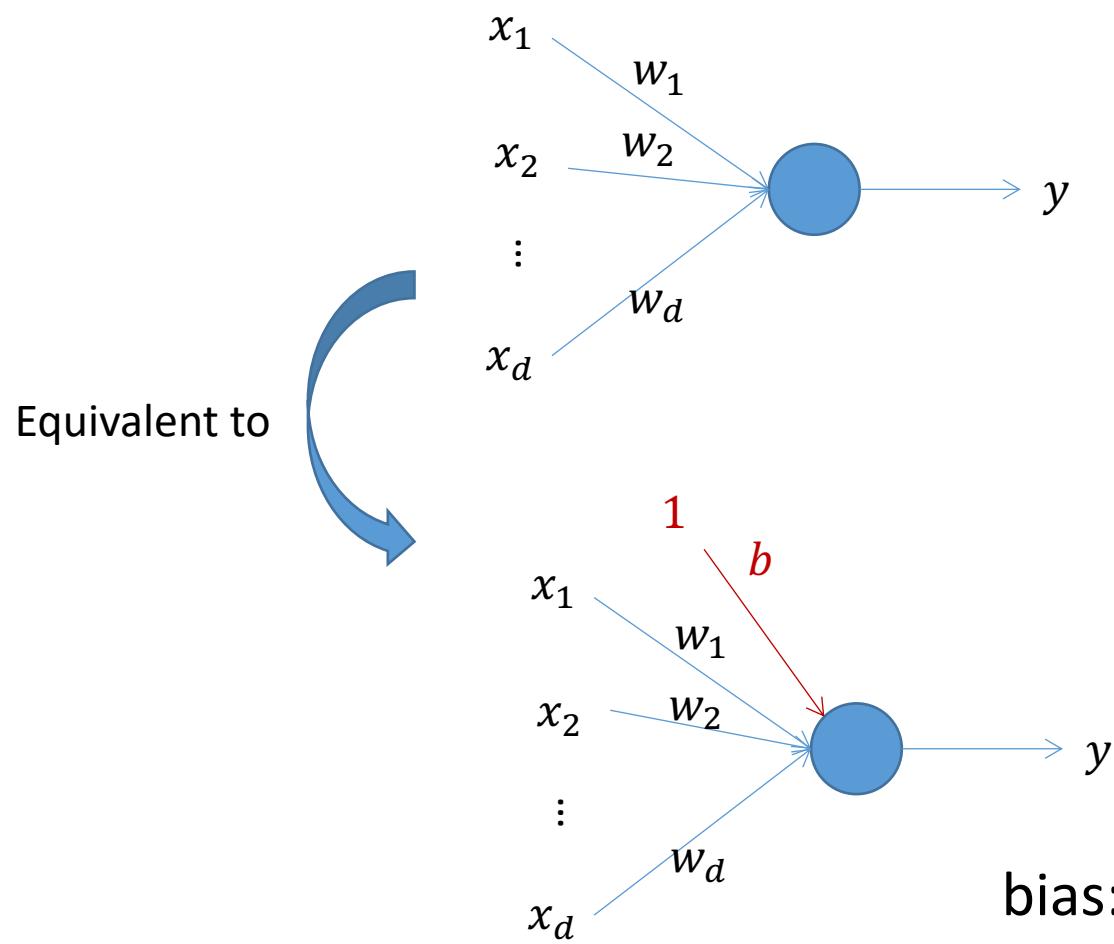
Most slides have been adapted from:  
Bhiksha Raj, 11-785, CMU

Impulses carried toward cell body

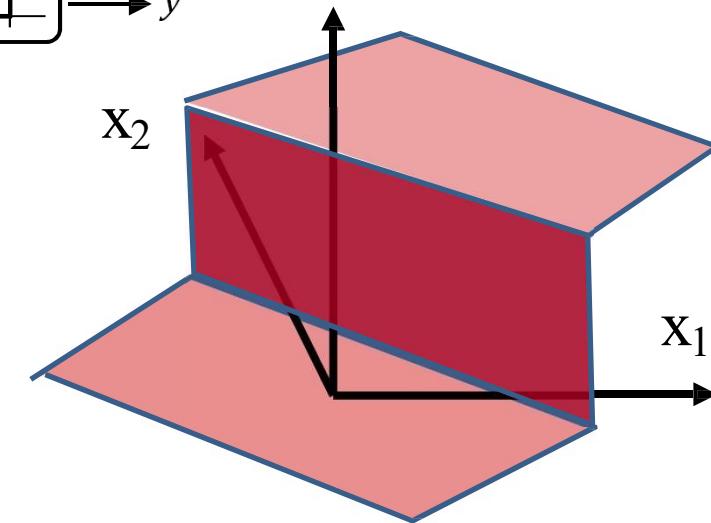
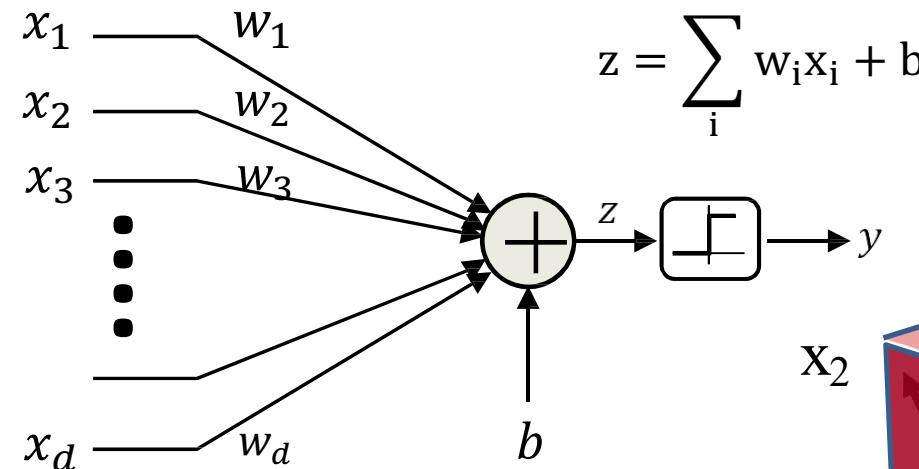
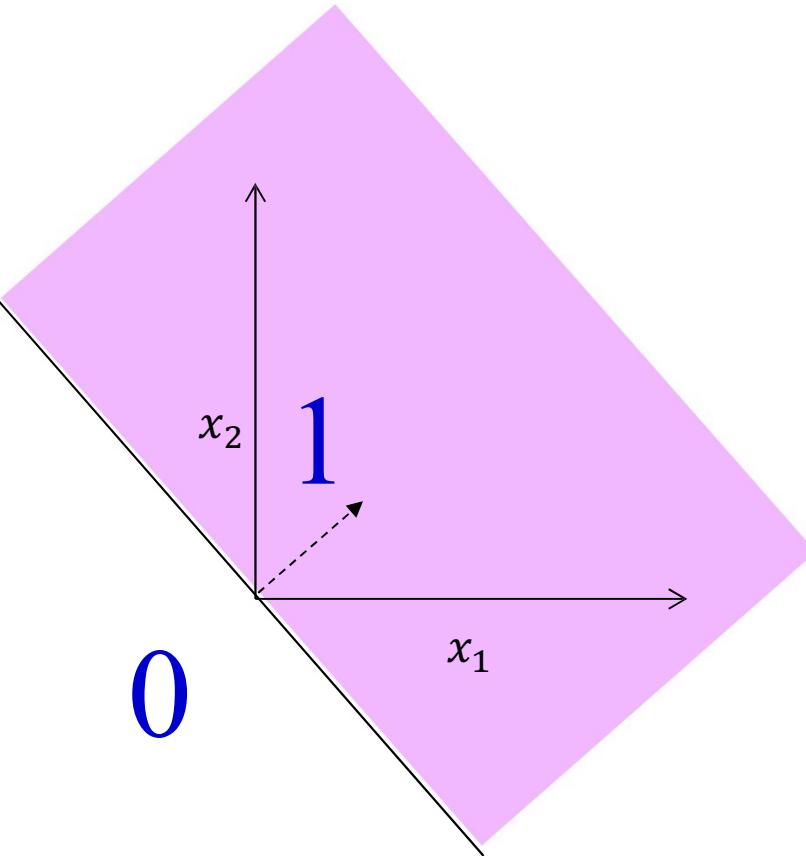


[This image](#) by Felipe Perucho  
is licensed under [CC-BY 3.0](#)

# McCulloch-Pitts neuron: binary threshold

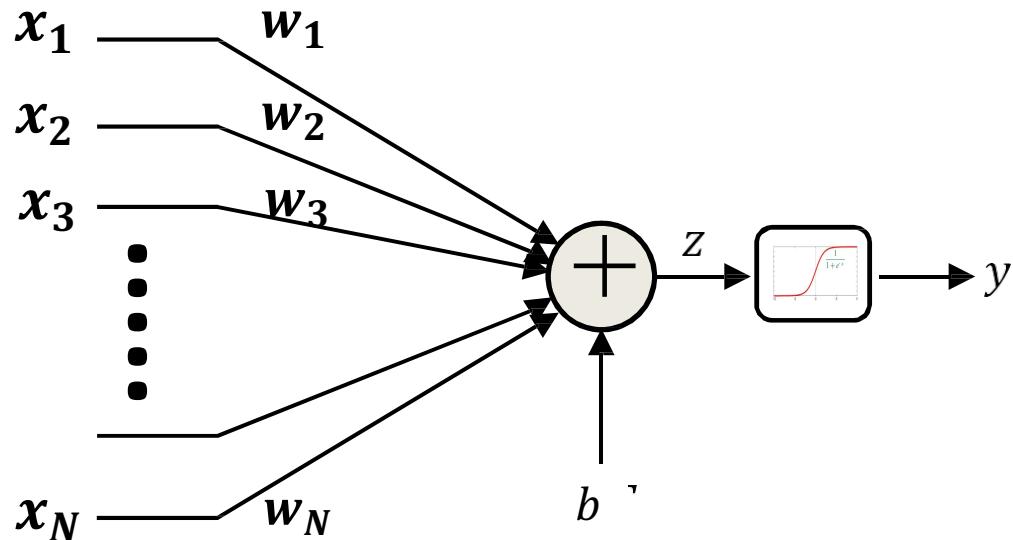


# Perceptron



- Learn this function
  - A step function across a hyperplane

# The “soft” perceptron (logistic)

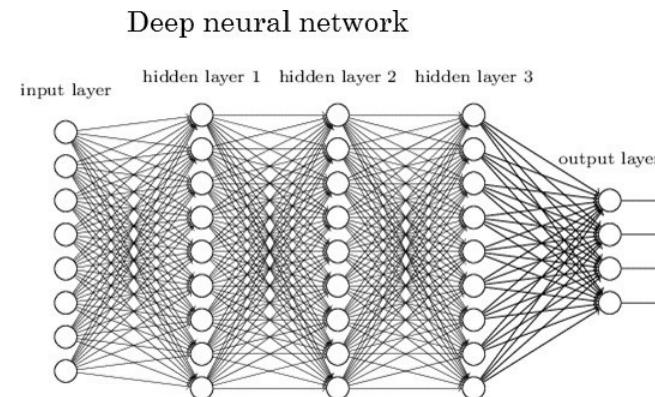
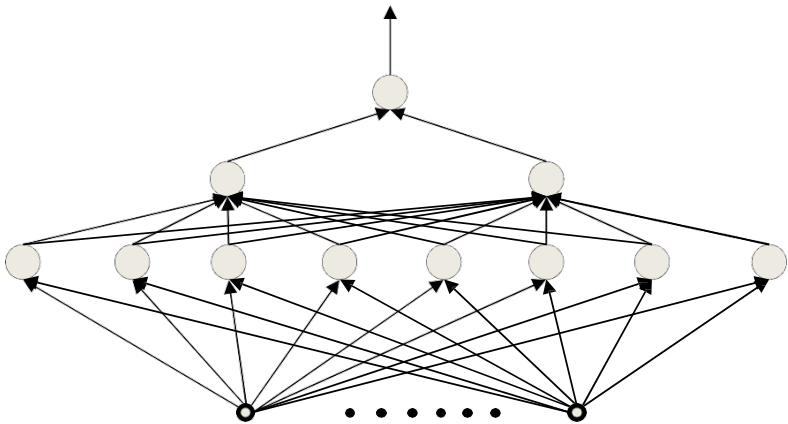


$$z = \sum_i w_i x_i + b$$

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

- 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)

# The multi-layer perceptron

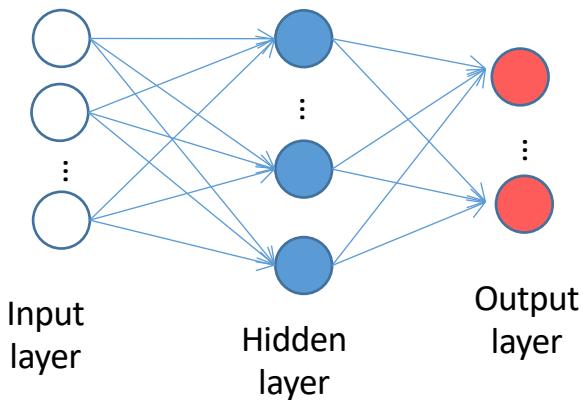


- A network of perceptrons
  - Generally “layered”

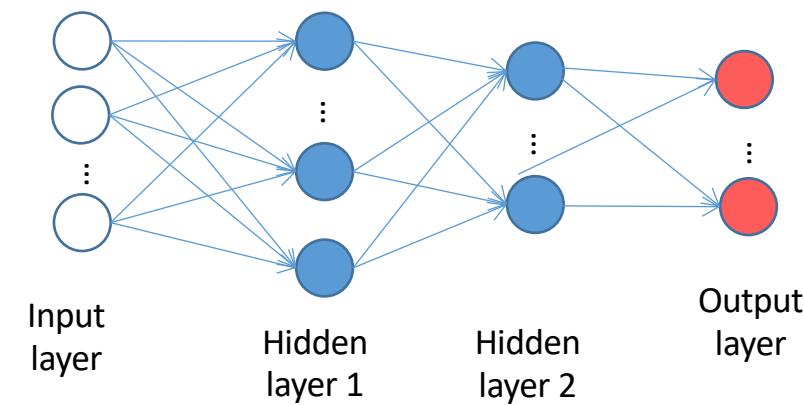


# Feed-forward neural networks

- Also called **Multi-Layer Perceptron (MLP)**



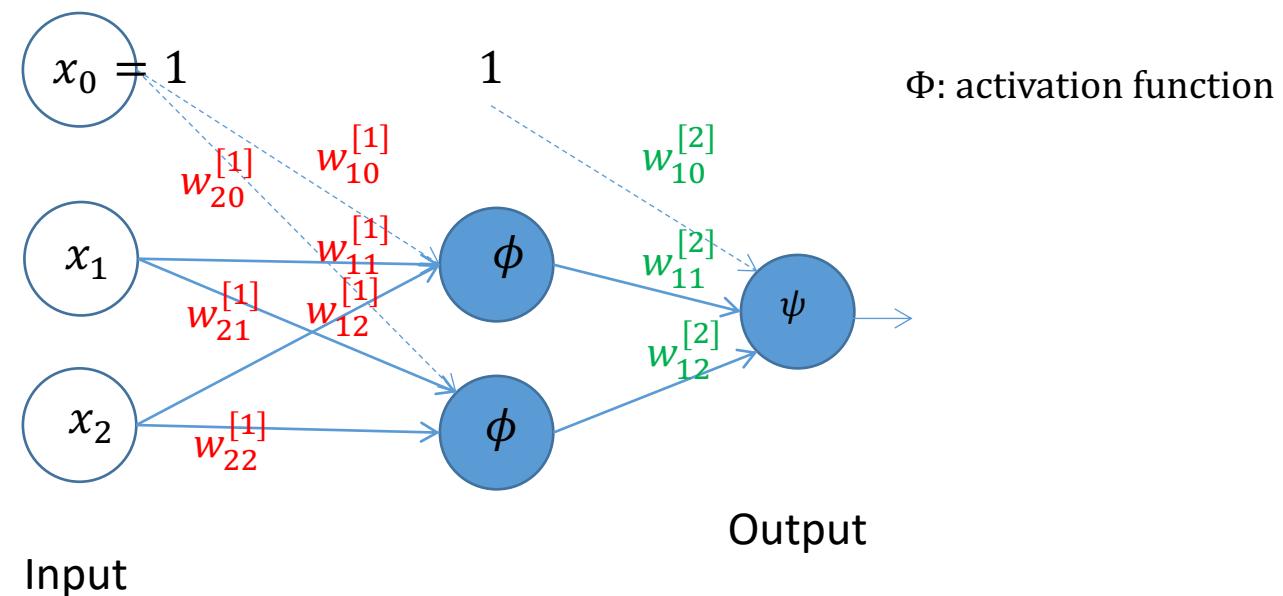
1-Hidden Layer  
or 2-layer network



2-Hidden Layers  
or 3-layer network

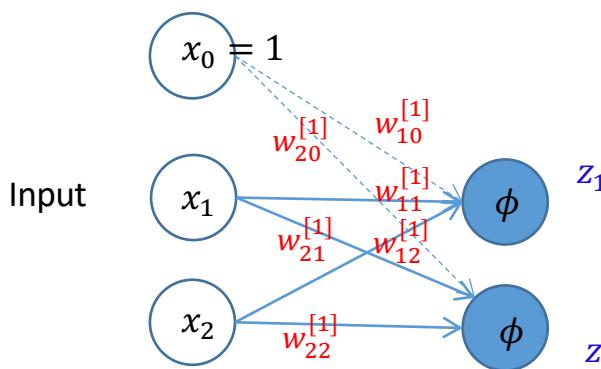
# Multi-layer Neural Network

- MLP with single hidden layer
  - Called two-layer MLP (Number of layers of adaptive weights is counted)



# Multi-layer Neural Network

$$\mathbf{W}^{[1]} = \begin{bmatrix} w_{10}^{[1]} & w_{11}^{[1]} & w_{12}^{[1]} \\ w_{20}^{[1]} & w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix}$$



$\phi$ : fixed activation function

Examples:

$$\phi(z) = \max(0, z)$$

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

$$z_j = \phi \left( \sum_{i=0}^d w_{ji}^{[1]} x_i \right)$$

Matrix form:

$$\mathbf{z} = \phi \left( \mathbf{W}^{[1]} \mathbf{x} \right)$$

# Multi-layer Neural Network

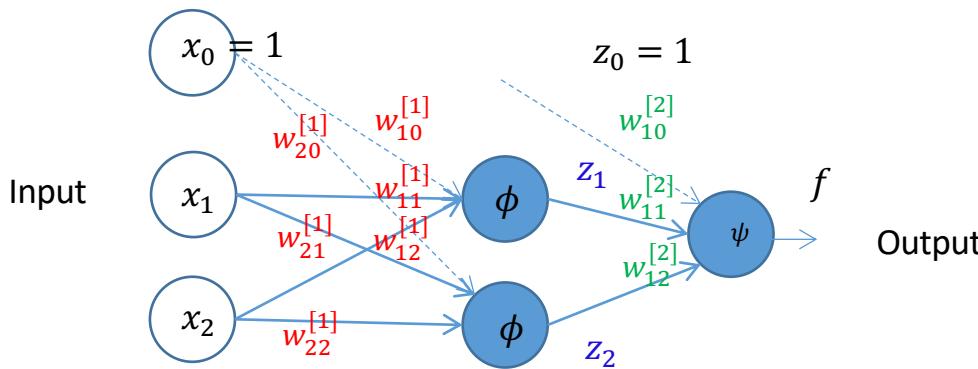
$$\mathbf{W}^{[1]} = \begin{bmatrix} w_{10}^{[1]} & w_{11}^{[1]} & w_{12}^{[1]} \\ w_{20}^{[1]} & w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix}$$

$\phi$ : fixed activation function

Examples:

$$\phi(z) = \max(0, z)$$

$$\phi(z) = \frac{1}{1 + e^{-z}}$$



Input

Output

$$z_j = \phi \left( \sum_{i=0}^d w_{ji}^{[1]} x_i \right)$$

$$f = \psi \left( \sum_{j=0}^2 w_{kj}^{[2]} z_j \right)$$

Matrix form:

$$\mathbf{z} = \phi(\mathbf{W}^{[1]} \mathbf{x})$$

$$\begin{aligned} f(\mathbf{x}) &= \psi(\mathbf{W}^{[2]} \mathbf{z}) \\ &= \psi(\mathbf{W}^{[2]} \phi(\mathbf{W}^{[1]} \mathbf{x})) \end{aligned}$$

# Beyond linear models

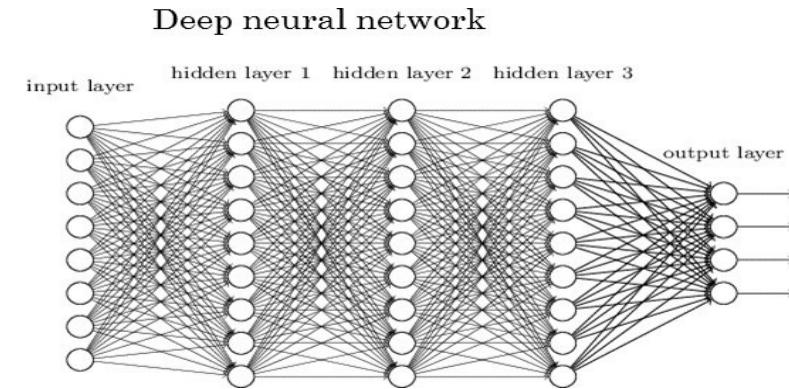
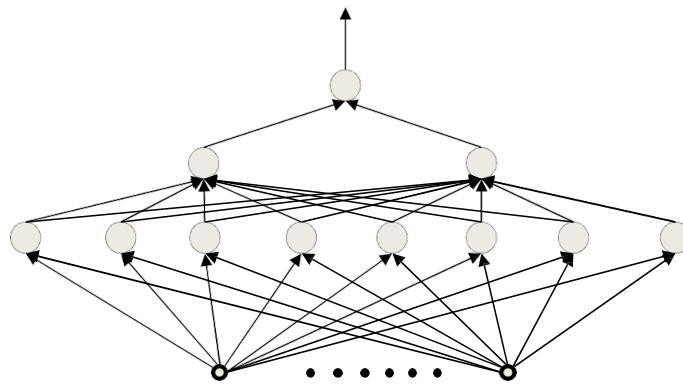
(Before) Linear score function:  $f = Wx$

(Now) 2-layer Neural Network  $f = W_2\phi(W_1x)$

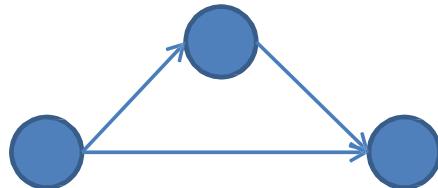
or 3-layer Neural Network

$$f = W_3\phi(W_2\phi(W_1x))$$

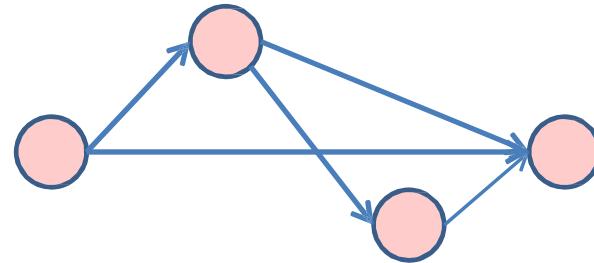
# Deep Structures



- “depth” is the length of the longest path from an input source to an output sink



Left: Depth = 2



Right: Depth = 3

- “Deep”  $\Rightarrow$  Depth  $> 2$

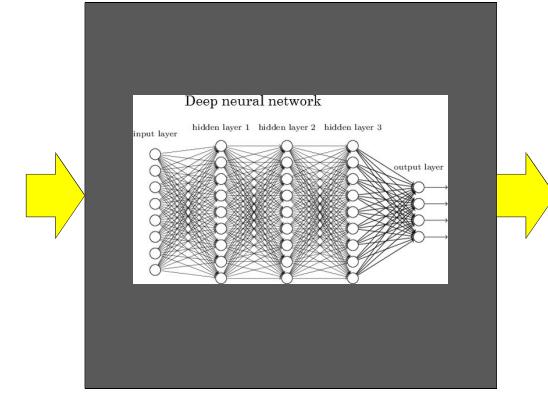
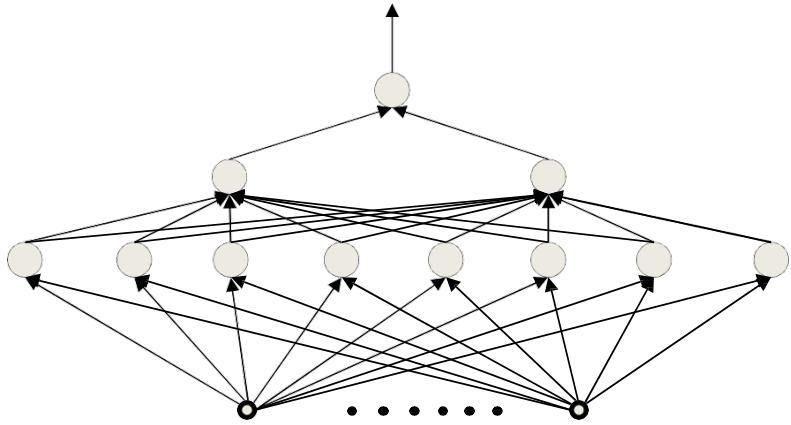
# MLP Universal Approximator

- A feed-forward net with a single hidden layer and linear outputs can approximate any continuous function on a compact subset of  $R^D$  to an arbitrary accuracy
  - under mild assumptions on the activation function
    - e.g., sigmoid activation functions (Cybenko, 1989)
  - when sufficiently large (but finite) number of hidden units is used

$$F_k(x) = \sum_{j=1}^M w_{kj}^{[2]} \phi \left( \sum_{i=0}^d w_{ji}^{[1]} x_i \right)$$

- It is of greater theoretical interest than practical
  - the construction of such a network requires weight values which are unknown

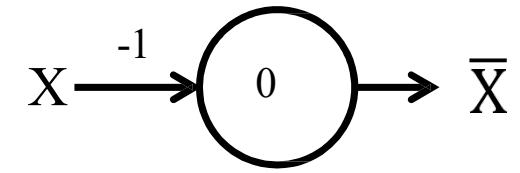
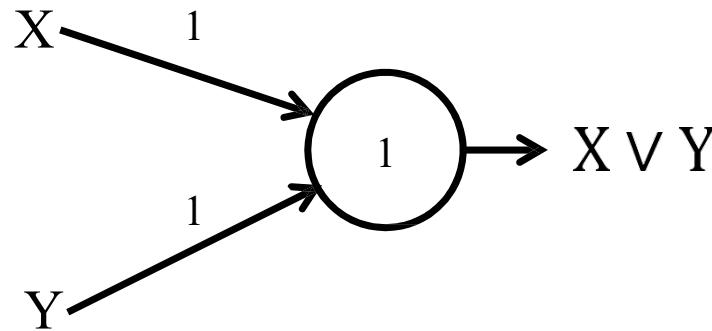
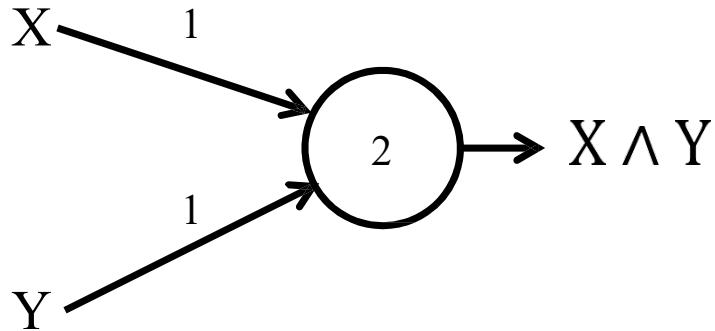
# The multi-layer perceptron



- 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?

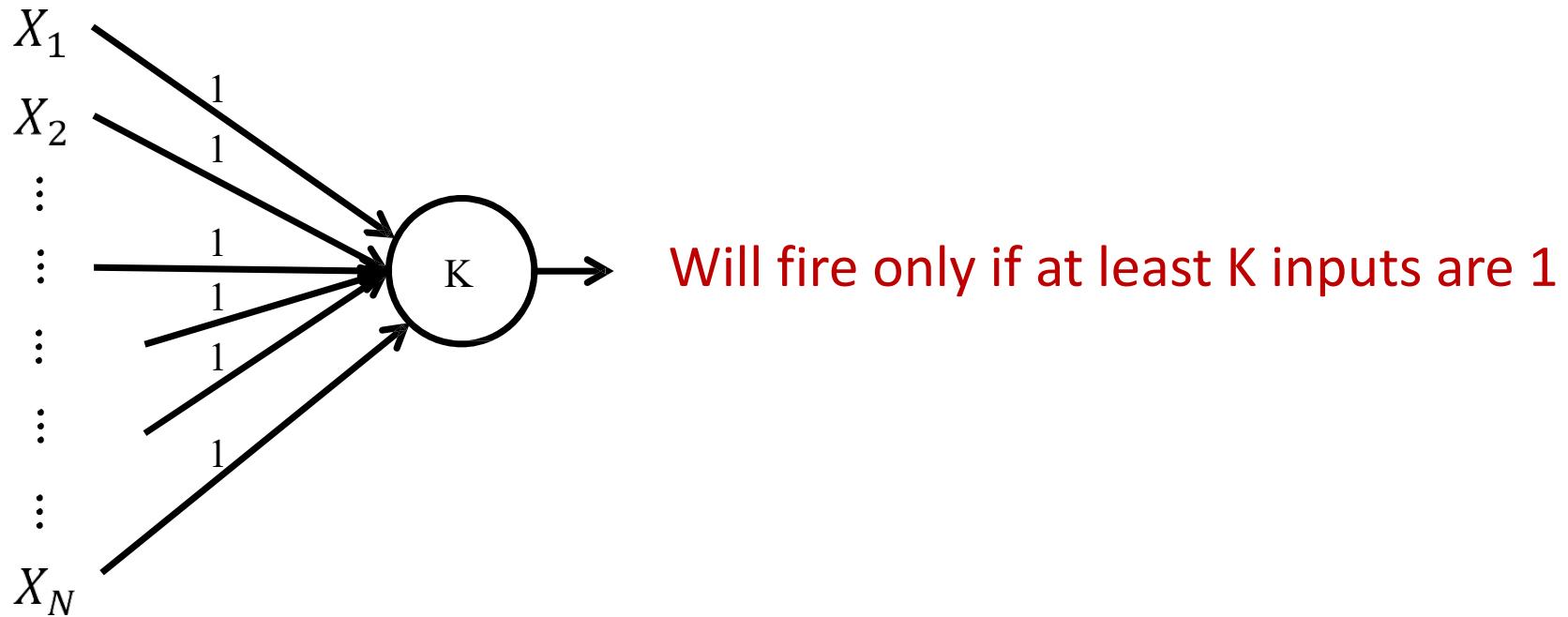
# Multi-layer Perceptrons as universal Boolean functions

# The perceptron as a Boolean gate



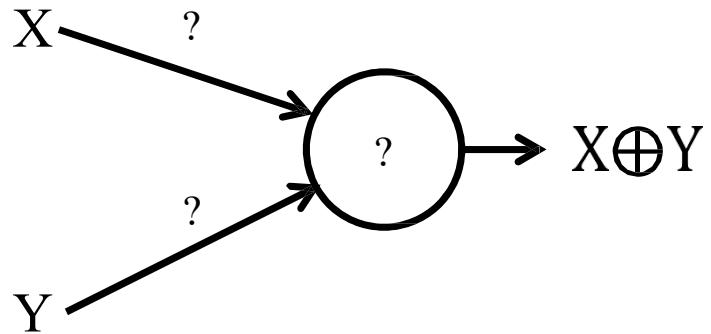
- A perceptron can model any simple binary Boolean gate

# Perceptron as a majority gate



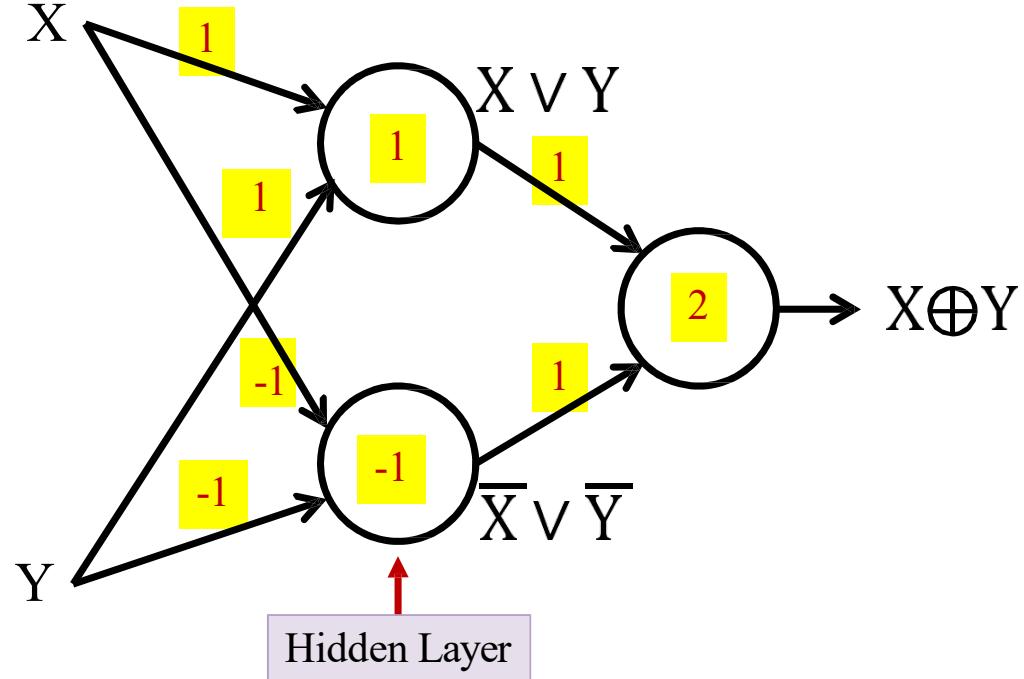
- Generalized **majority** gate
  - Fire if at least K inputs are of the desired polarity

# The perceptron is not enough



- Cannot compute an XOR

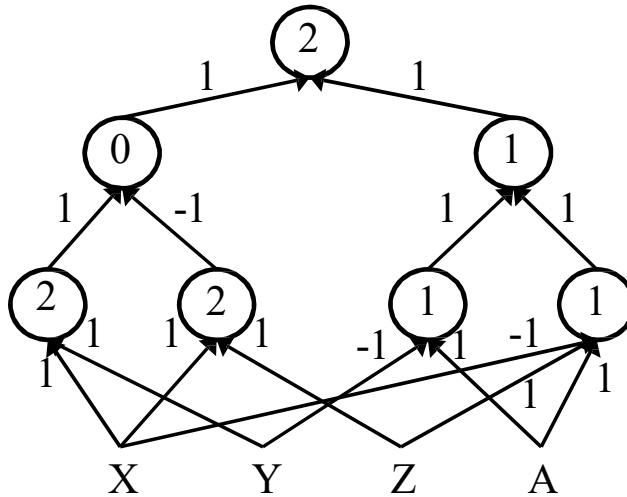
# Multi-layer perceptron XOR



- An XOR takes three perceptrons

# Multi-layer perceptron

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



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

# How many layers for a Boolean MLP?

Truth Table

X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	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

# How many layers for a Boolean MLP?

Truth Table					
X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	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

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

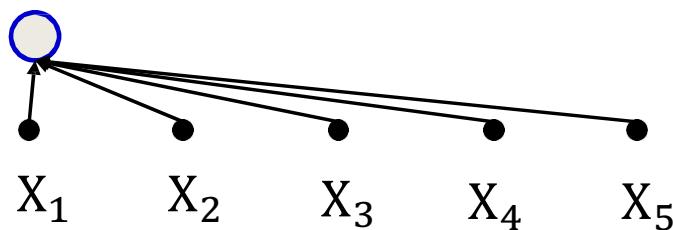
- Expressed in disjunctive normal form

# How many layers for a Boolean MLP?

Truth Table					
X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	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

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



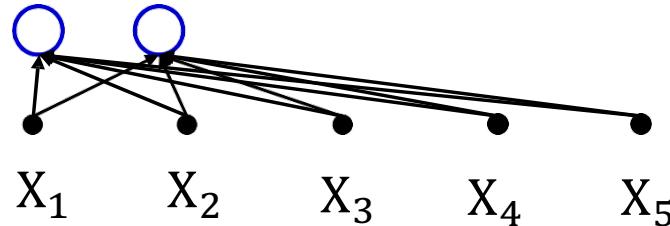
- Expressed in disjunctive normal form

# How many layers for a Boolean MLP?

Truth Table					
X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	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

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



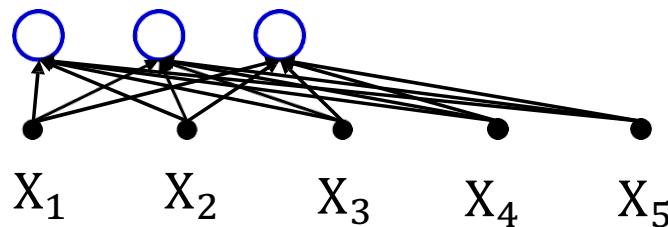
- Expressed in disjunctive normal form

# How many layers for a Boolean MLP?

Truth Table					
X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	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

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



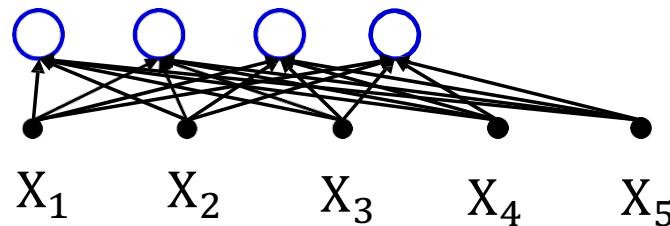
- Expressed in disjunctive normal form

# How many layers for a Boolean MLP?

Truth Table					
X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	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

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



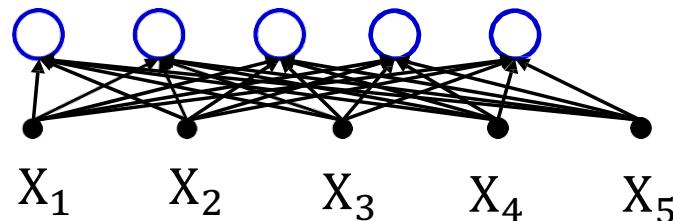
- Expressed in disjunctive normal form

# How many layers for a Boolean MLP?

Truth Table					
X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	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

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



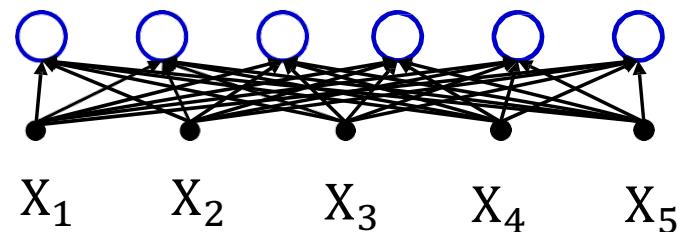
- Expressed in disjunctive normal form

# How many layers for a Boolean MLP?

Truth Table					
X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	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

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



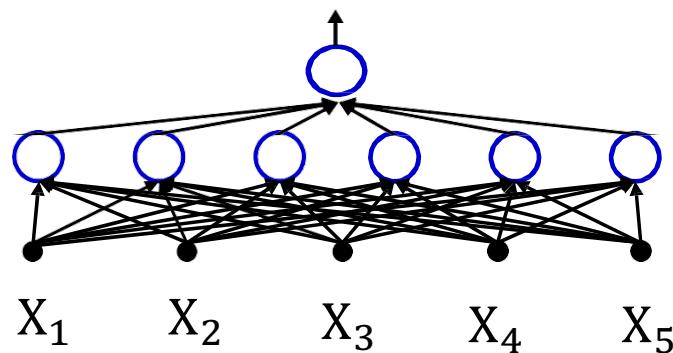
- Expressed in disjunctive normal form

# How many layers for a Boolean MLP?

Truth Table					
X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	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

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

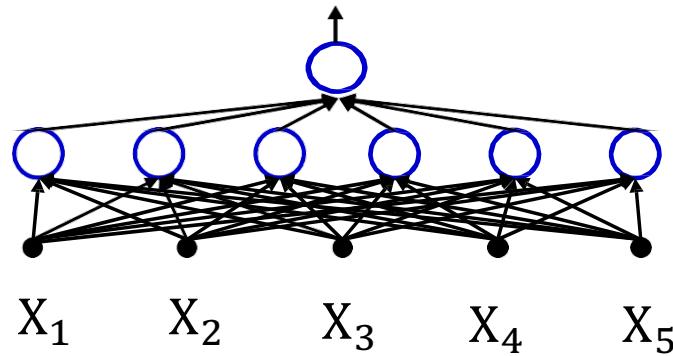


- Expressed in disjunctive normal form

# How many layers for a Boolean MLP?

Truth Table					
X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	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

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



- 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?

# Worst case

- Which truth tables cannot be reduced further simply?
- Largest width needed for a single-layer Boolean network on N inputs
  - Worst case:  $2^{N-1}$ 
    - Example: Parity function

$X, Y$	00	01	11	10
$W, Z$	1	0	1	0
00	1	0	0	1
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

$X \oplus Y \oplus Z \oplus W$

# Boolean functions

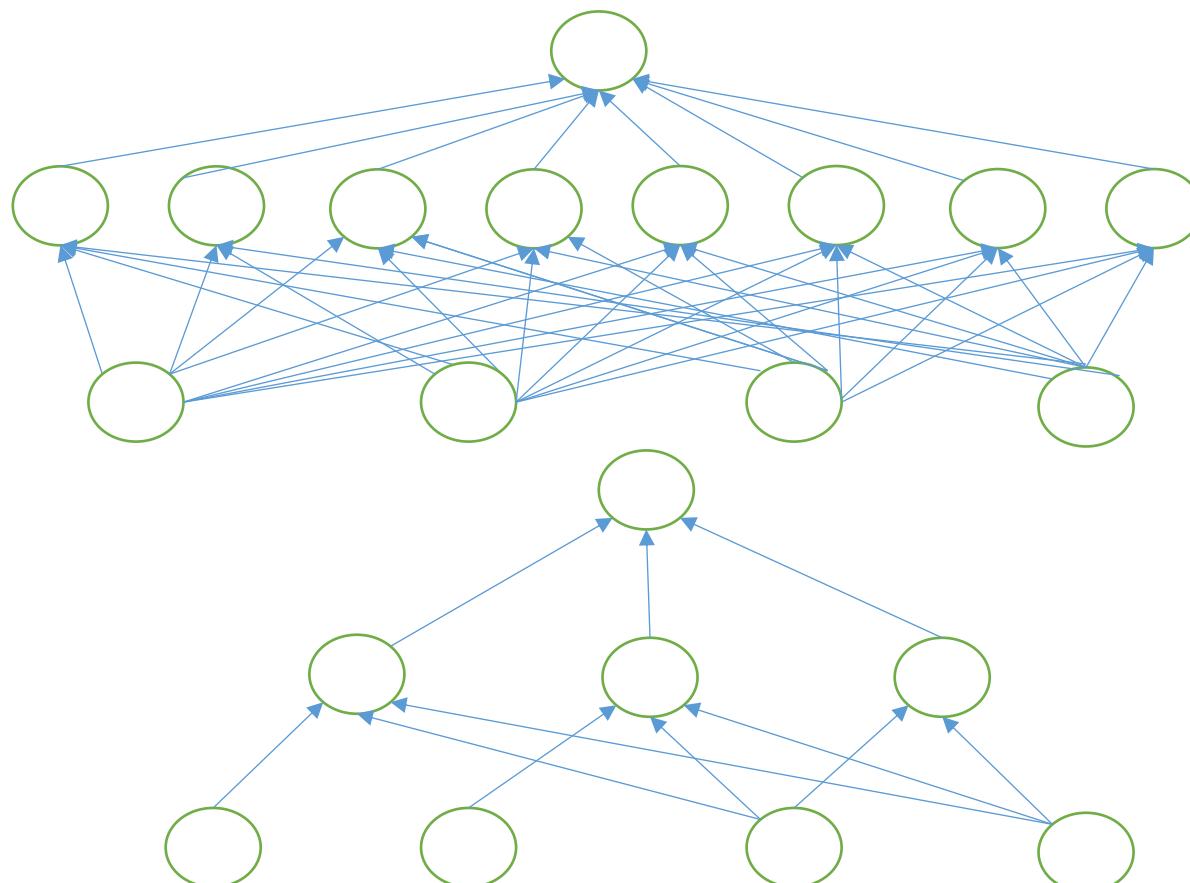
- Input: N Boolean variable
- How many neurons in a one hidden layer MLP is required?
- More compact representation of a Boolean function
  - “Karnaugh Map”
    - representing the truth table as a grid
    - Grouping adjacent boxes to reduce the complexity of the Disjunctive Normal Form (DNF) formula

$X, Y$	00	01	10	11
$W, Z$	1	1	1	1
00				
01				
10		1	1	
11	1			1

# How many neurons in the hidden layer?

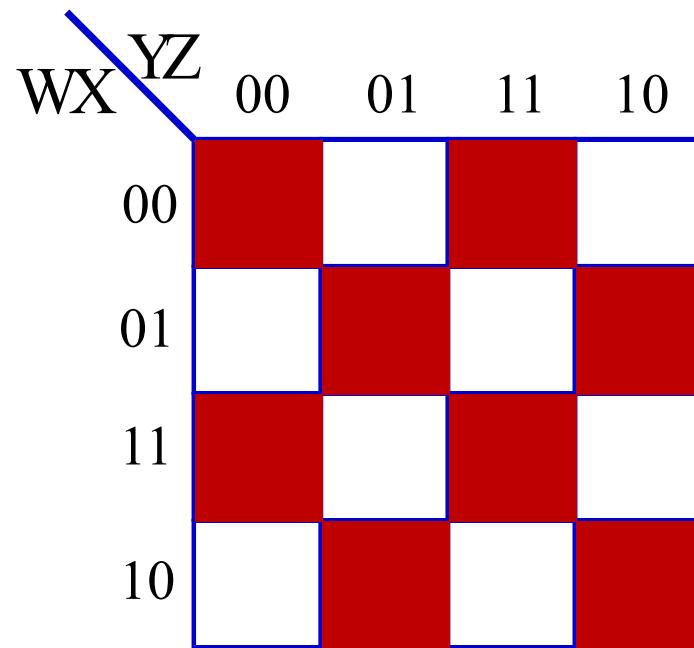
- $\bar{X}\bar{Y}\bar{W}\bar{Z} \vee \bar{X}Y\bar{W}\bar{Z} \vee X\bar{Y}\bar{W}\bar{Z} \vee XY\bar{W}\bar{Z} \vee \bar{X}\bar{Y}WZ \vee X\bar{Y}W\bar{Z} \vee XYW\bar{Z} \vee X\bar{Y}WZ$

$X, Y$	00	01	11	10
$W, Z$	1	1	1	1
00	1			
01				
11	1			1
10			1	1



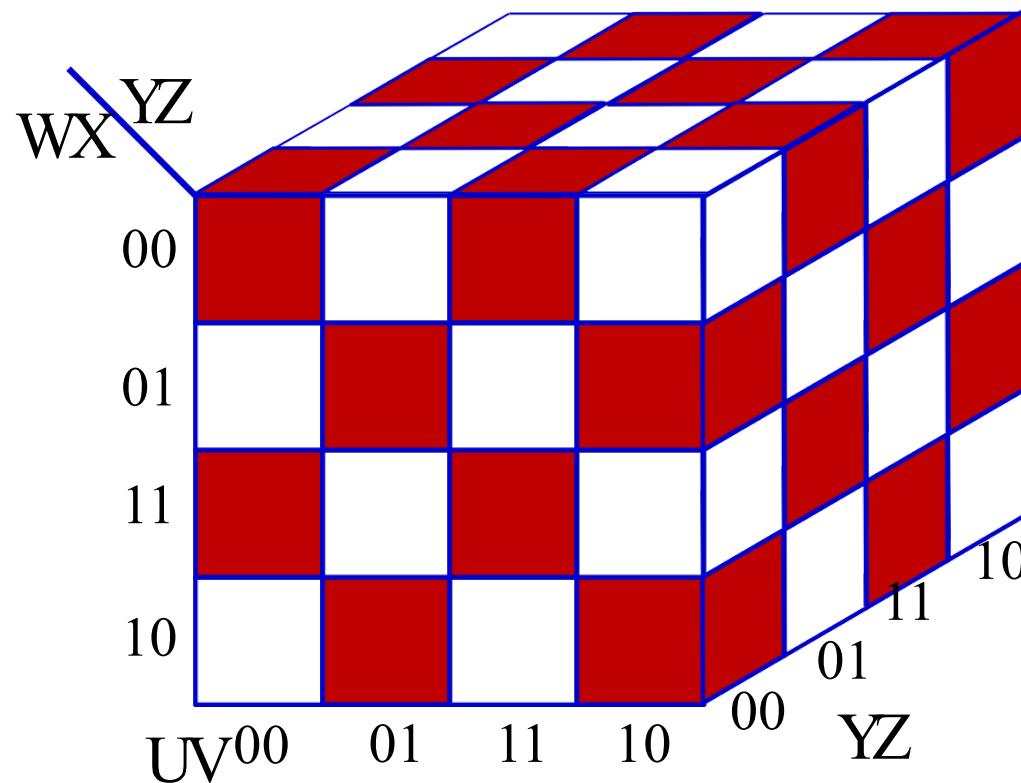
- $\bar{W}\bar{Z} \vee \bar{Y}WZ \vee XW\bar{Z}$

# Largest irreducible DNF?



- What arrangement of ones and zeros simply cannot be reduced further?
- How many neurons in a DNF (one hidden-layer) MLP for this Boolean function?

# Width of a single-layer Boolean MLP



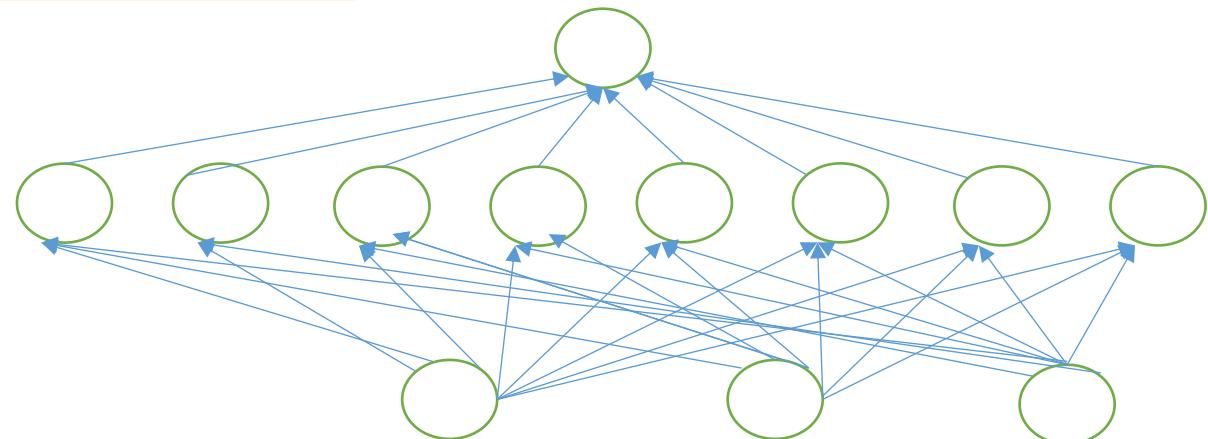
- How many neurons in a DNF (one hidden-layer) MLP for this Boolean function of 6 variables?
- Can be generalized: Will require  $2^{D-1}$  perceptrons in hidden layer **Exponential in D**
- How many units if we use multiple layers?

# Using deep network: Parity function on N inputs

- Simple MLP with one hidden layer:

$2^{D-1}$  Hidden units

$(D + 2)2^{D-1} + 1$  Weights and biases



# Using deep network: Parity function on N inputs

- Simple MLP with one hidden layer:

$2^{D-1}$  Hidden units

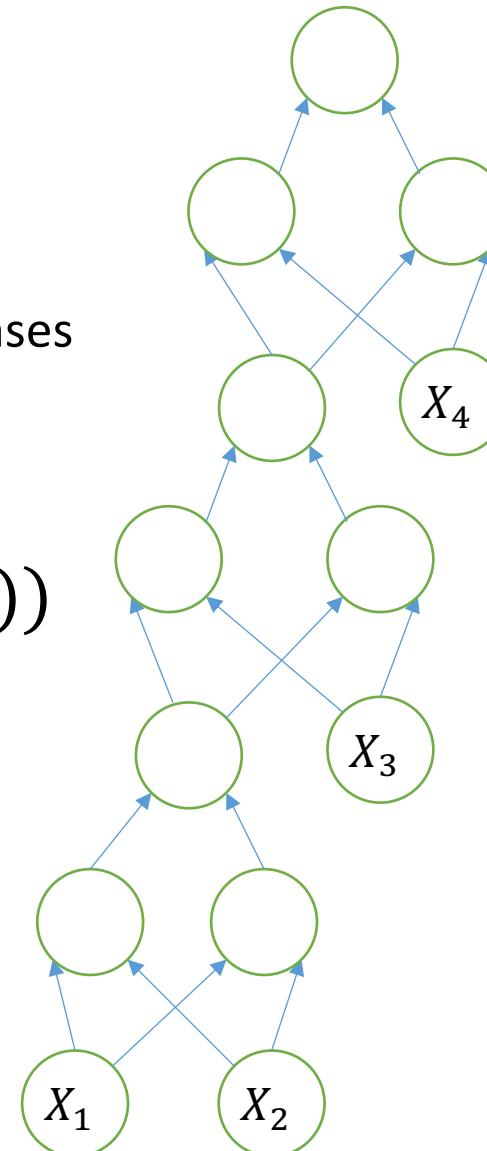
$(D + 2)2^{D-1} + 1$  Weights and biases

- $f = ((X_1 \oplus X_2) \oplus X_2) \dots \oplus X_D)))))$

3( $D - 1$ ) Nodes

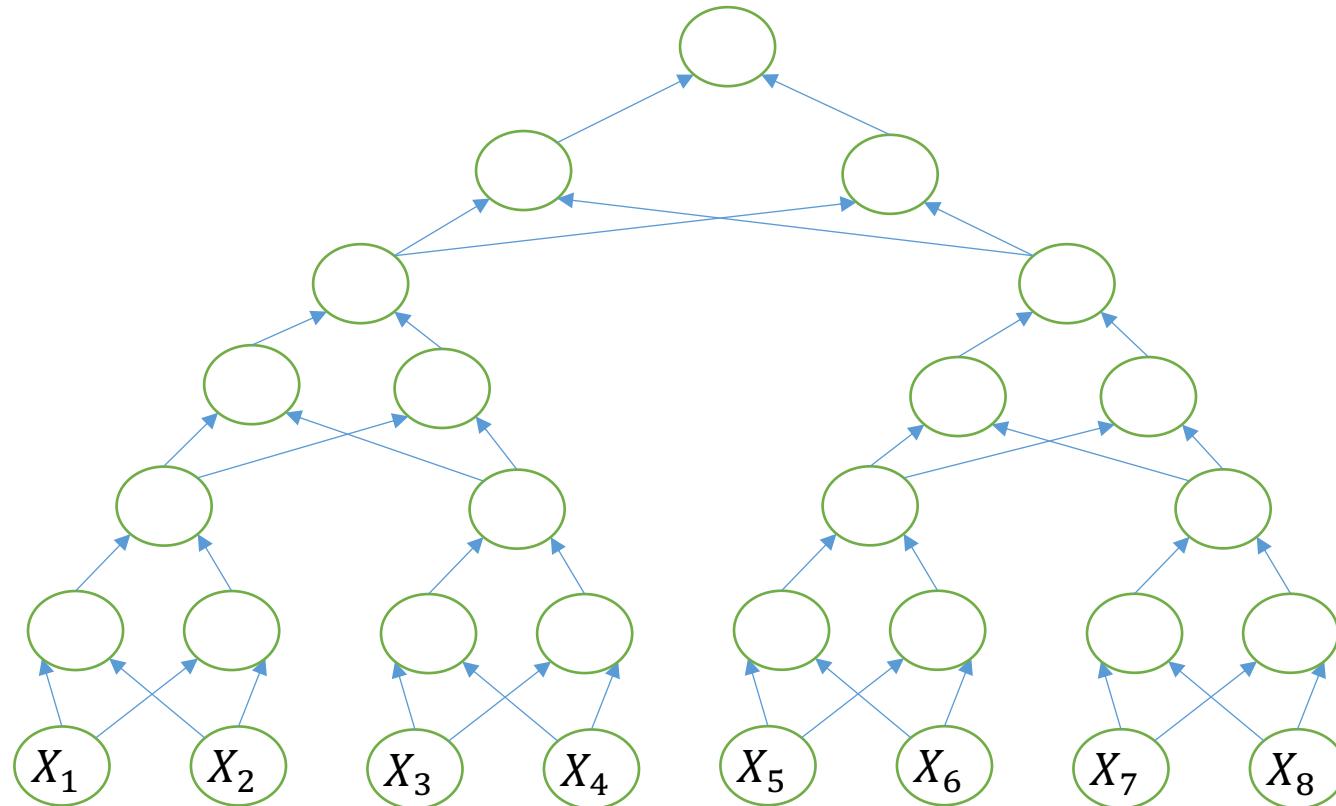
9( $D - 1$ ) Weights and biases

The actual number of parameters in a network is the number that really matters in software or hardware implementations



# A better architecture

- Only requires  $2\log N$  layers
- $f = ((X_1 \oplus X_2) \oplus (X_3 \oplus X_4)) \oplus ((X_4 \oplus X_5) \oplus (X_6 \oplus X_7))$



# Summary: Wide vs. deep network

- MLP with a single hidden layer is a universal Boolean function
- However, a single-layer network might need an exponential number of hidden units w.r.t. the number of inputs
- **Depth efficiency:** Deeper networks may require far fewer neurons than shallower networks to express the same function
  - Could be exponentially smaller
- 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

# Caveat 1: Not all Boolean functions..

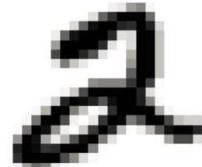
- Not all Boolean circuits have such clear depth-vs-size tradeoff
- Shannon's theorem: For  $D > 2$ , there is Boolean function of  $D$  variables that requires at least  $2^D/D$  gates
  - More correctly, for large  $D$ , almost all  $D$ -input Boolean function need more than  $2^D/D$  gates
    - Regardless of depth
- Note: if all Boolean functions over  $D$  inputs could be computed using a circuit of size that is polynomial in  $D$ ,  $P=NP$  !

## Caveat 2

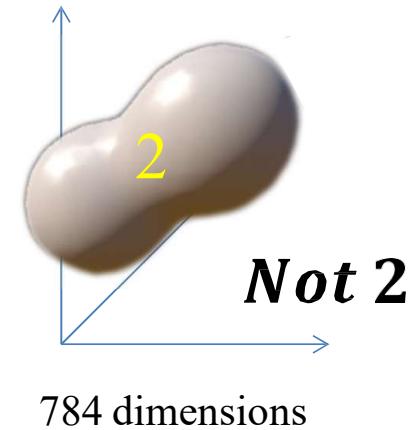
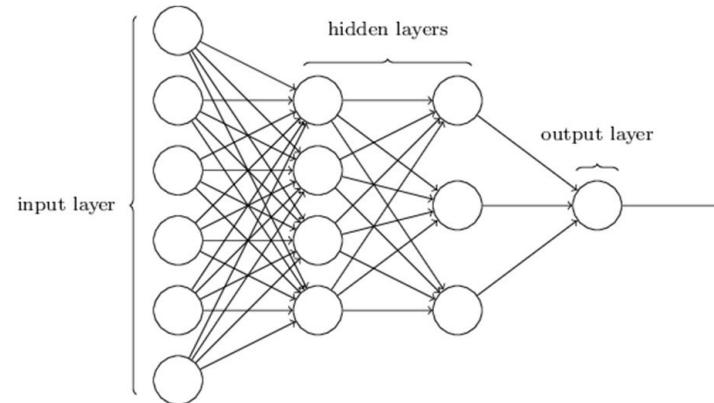
- Used a simple “Boolean circuit” analogy for explanation
- We actually have threshold circuit (TC) not, just a Boolean circuit (AC)
  - Specifically composed of threshold gates
    - More versatile than Boolean gates (can compute majority function)
      - E.g. “at least K inputs are 1” is a single TC gate, but an exponential size AC
      - For fixed depth, *Boolean circuits*  $\subset$  *threshold circuits* (strict subset)
- Other formal analyses typically view NNs as arithmetic circuits
  - Circuits which compute polynomials over any field
- So lets consider functions over the field of reals

# MLPs as universal classifiers

# The MLP as a classifier

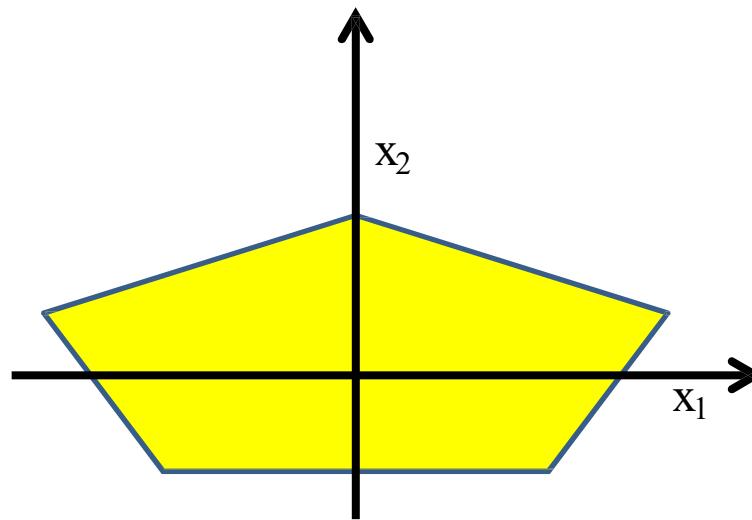


784 dimensions  
(MNIST)



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

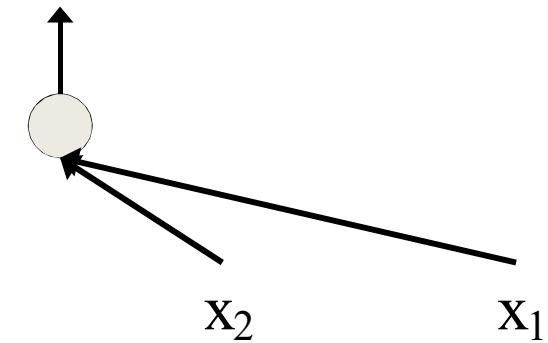
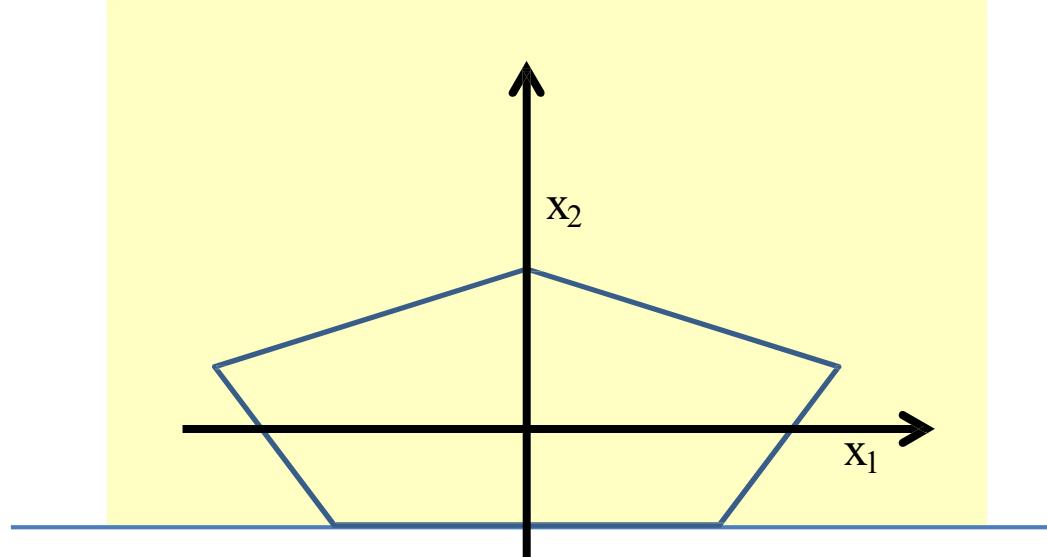
# Composing complicated “decision” boundaries



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

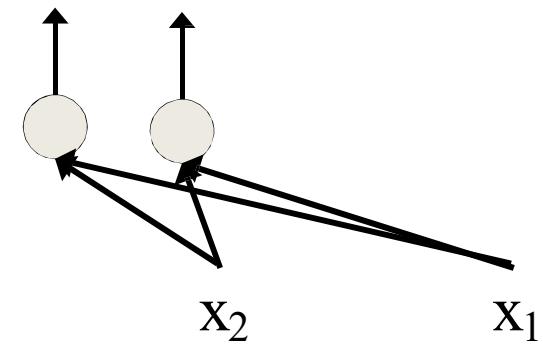
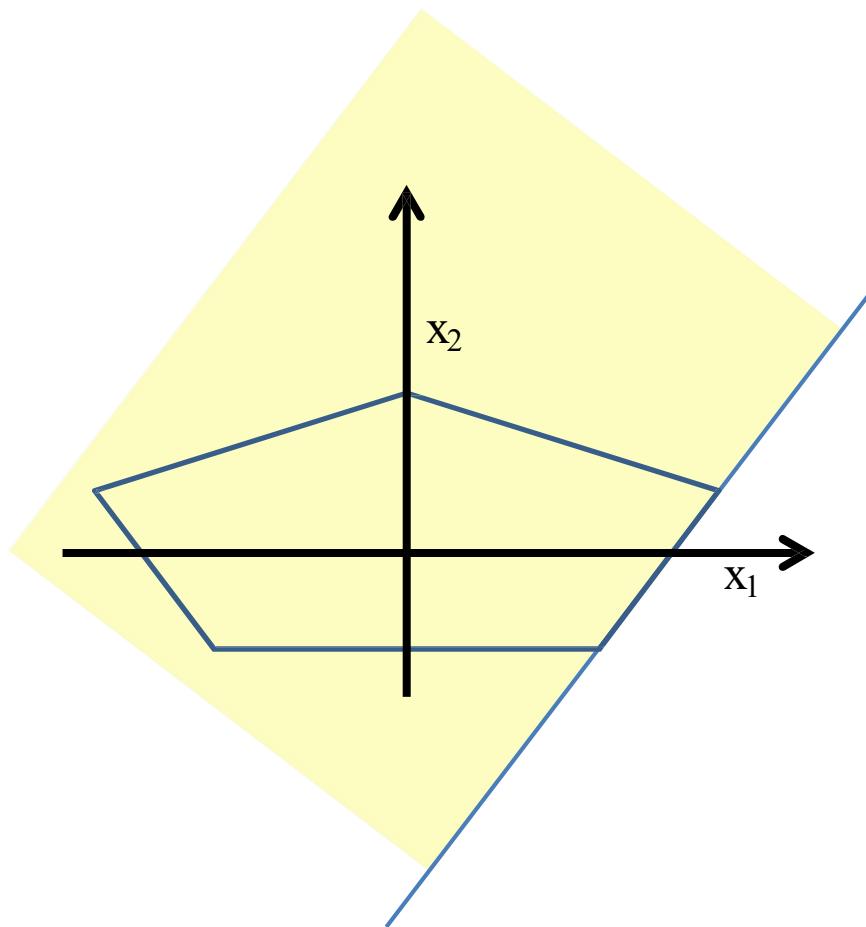
- Build a network of units with a single output that fires if the input is in the coloured area

# Booleans over the reals



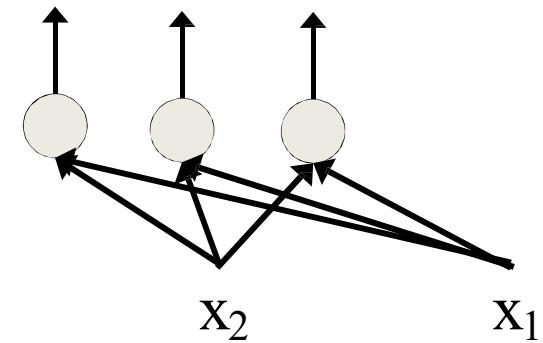
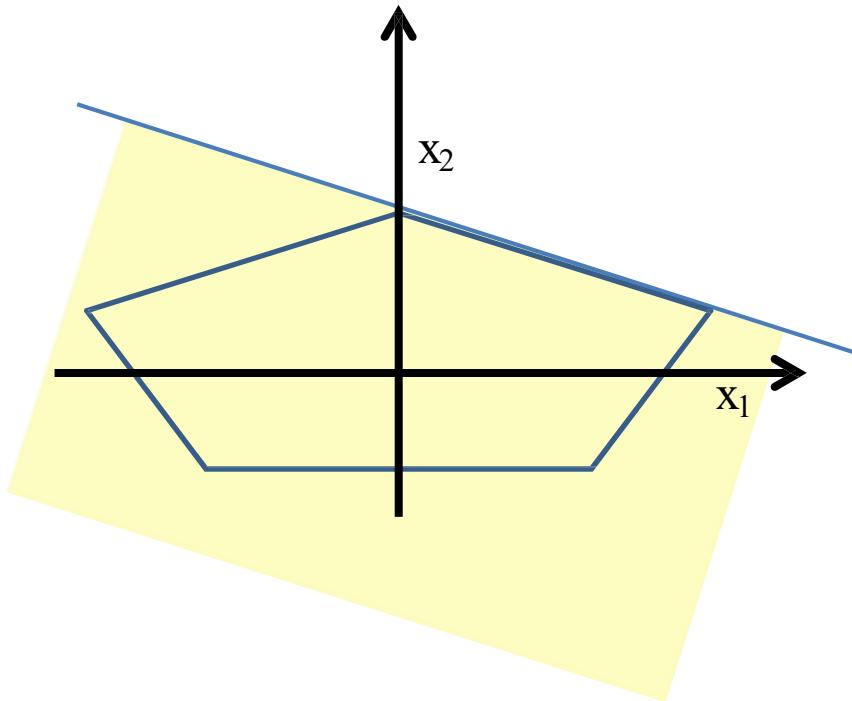
- The network must fire if the input is in the coloured area

# Booleans over the reals



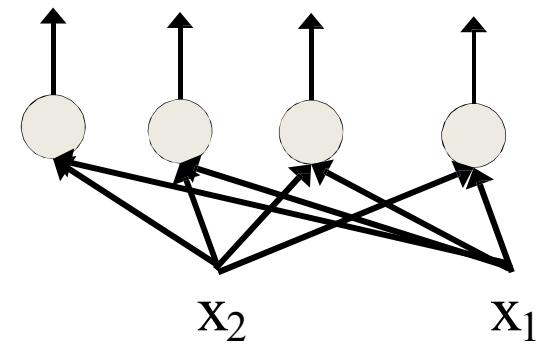
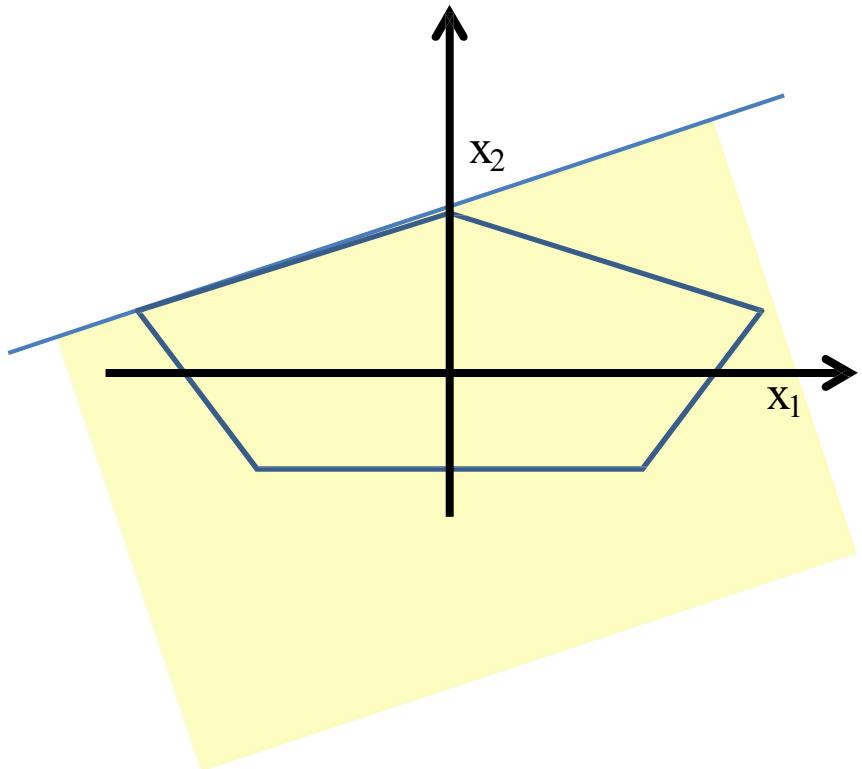
- The network must fire if the input is in the coloured area

# Booleans over the reals



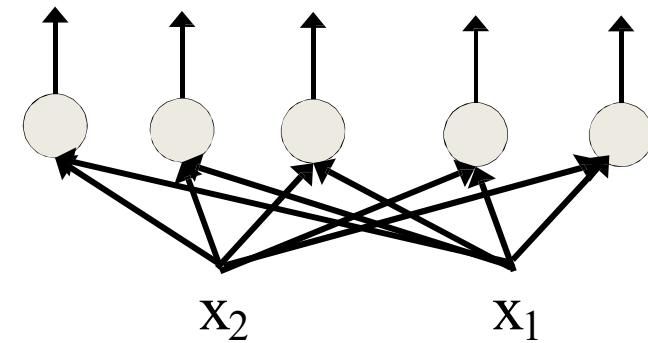
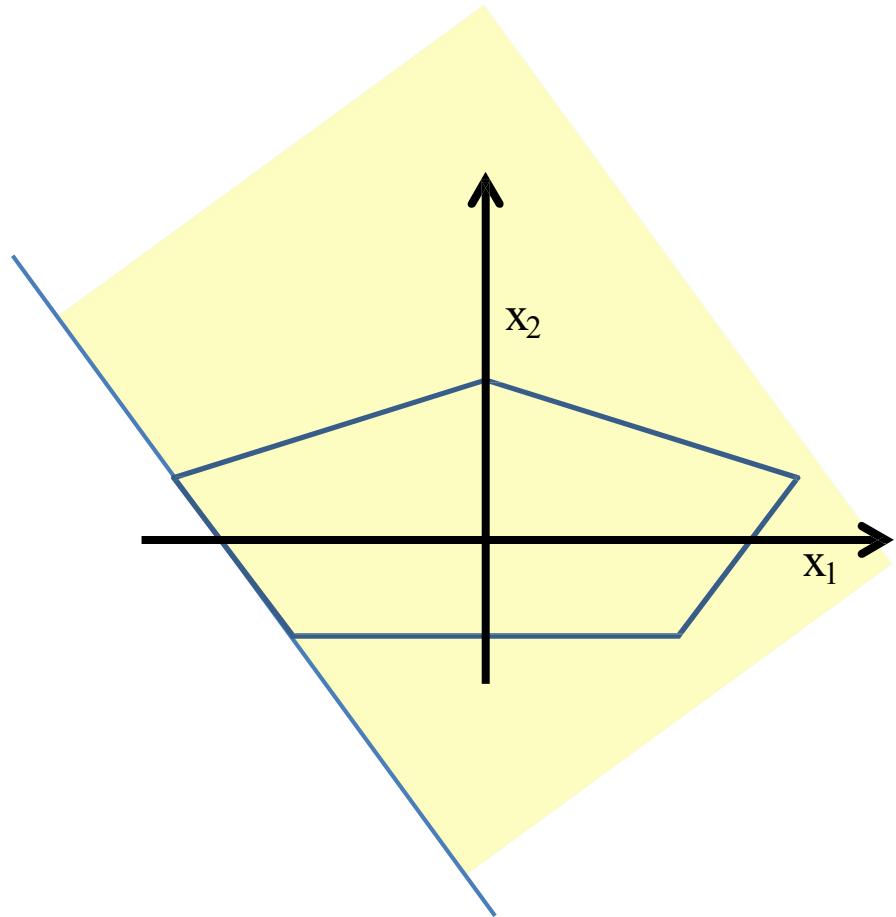
- The network must fire if the input is in the coloured area

# Booleans over the reals



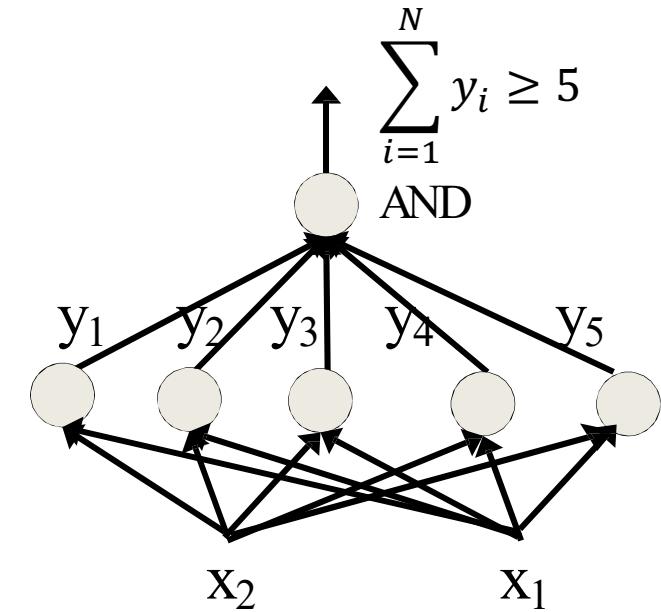
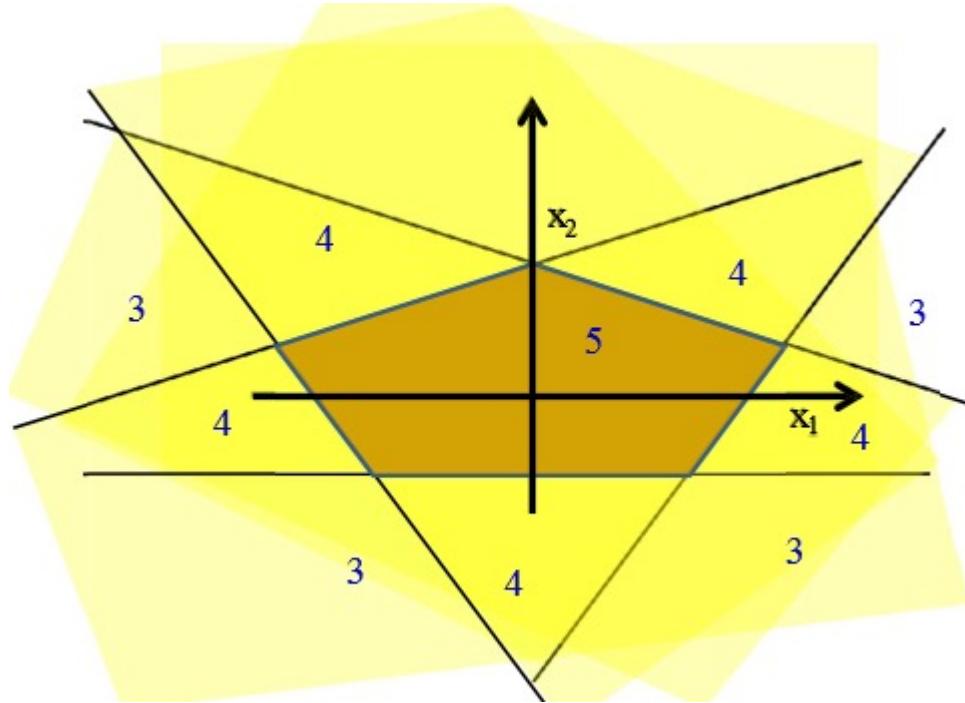
- The network must fire if the input is in the coloured area

# Booleans over the reals



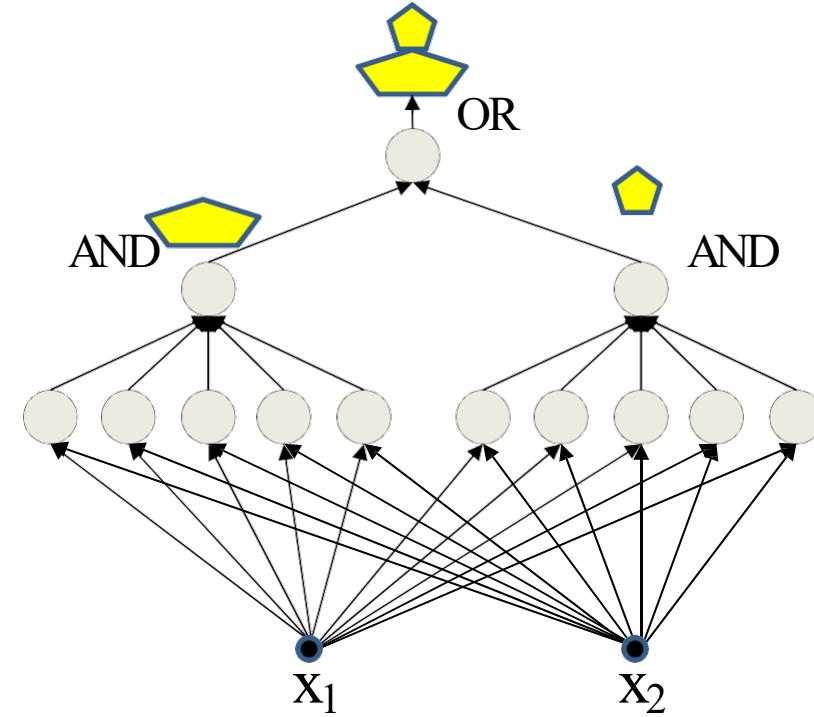
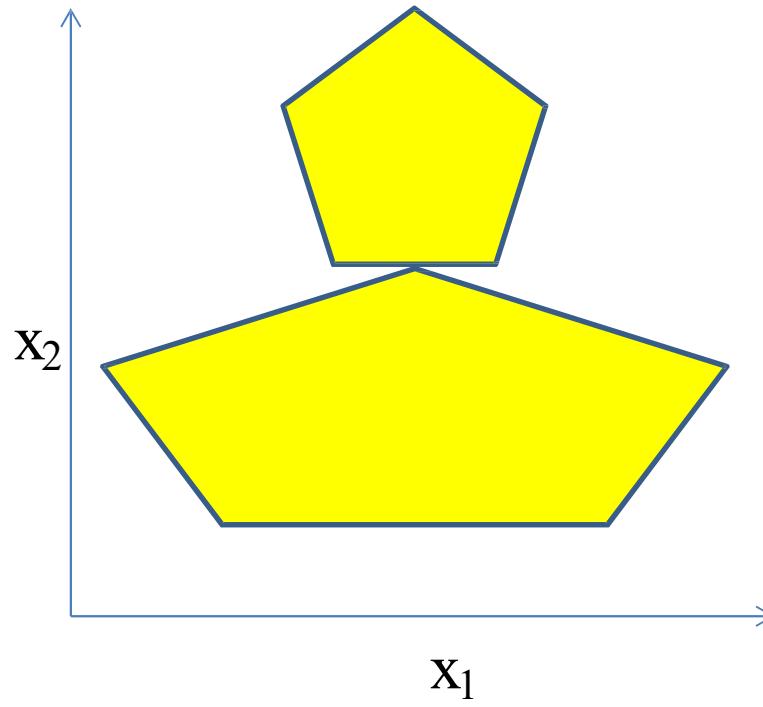
- The network must fire if the input is in the coloured area

# Booleans over the reals



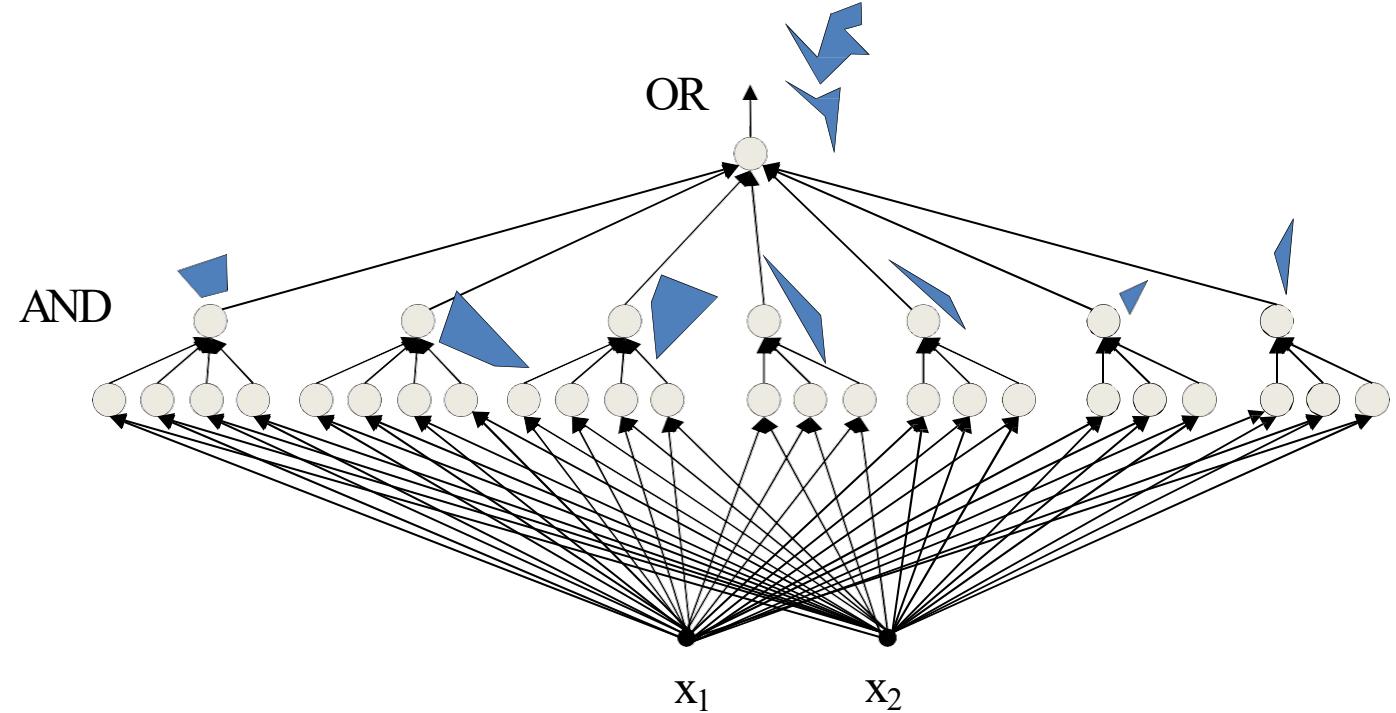
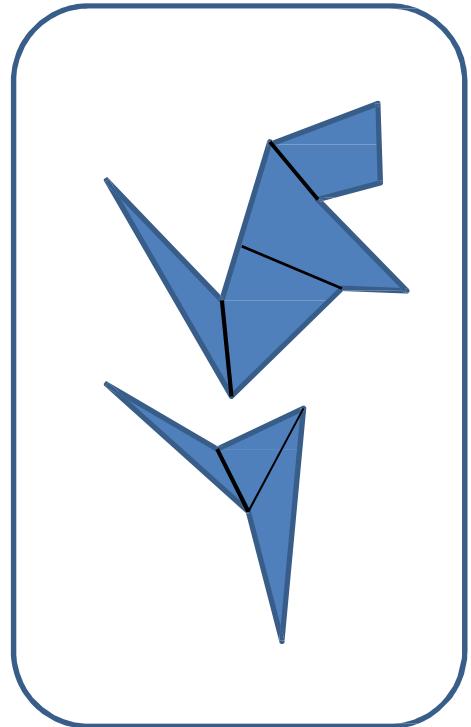
- The network must fire if the input is in the coloured area

# 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

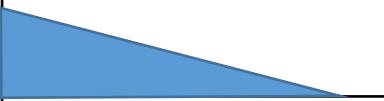


- Can compose arbitrarily complex decision boundaries
  - With only one hidden layer!
  - How?

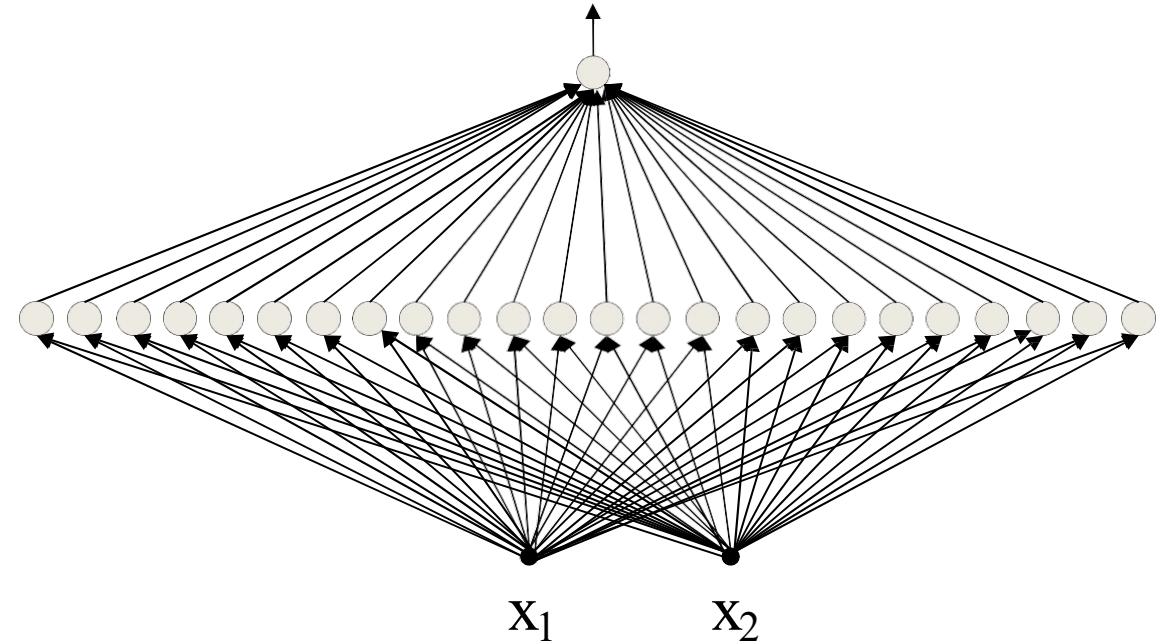
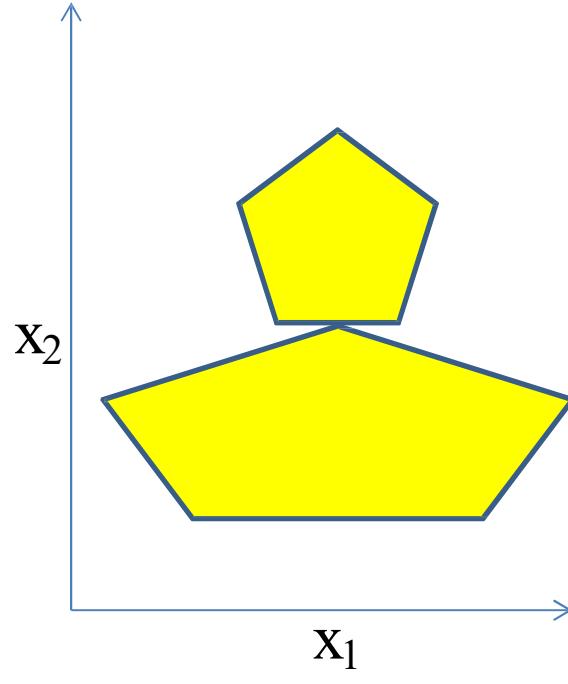
# MLP with Different Number of Layers

MLP with unit step activation function

Decision region found by an output unit.

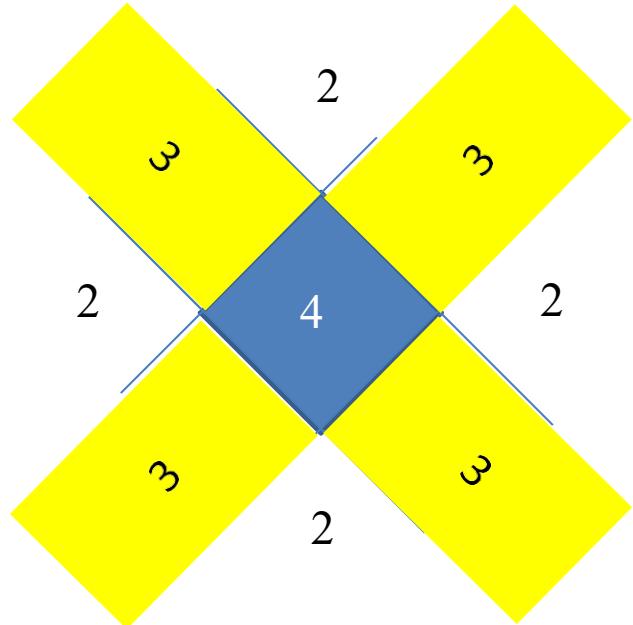
Structure	Type of Decision Regions	Interpretation	Example of region
Single Layer (no hidden layer)	Half space	Region found by a hyper-plane	
Two Layer (one hidden layer)	Polyhedral (open or closed) region	Intersection of half spaces	
Three Layer (two hidden layers)	Arbitrary regions	Union of polyhedrals	

# Exercise: compose this with one hidden layer

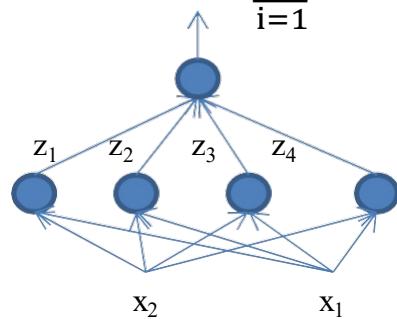


- How would you compose the decision boundary to the left with only one hidden layer?

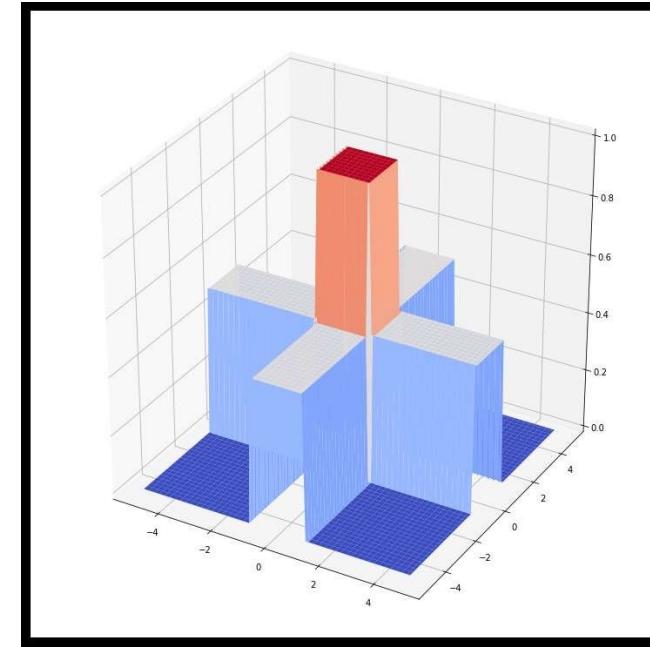
# Composing a square decision boundary



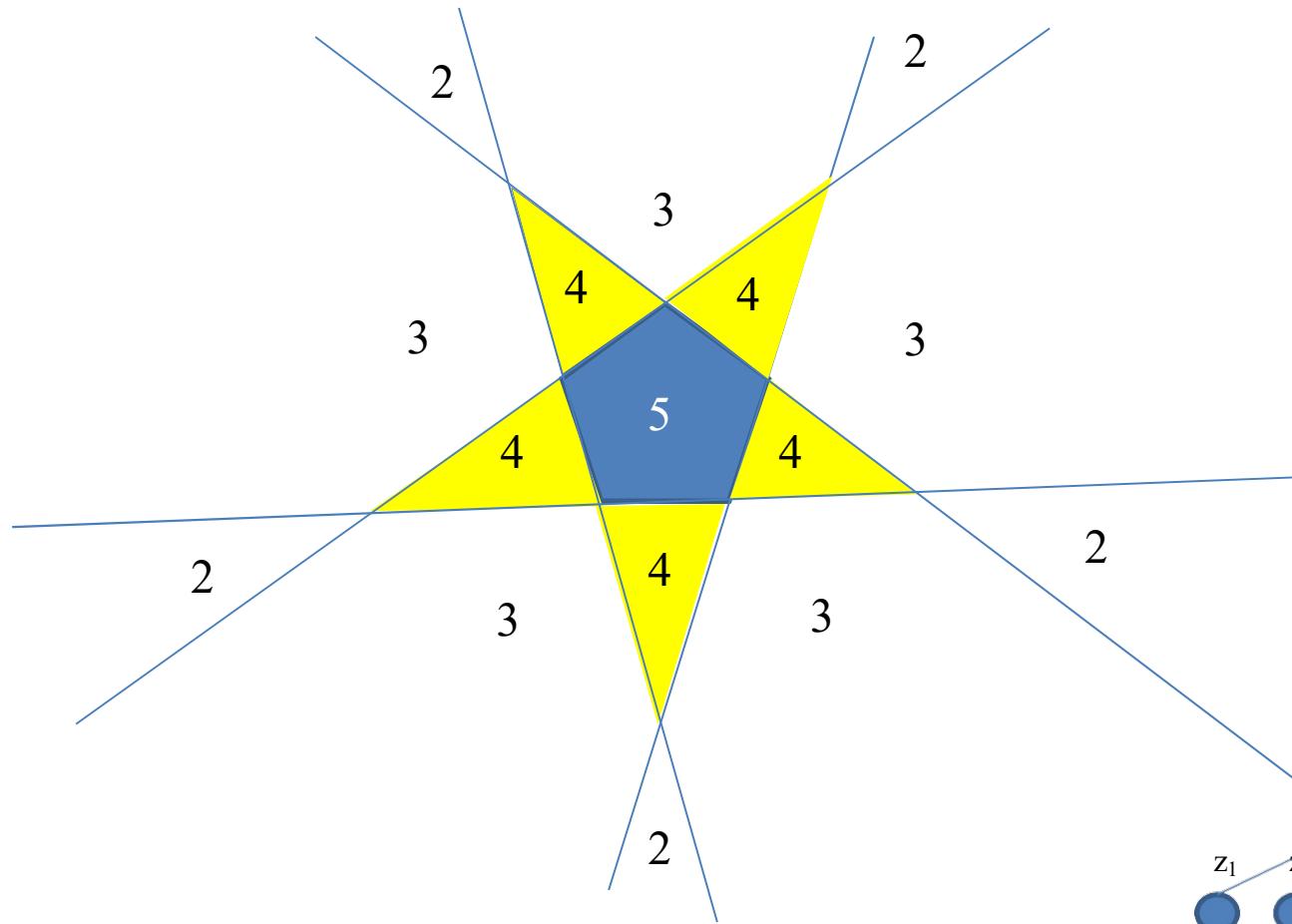
$$\sum_{i=1}^4 z_i \geq 4 ?$$



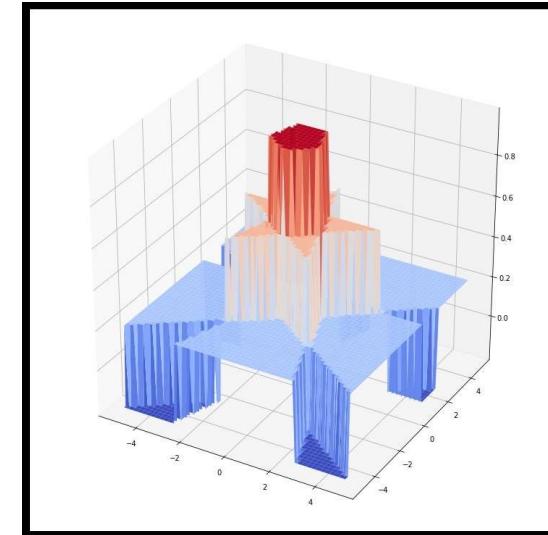
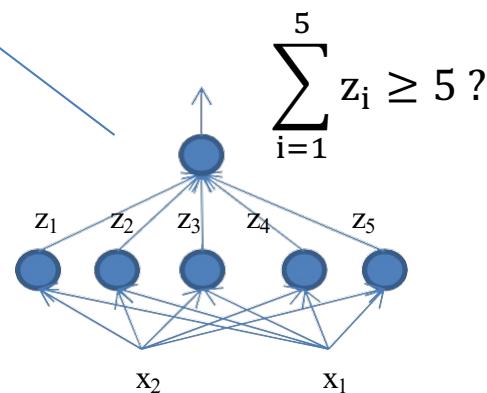
- The polygon net



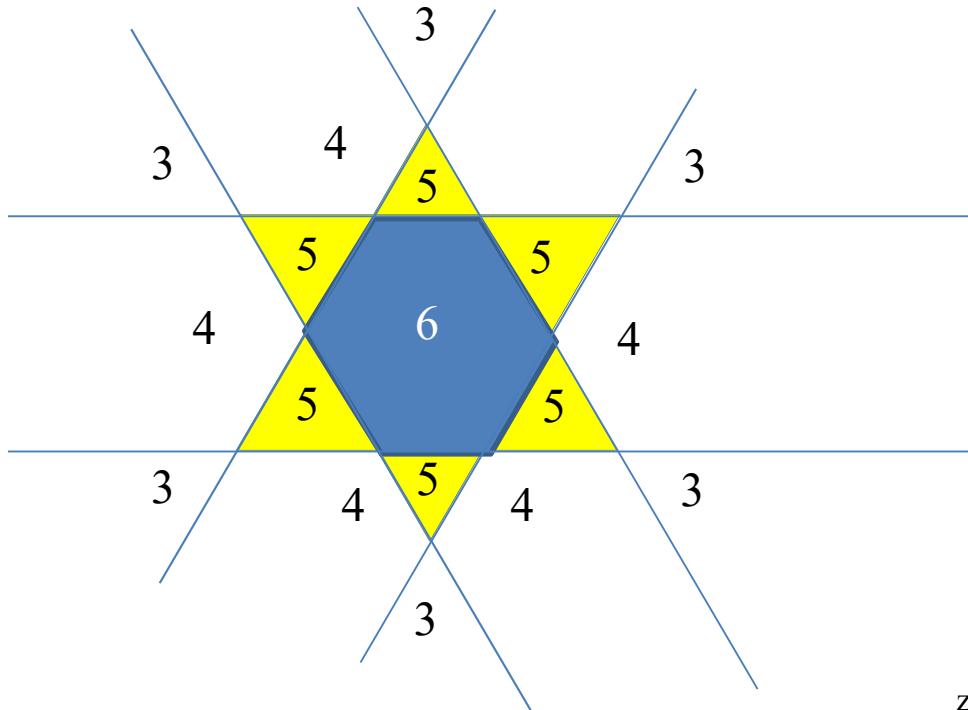
# Composing a square decision boundary



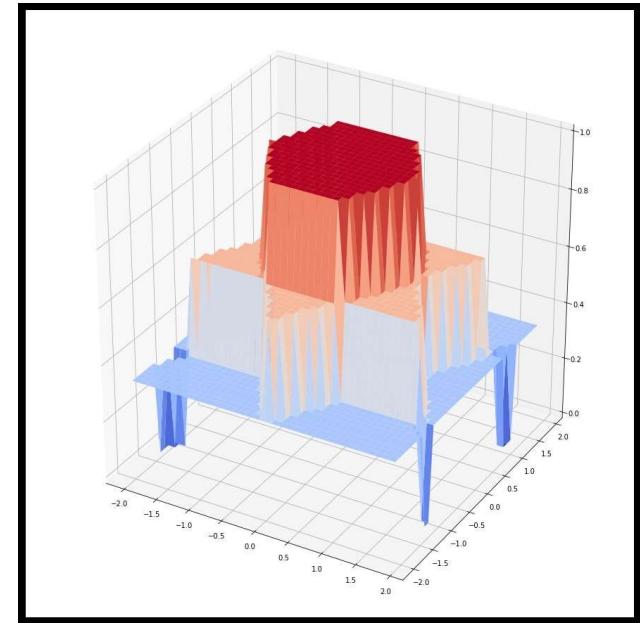
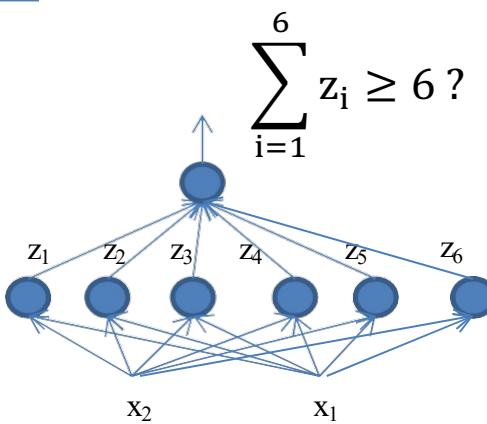
- The polygon net



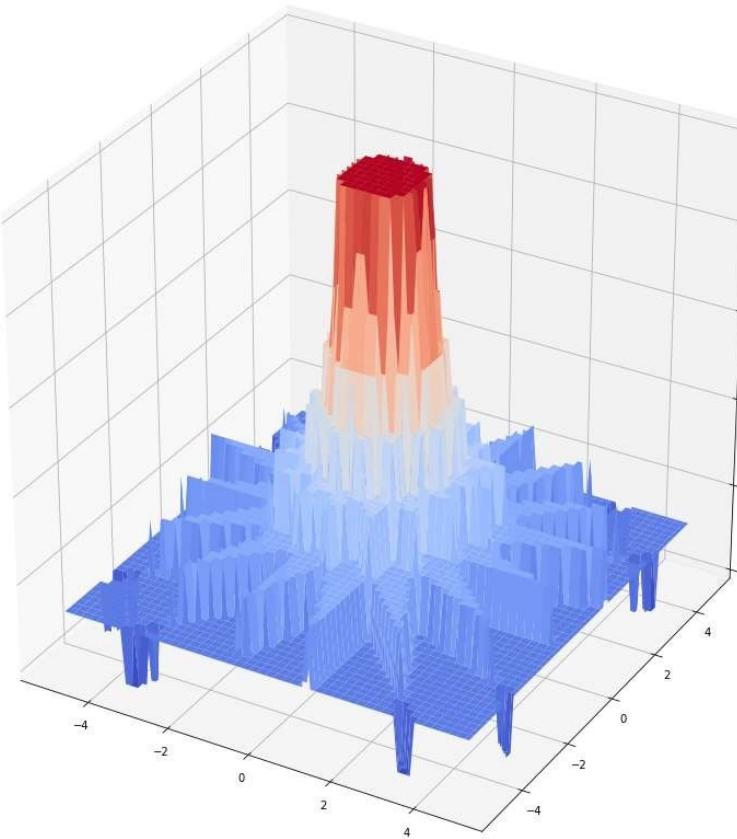
# Composing a pentagon



- The polygon net

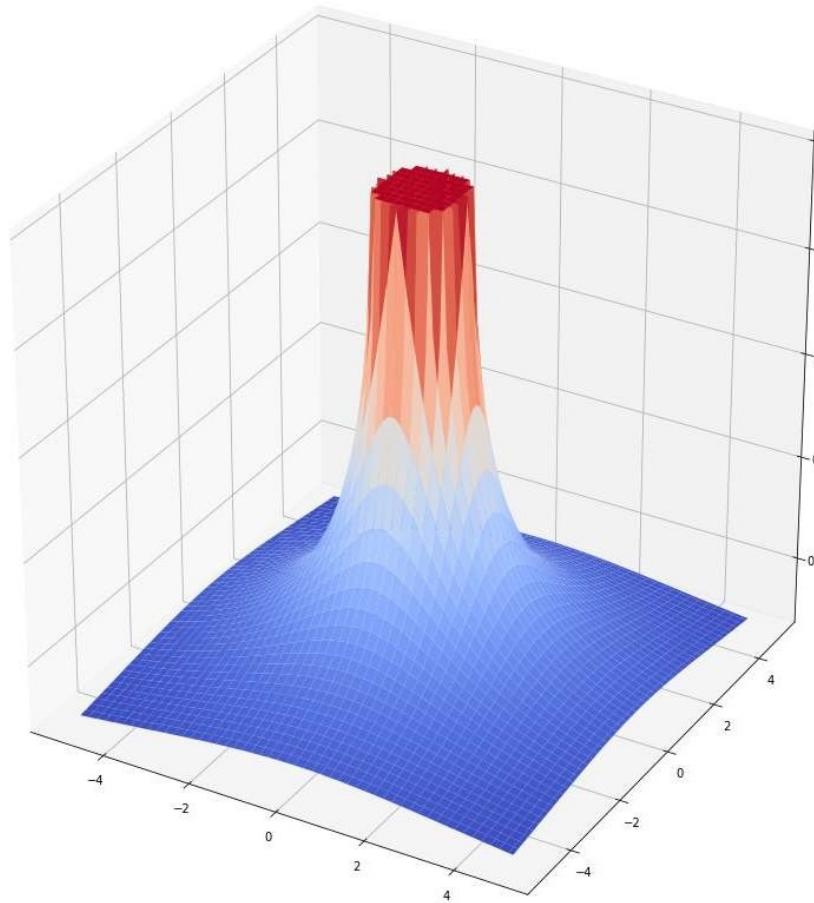


# 16 sides



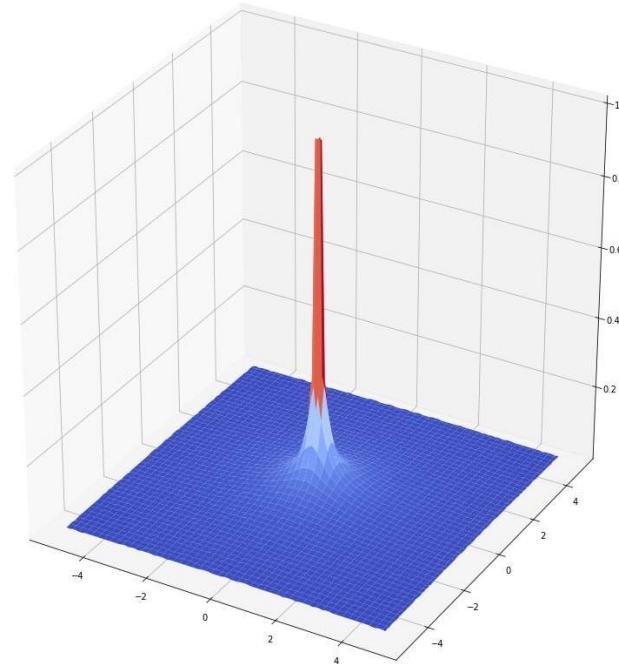
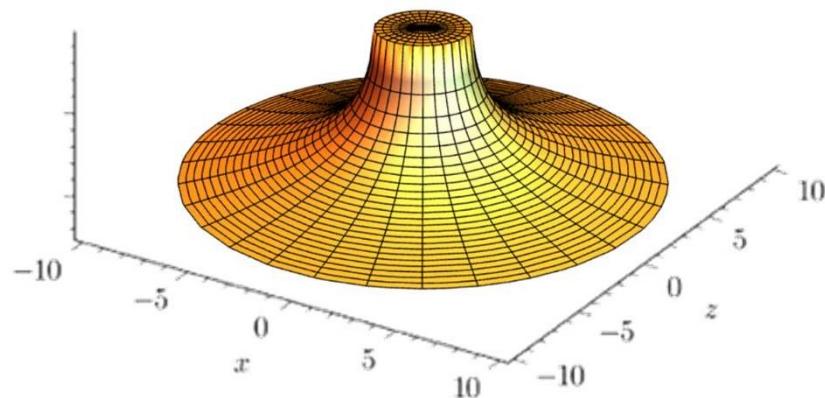
- What are the sums in the different regions?

# Polygon net



- Increasing the number of sides  $M$  reduces the area outside the polygon that have  $M/2 < \text{Sum} < M$

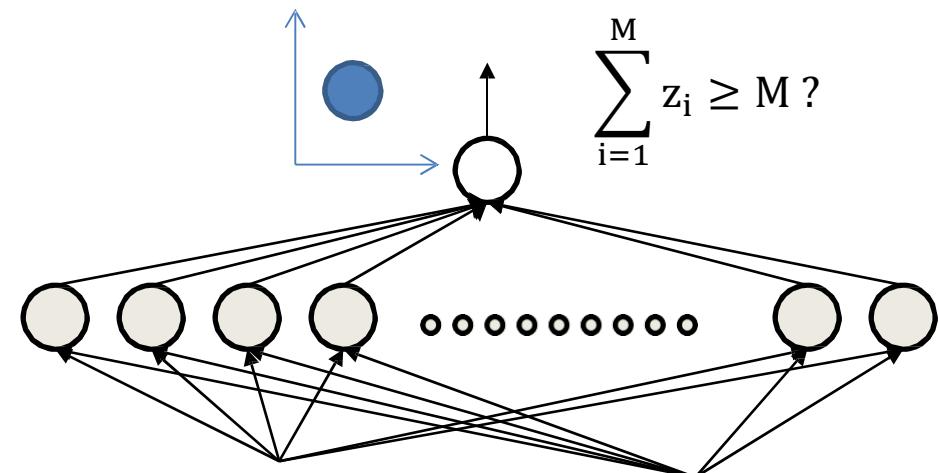
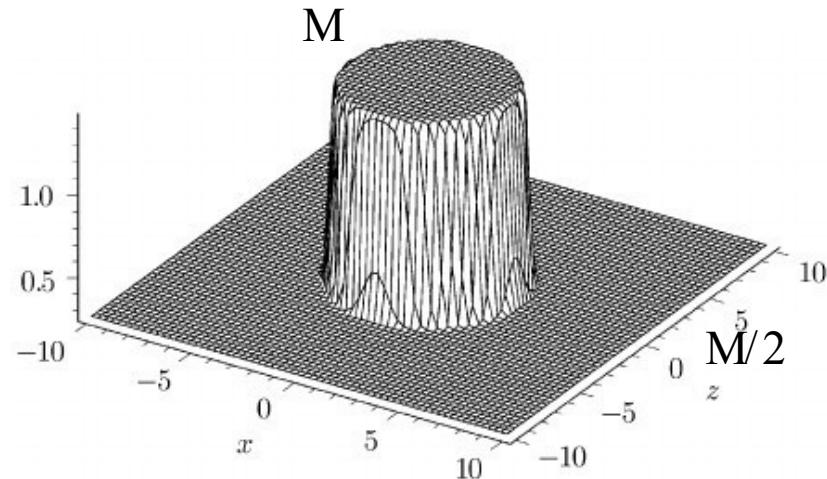
# In the limit



$$\sum_i z_i = M \left( 1 - \frac{1}{\pi} \arccos \left( \min \left( 1, \frac{\text{radius}}{|x - cent|} \right) \right) \right)$$

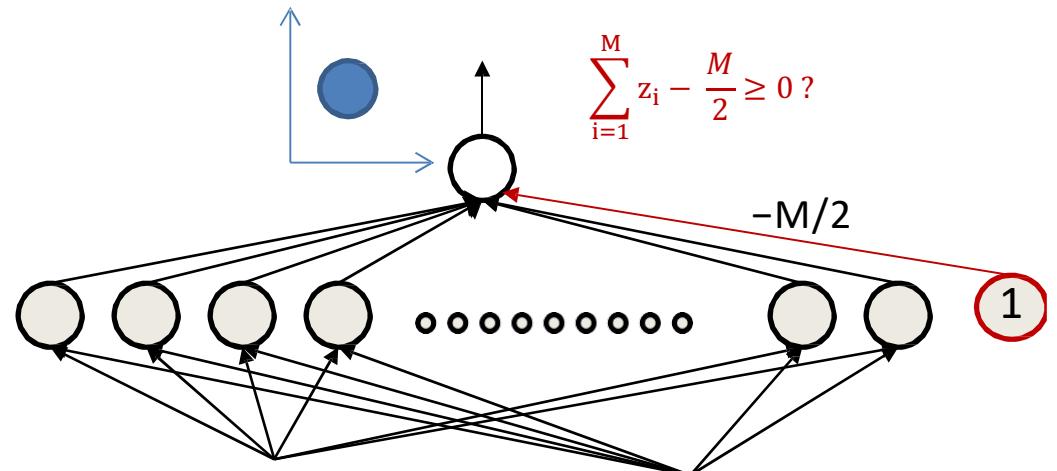
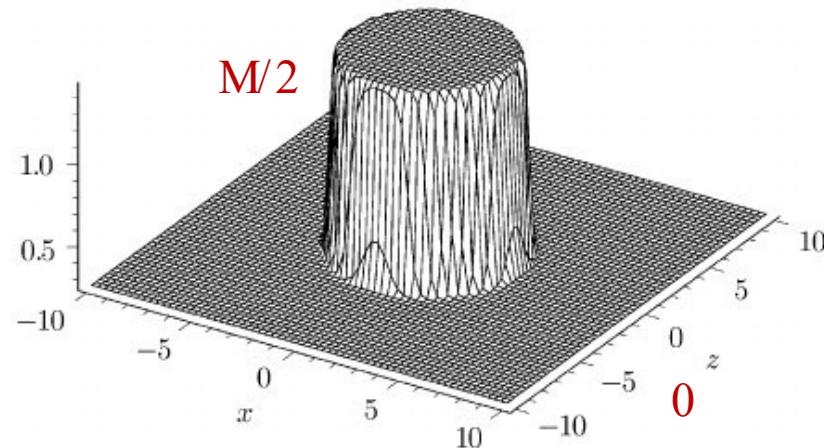
- For small radius, it's a near perfect cylinder
  - $M$  in the cylinder,  $M/2$  outside

# Composing a circle



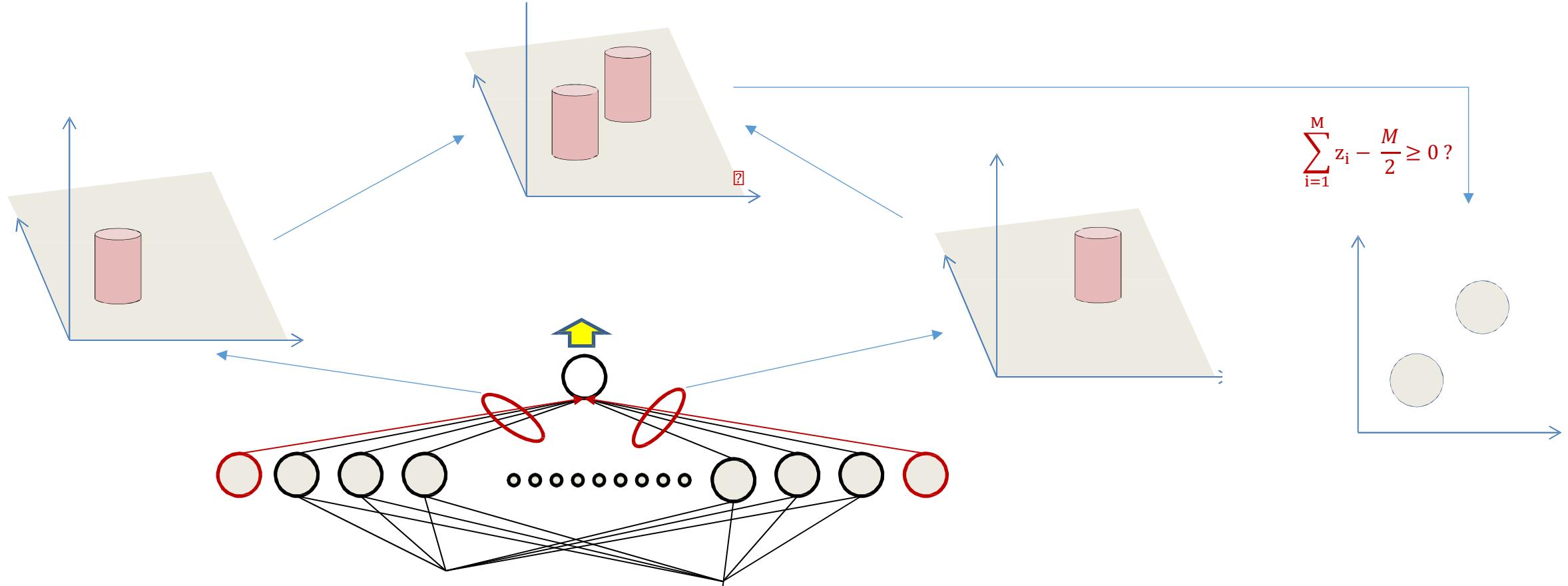
- The circle net
  - Very large number of neurons
  - Sum is  $M$  inside the circle,  $M/2$  outside almost everywhere
  - Circle can be at any location

# Composing a circle



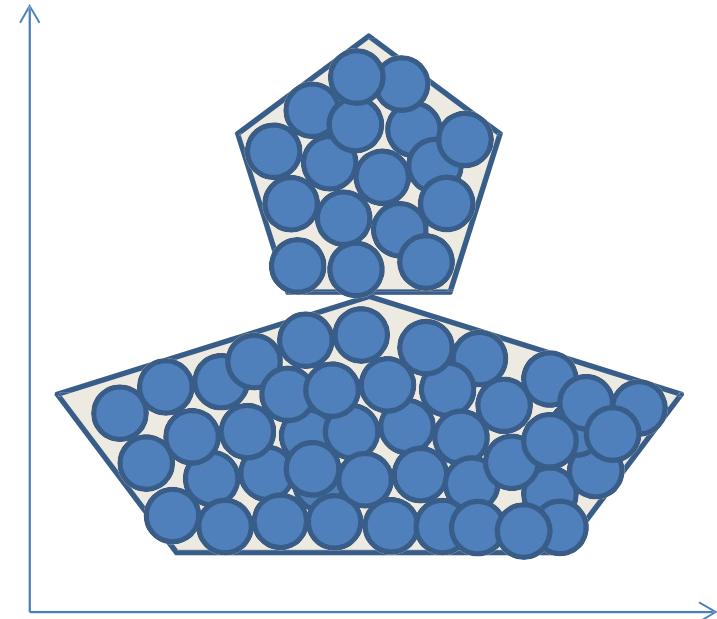
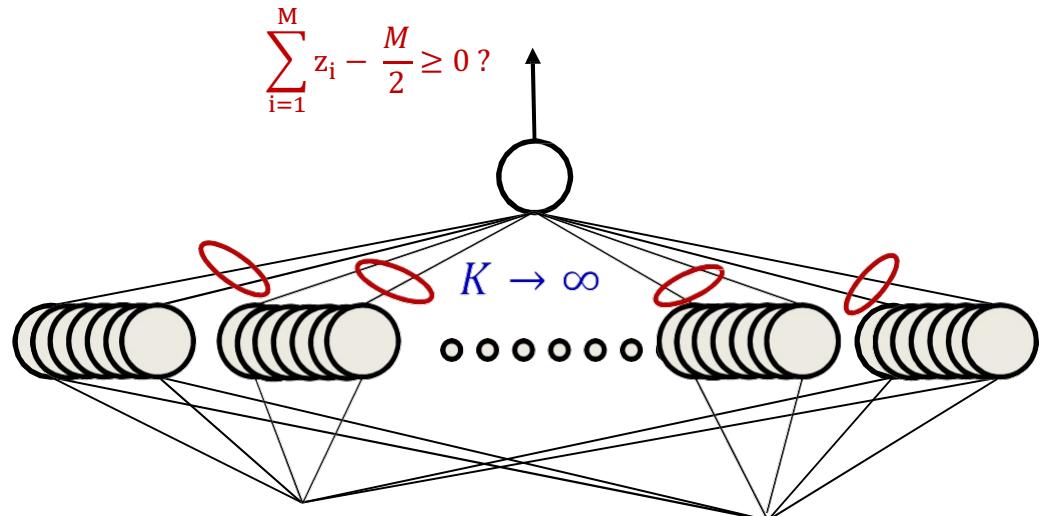
- The circle net
  - Very large number of neurons
  - Sum is  $M/2$  inside the circle,  $0$  outside almost everywhere
  - Circle can be at any location

# Adding circles



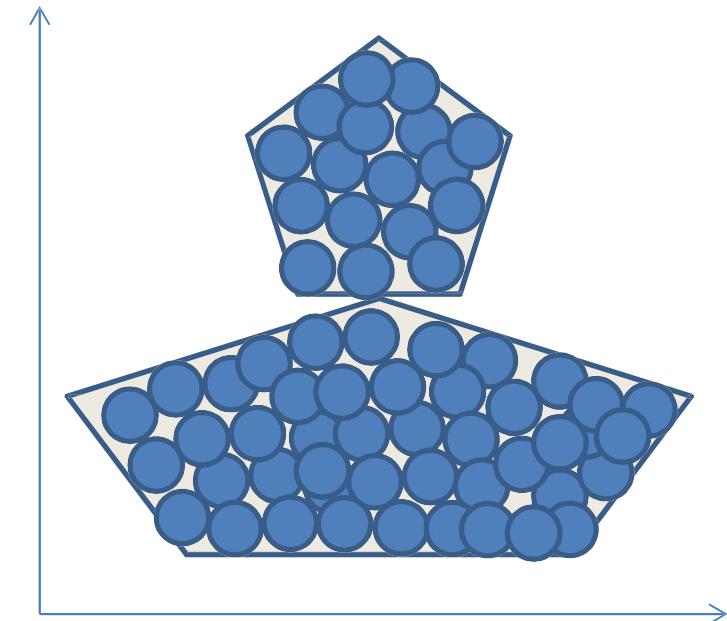
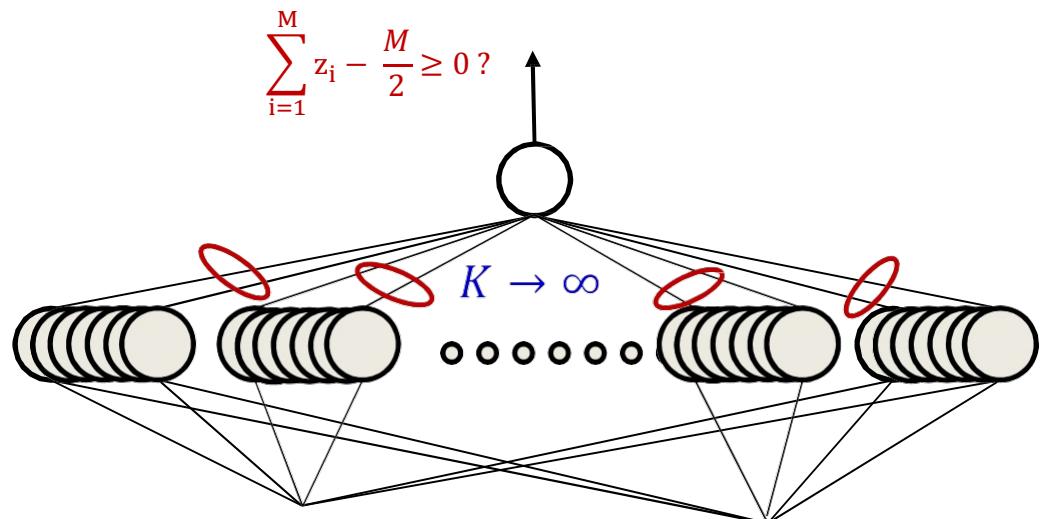
- The “sum” of two circles sub nets is exactly  $M/2$  inside either circle, and 0 almost everywhere outside

# Composing an arbitrary figure



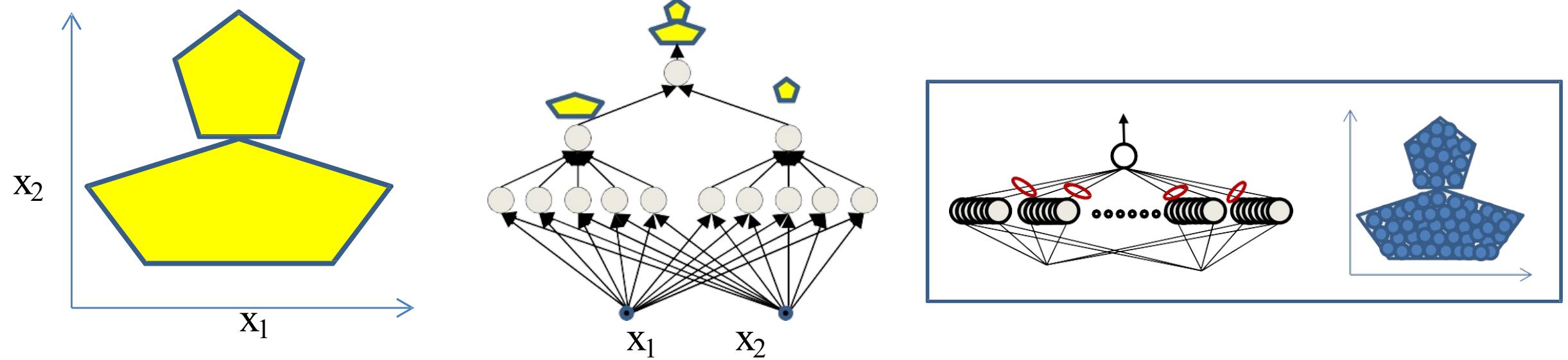
- Just fit in an arbitrary number of circles
  - More accurate approximation with greater number of smaller circles
  - Can achieve arbitrary precision

# MLP: Universal classifier



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

# Depth and the universal classifier

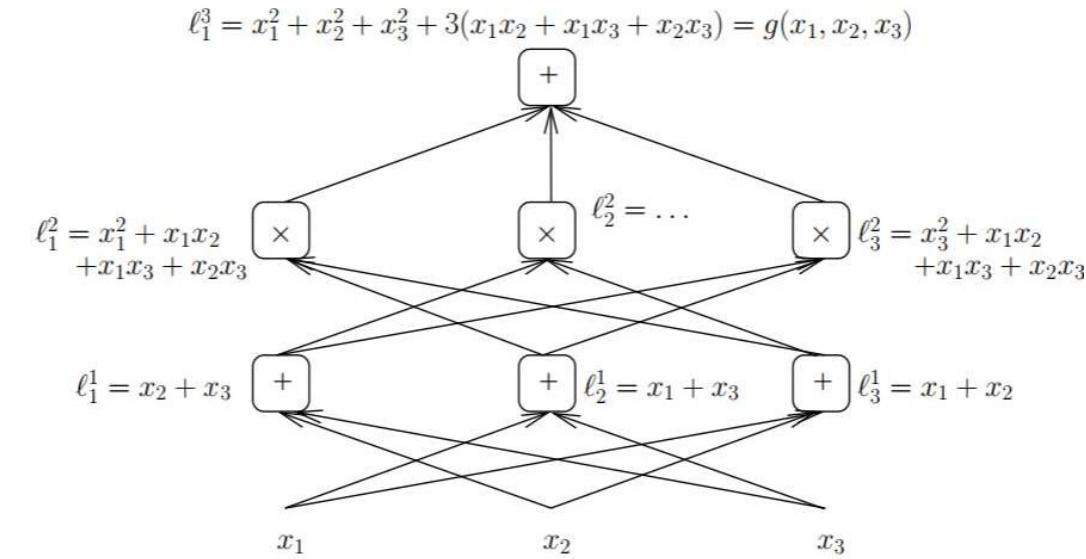


- Deeper networks can require far fewer neurons

# 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  $\infty$ ) 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

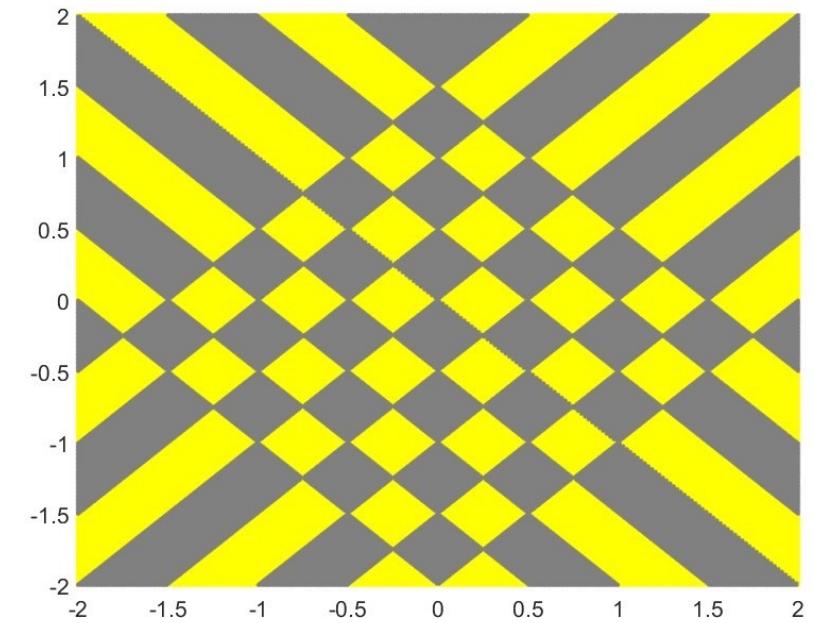
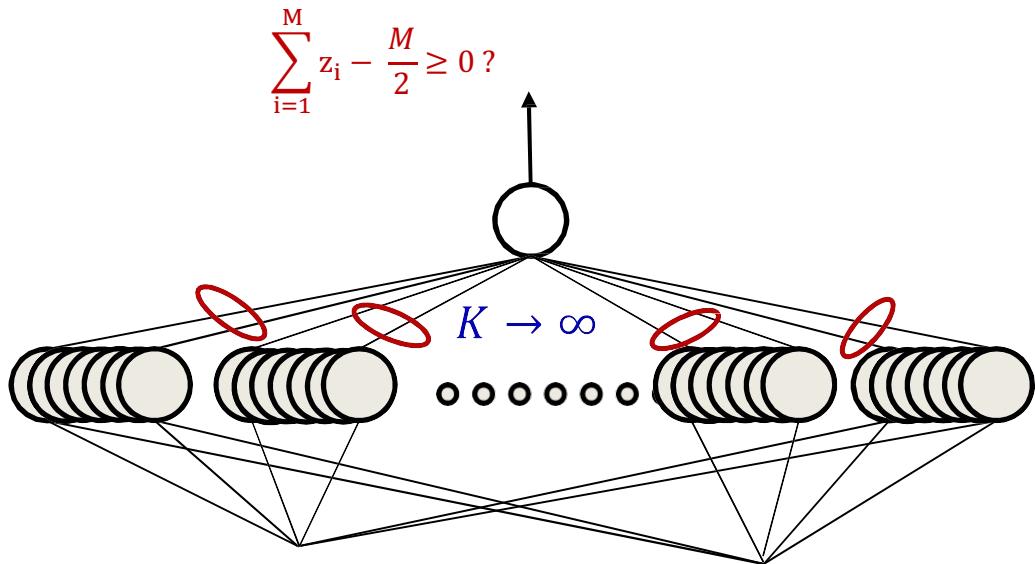


- “Shallow vs deep sum-product networks,” O. Dellaleau and Y. Bengio
  - For networks where layers alternately perform either sums or products, a **deep network may require an exponentially fewer number of layers than a shallow one**

# Optimal depth in generic nets

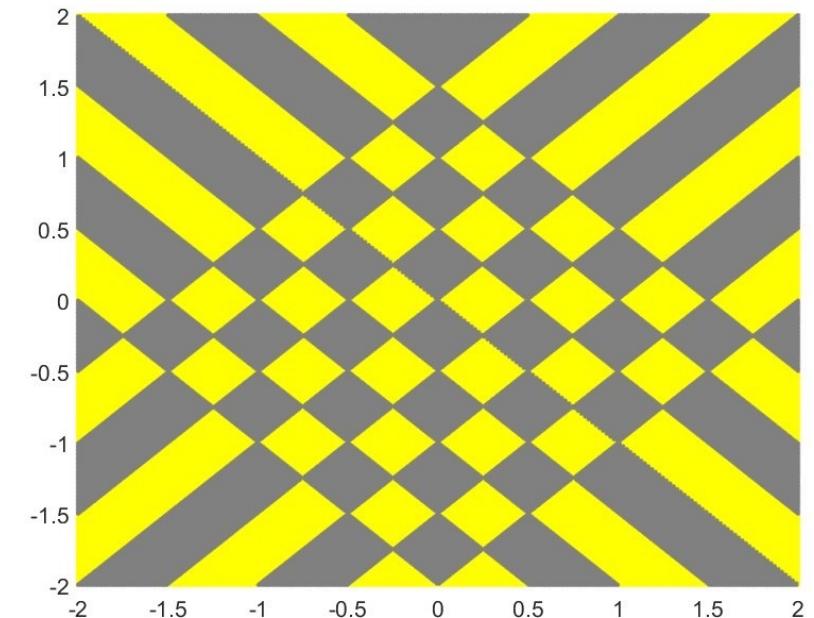
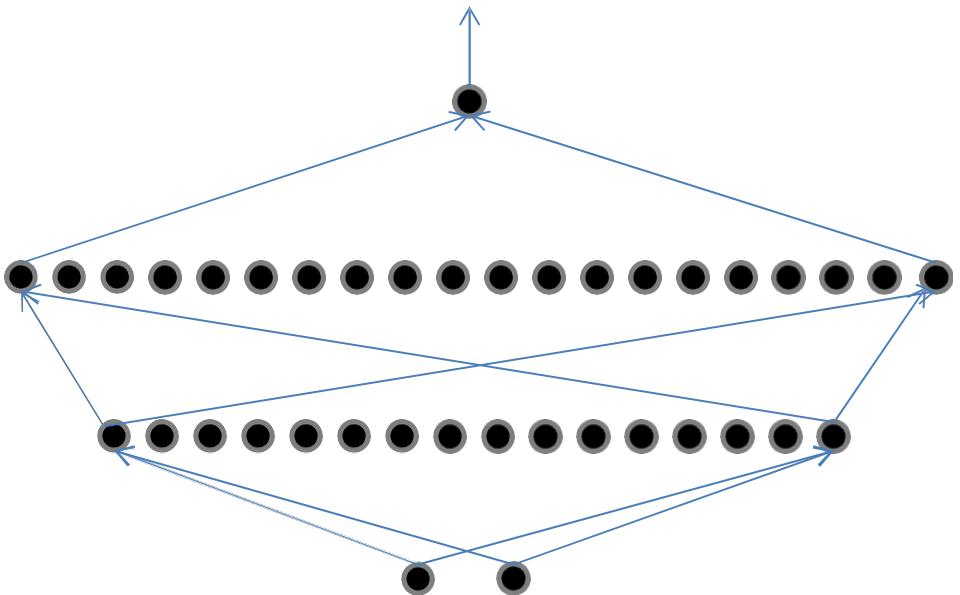
- We look at a different pattern:
  - “worst case” decision boundaries
- For threshold-activation networks
  - Generalizes to other nets

# Optimal depth



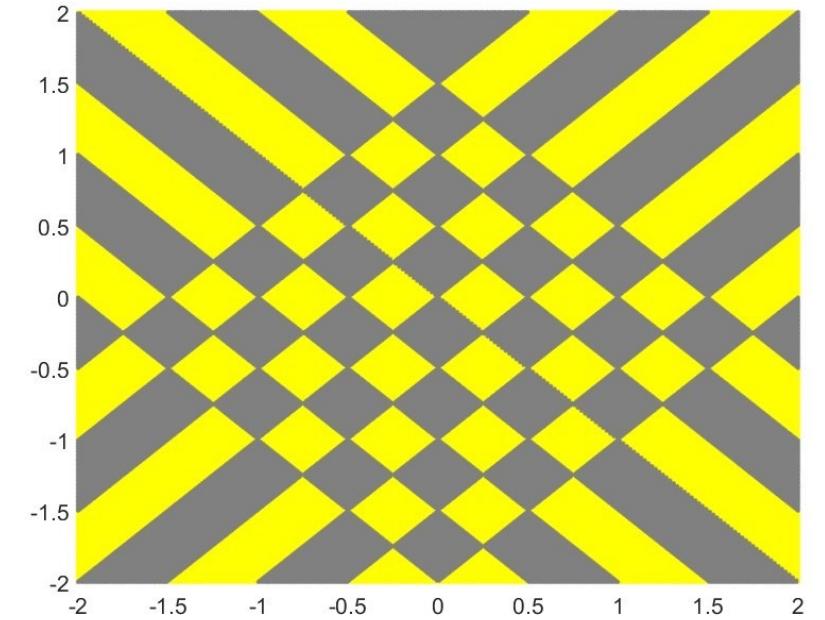
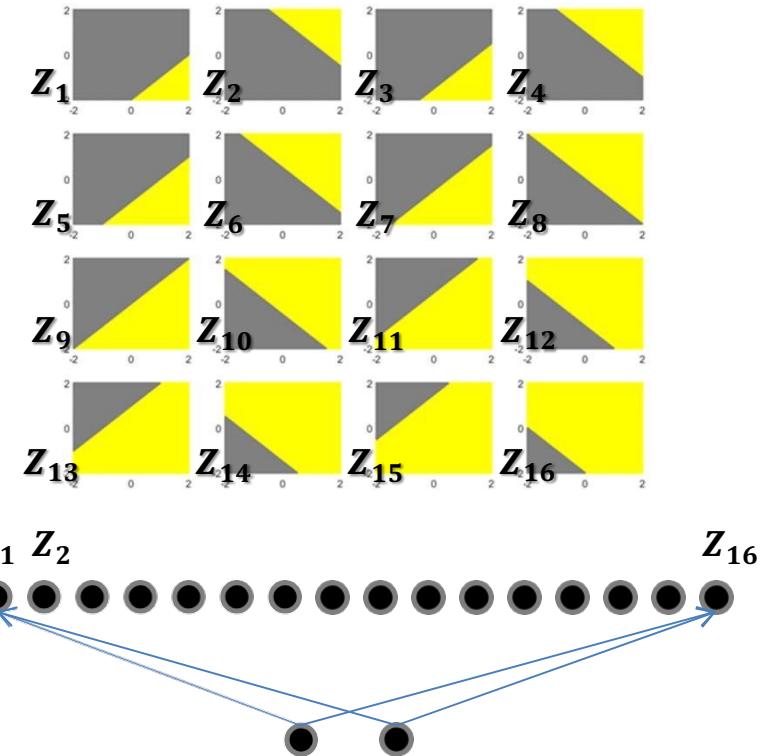
- A naïve one-hidden-layer neural network will required infinite hidden neurons

# Optimal depth



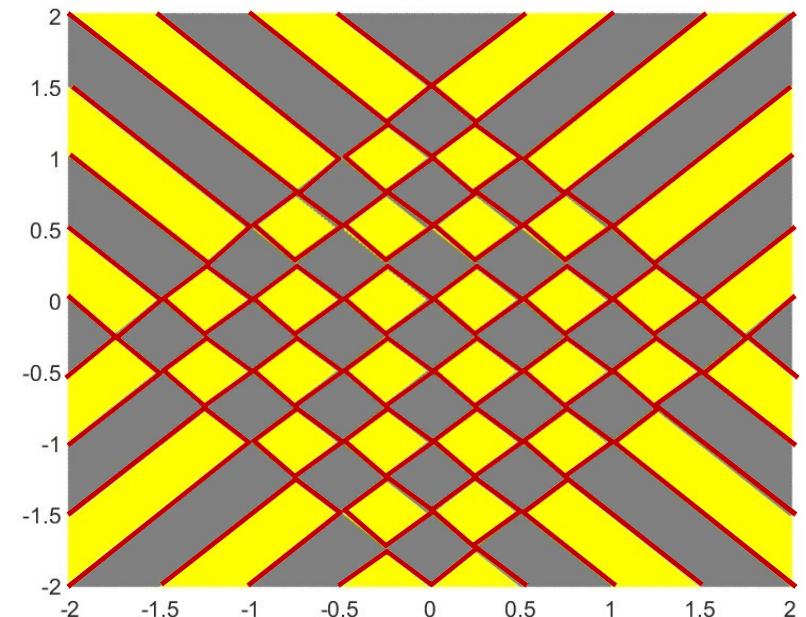
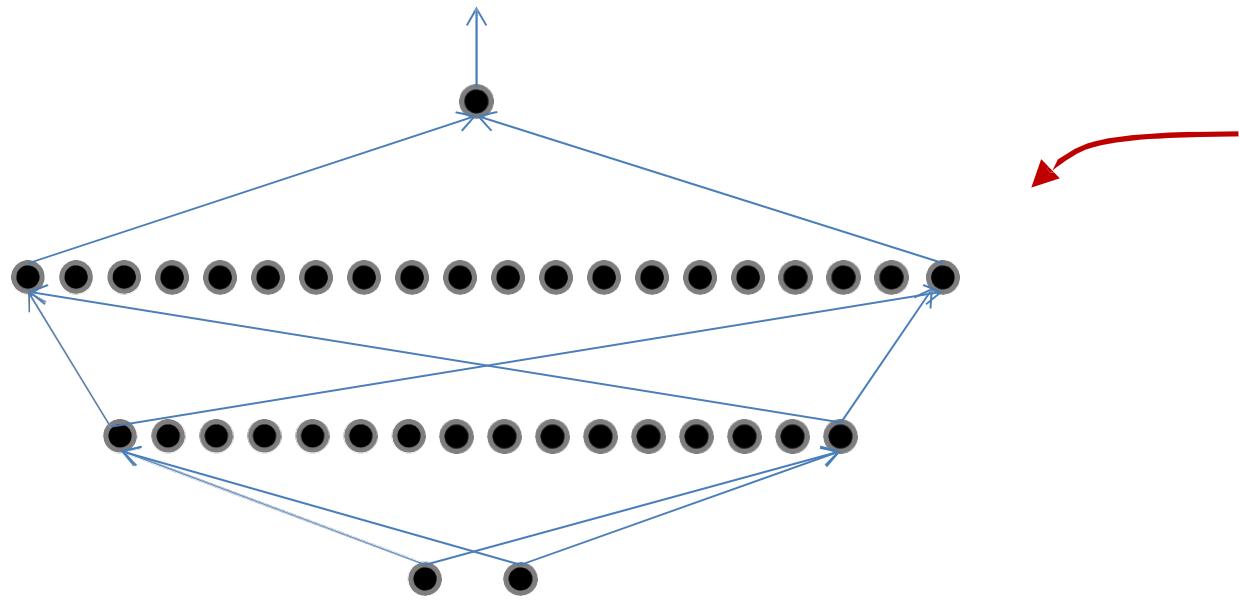
- Two hidden-layer network: 56 hidden neurons

# Optimal depth



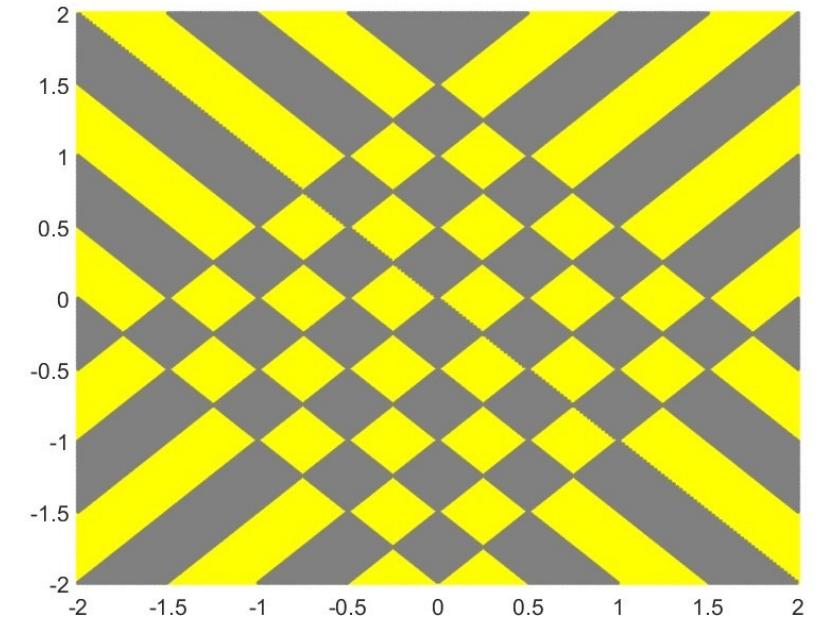
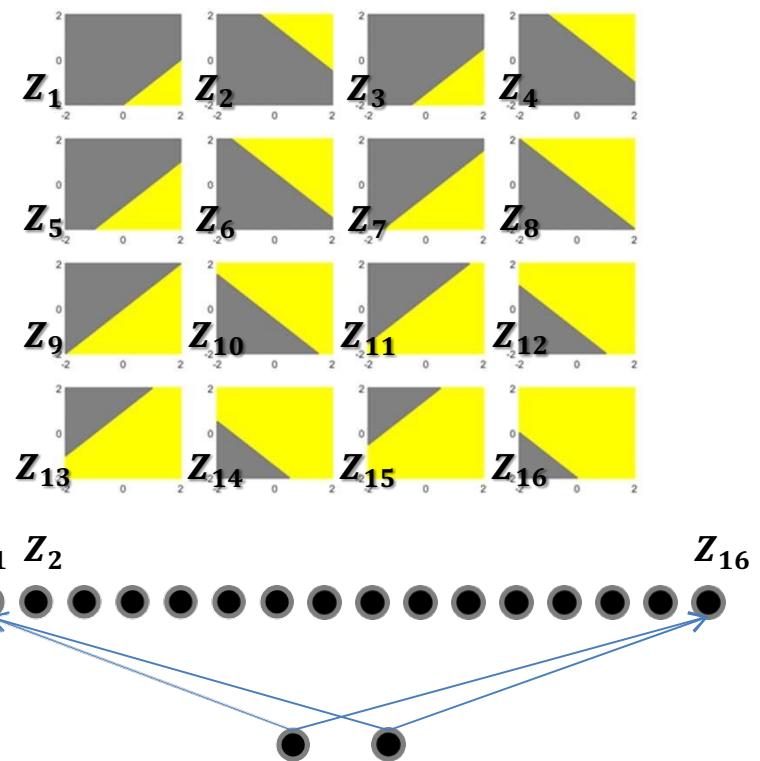
- Two layer network: 56 hidden neurons
  - 16 neurons in hidden layer 1

# Optimal depth



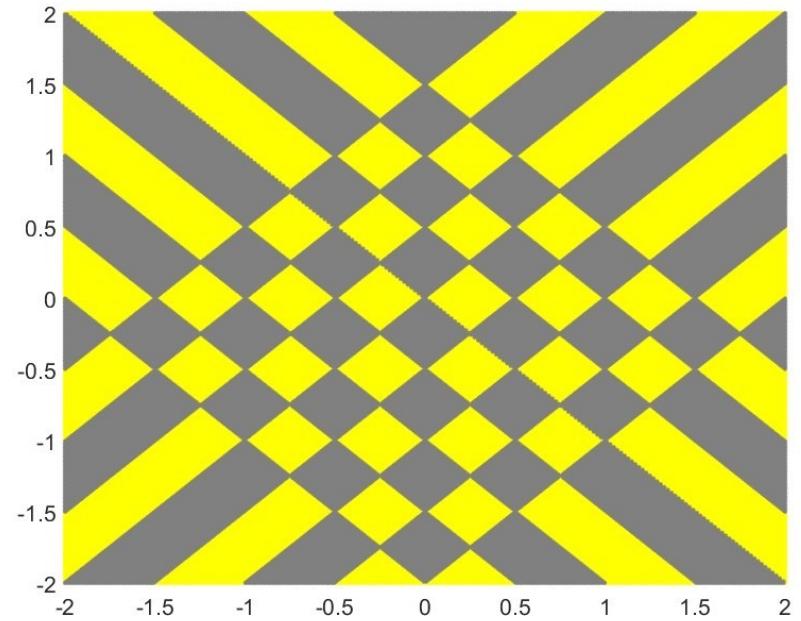
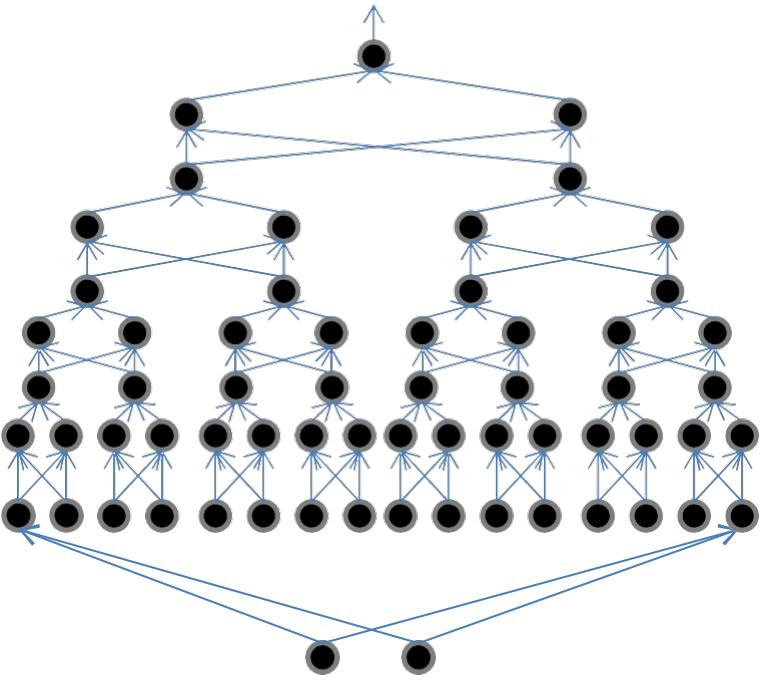
- Two-layer network: 56 hidden neurons
  - 16 in hidden layer 1
  - 40 in hidden layer 2
  - 57 total neurons, including output neuron

# Optimal depth



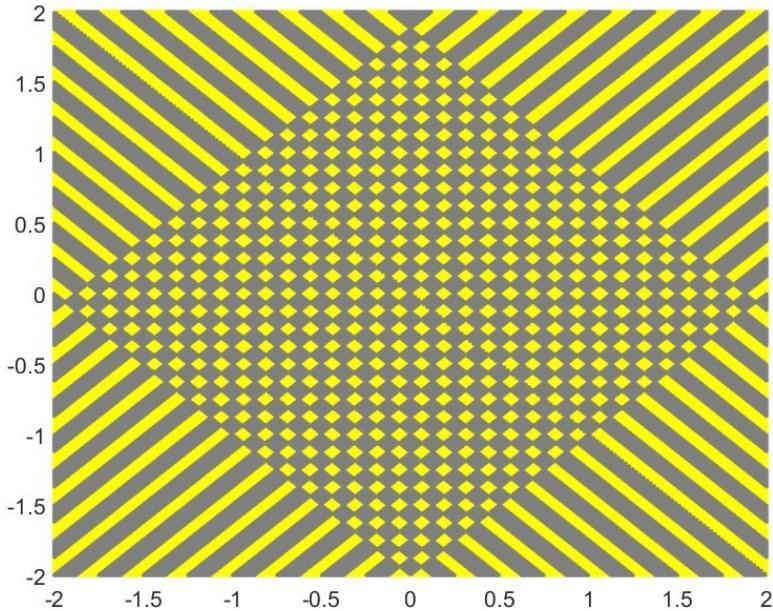
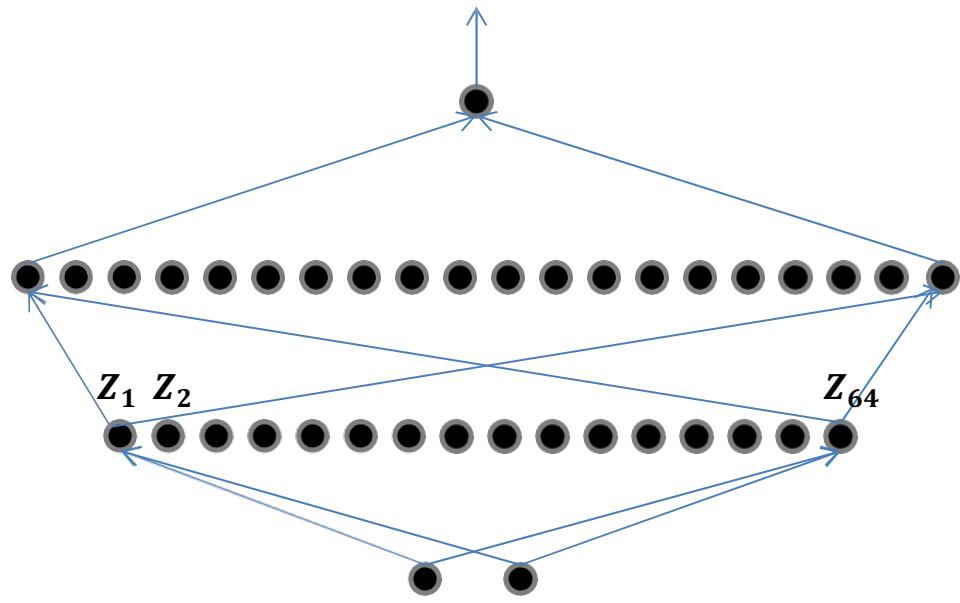
- But this is just  $Z_1 \oplus Z_2 \oplus \dots \oplus Z_{16}$

# Optimal depth



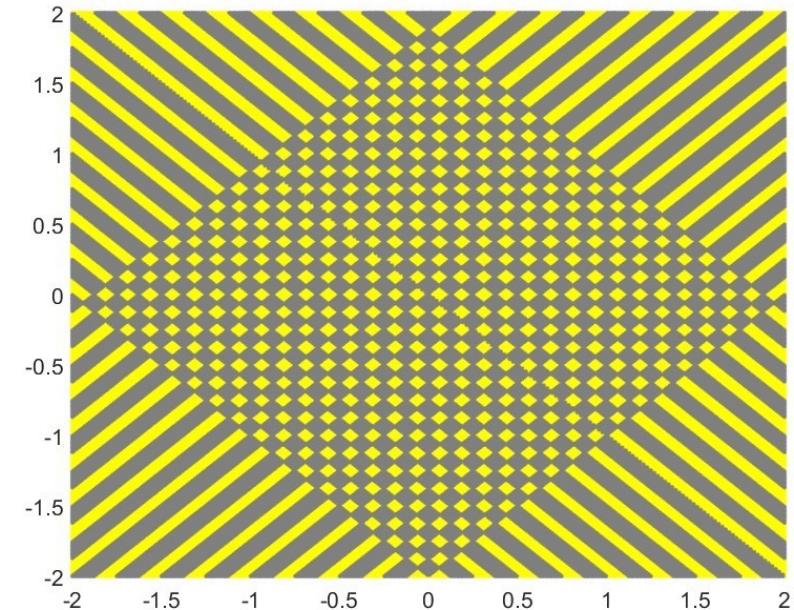
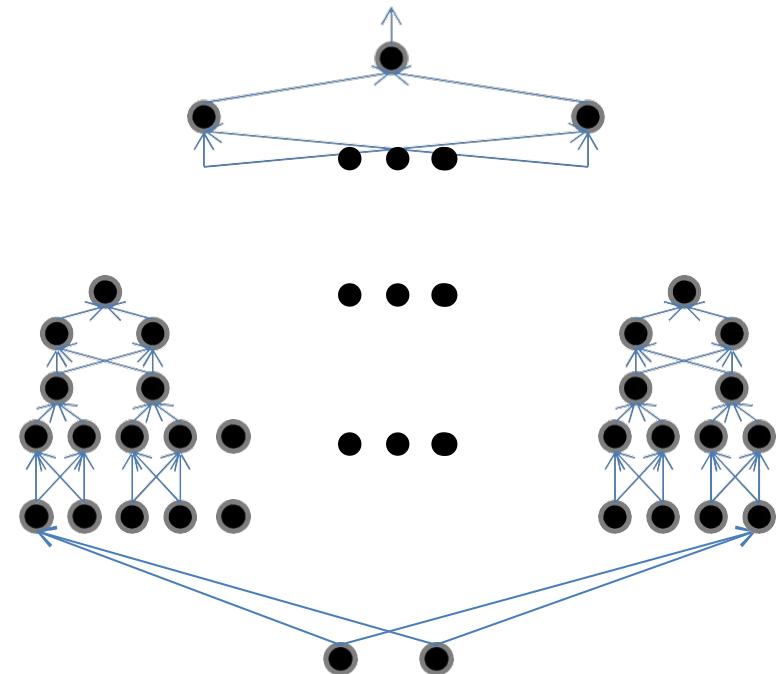
- But this is just  $Z_1 \oplus Z_2 \oplus \dots \oplus Z_{16}$ 
  - The XOR net will require  $16 + 15 \times 3 = 61$  neurons
    - 46 neurons if we use a two-gate XOR

# Optimal depth



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

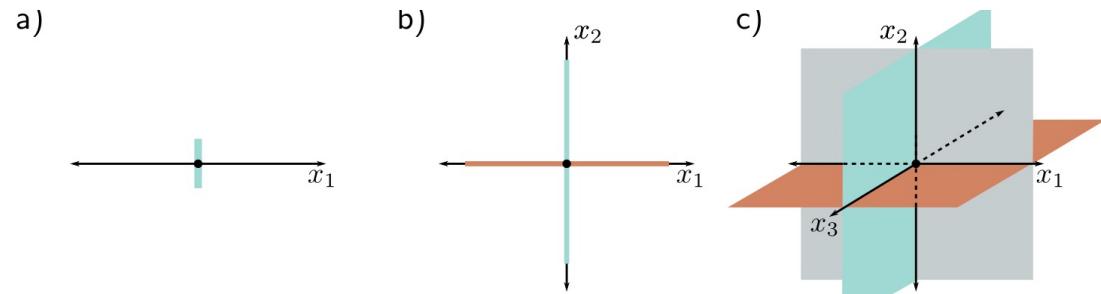
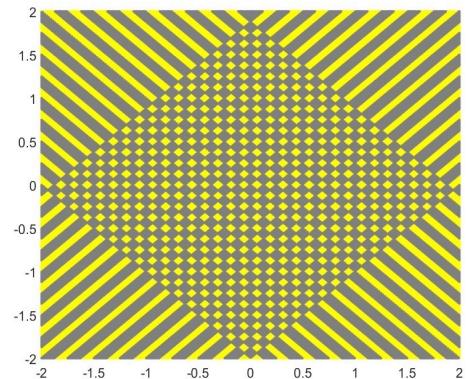
# Optimal depth



- 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 no of neurons in 1<sup>st</sup> hidden layer
  - $O(M^2)$  neurons in second hidden layer
  - Not exponential even though the pattern is an XOR
  - Why?
    - The data are two-dimensional and its exponential w.r.t.  $D = 2$
- For general case of  $M$  mutually intersecting hyperplanes in  $D$  dimensions, we will need  $O\left(\frac{M^D}{(D-1)!}\right)$  weights (assuming  $M \gg D$ ).
  - Increasing input dimensions can increase the words-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 ( $M$ )

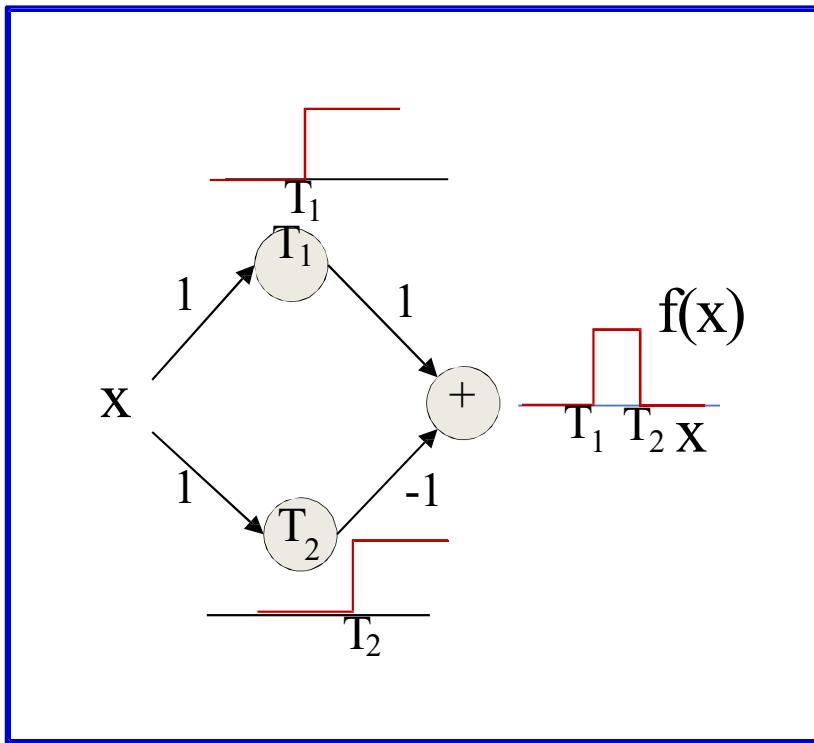


# Summary

- 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

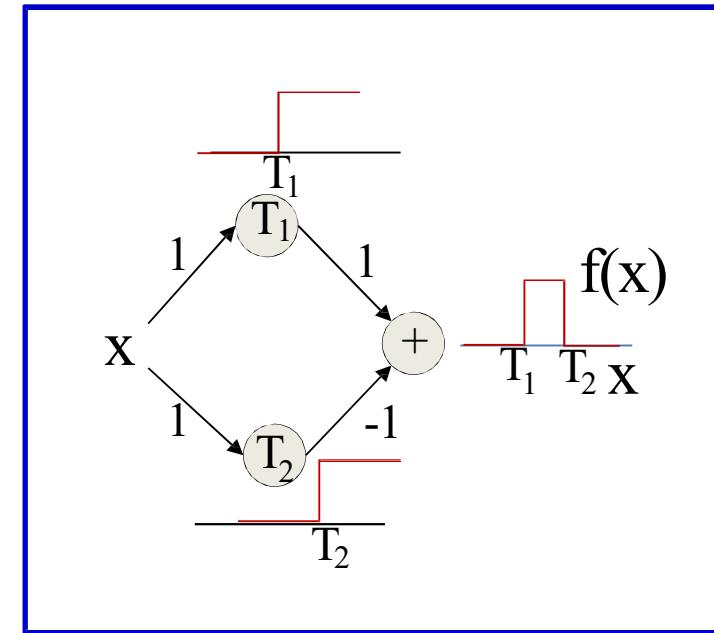
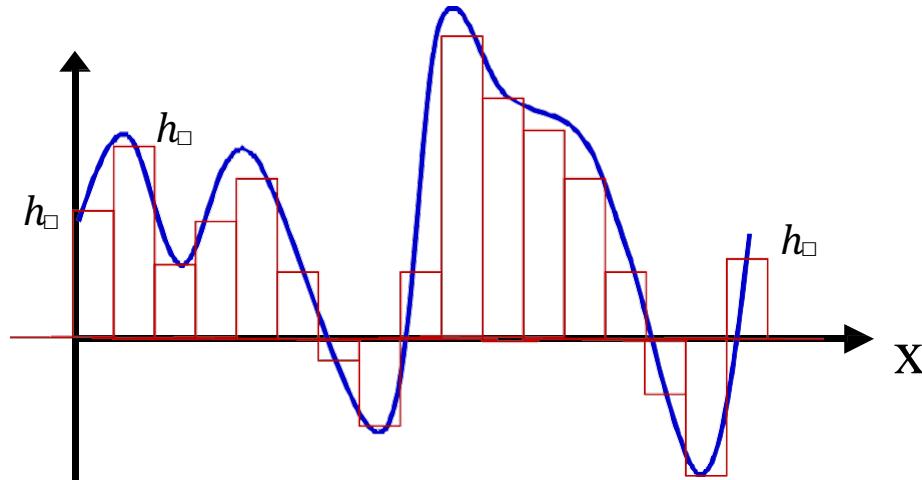
# MLPs as universal approximators

# MLPs as a continuous-valued regression



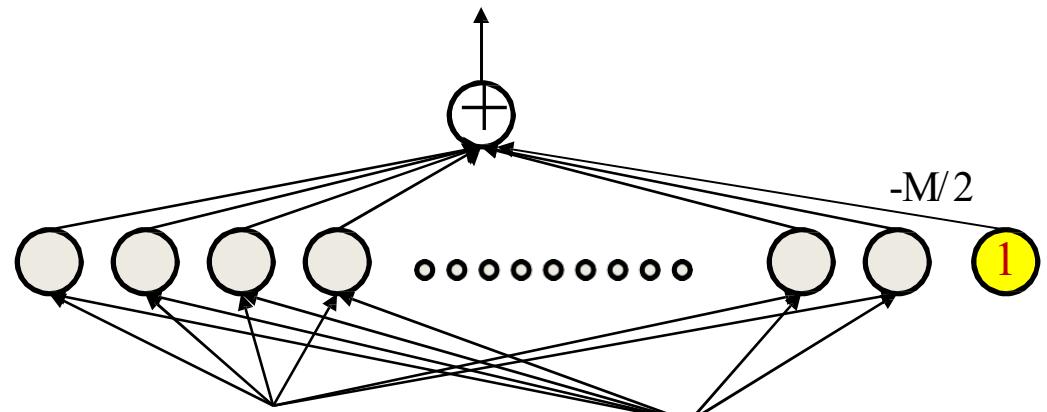
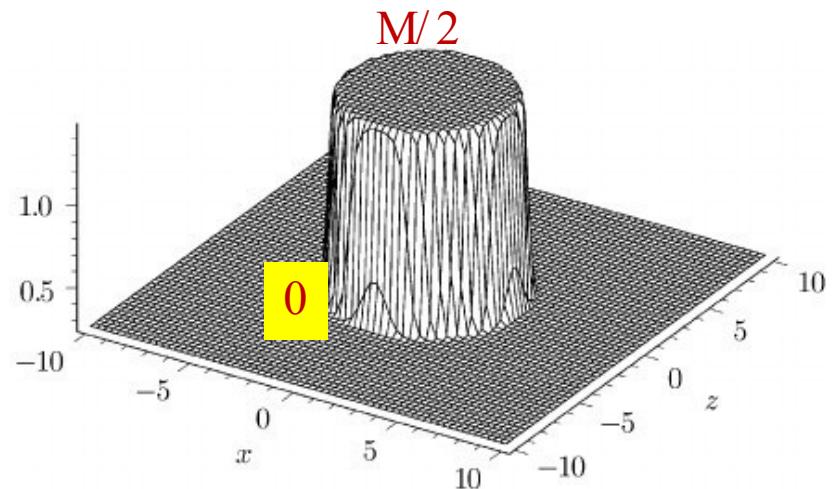
- A simple 3-unit MLP with a “summing” output unit can generate a “square pulse” over an input
  - Output is 1 only if the input lies between  $T_1$  and  $T_2$
  - $T_1$  and  $T_2$  can be arbitrarily specified

# MLPs as a continuous-valued regression



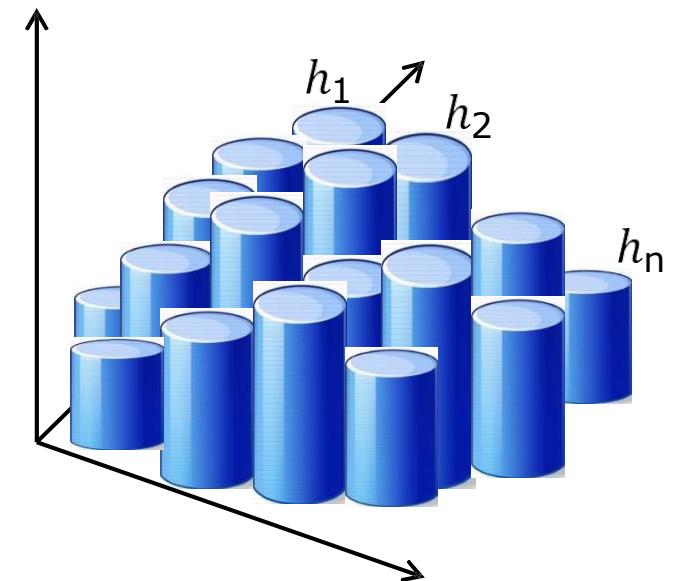
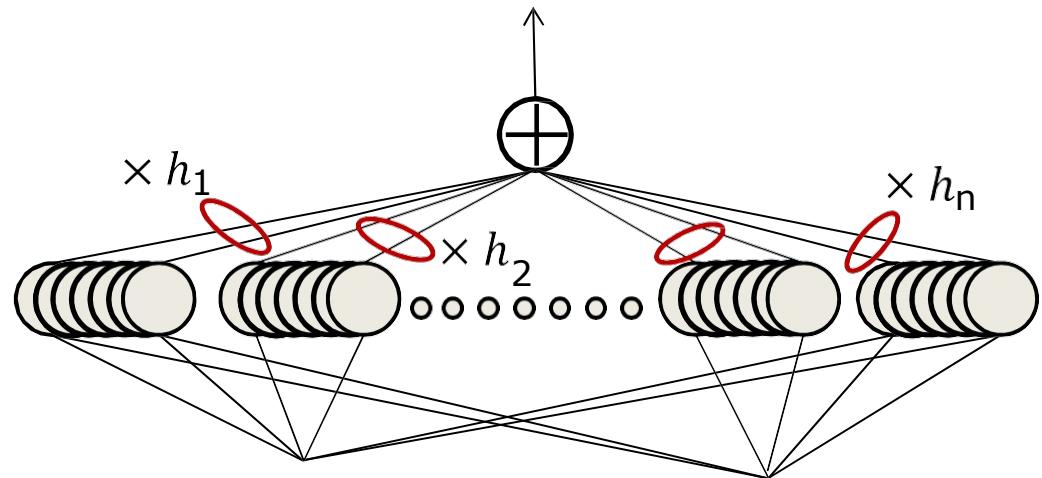
- A simple 3-unit MLP can generate a “square pulse” over an input
- An MLP with many units can model an arbitrary function over an input
  - To arbitrary precision
    - Simply make the individual pulses narrower
- A one-layer MLP can model an arbitrary function of a single input

# For higher dimensions



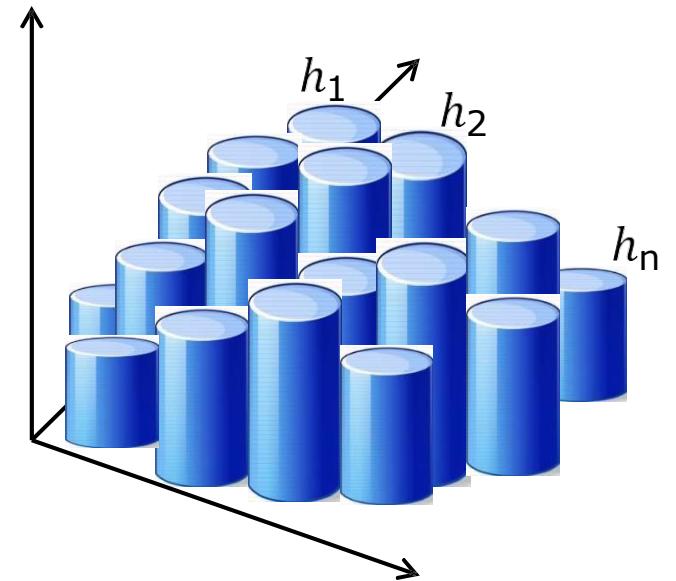
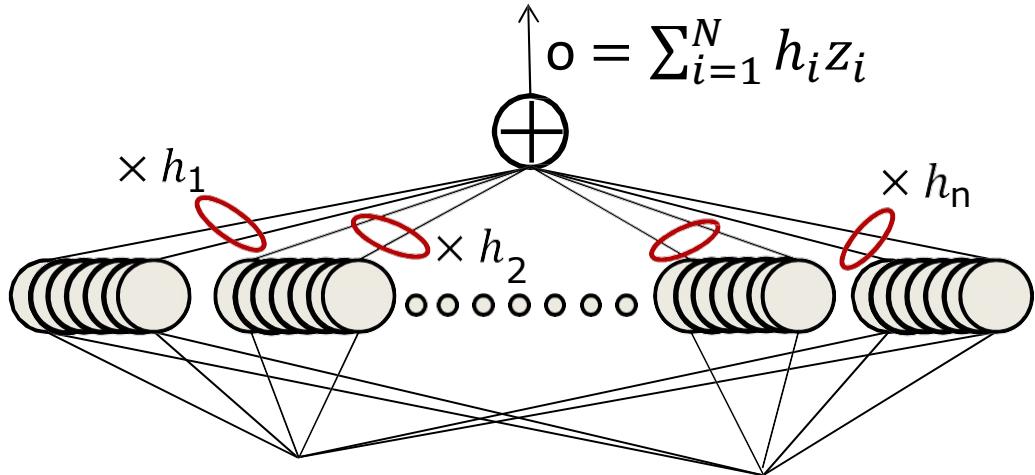
- An MLP can compose a cylinder
  - $\frac{M}{2}$  in the circle, 0 outside

# MLPs as a continuous-valued function



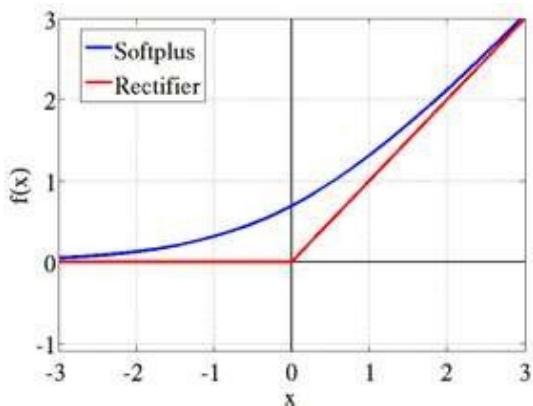
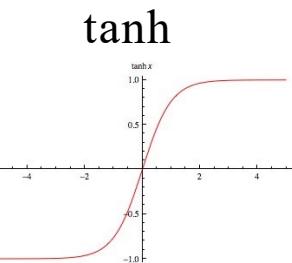
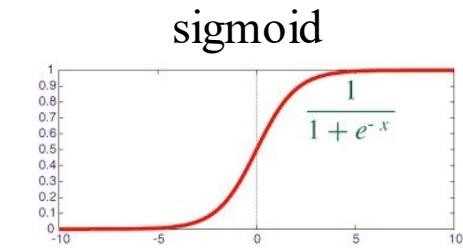
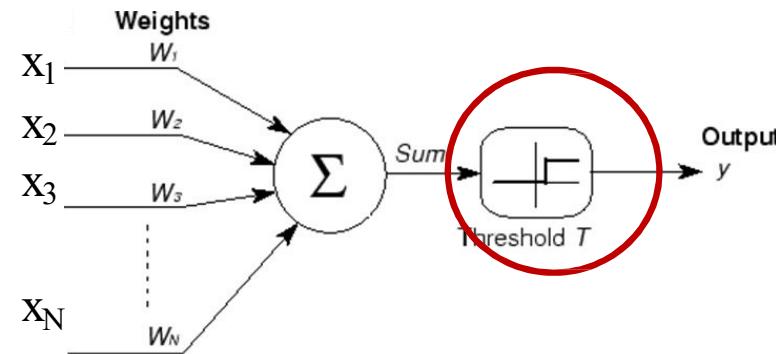
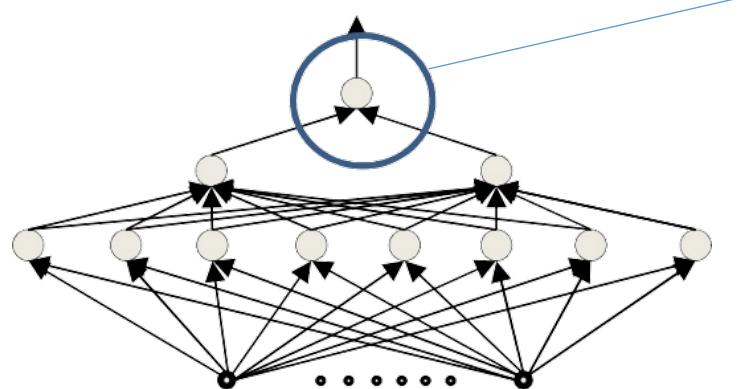
- MLPs can actually compose arbitrary functions in any number of dimensions!
  - Even with only one layer
    - As sums of scaled and shifted cylinders
  - To arbitrary precision
    - By making the cylinders thinner
  - The MLP is a universal approximator!

# Caution: MLPs with additive output units are universal approximators



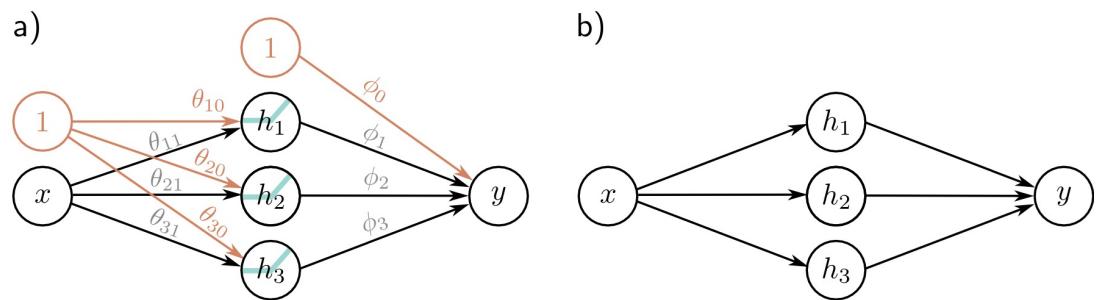
- MLPs can actually compose arbitrary functions in any number of dimensions!
- But explanation so far only holds if the output unit only performs summation
  - i.e. does not have an additional “activation”

# “Proper” networks: Outputs with activations

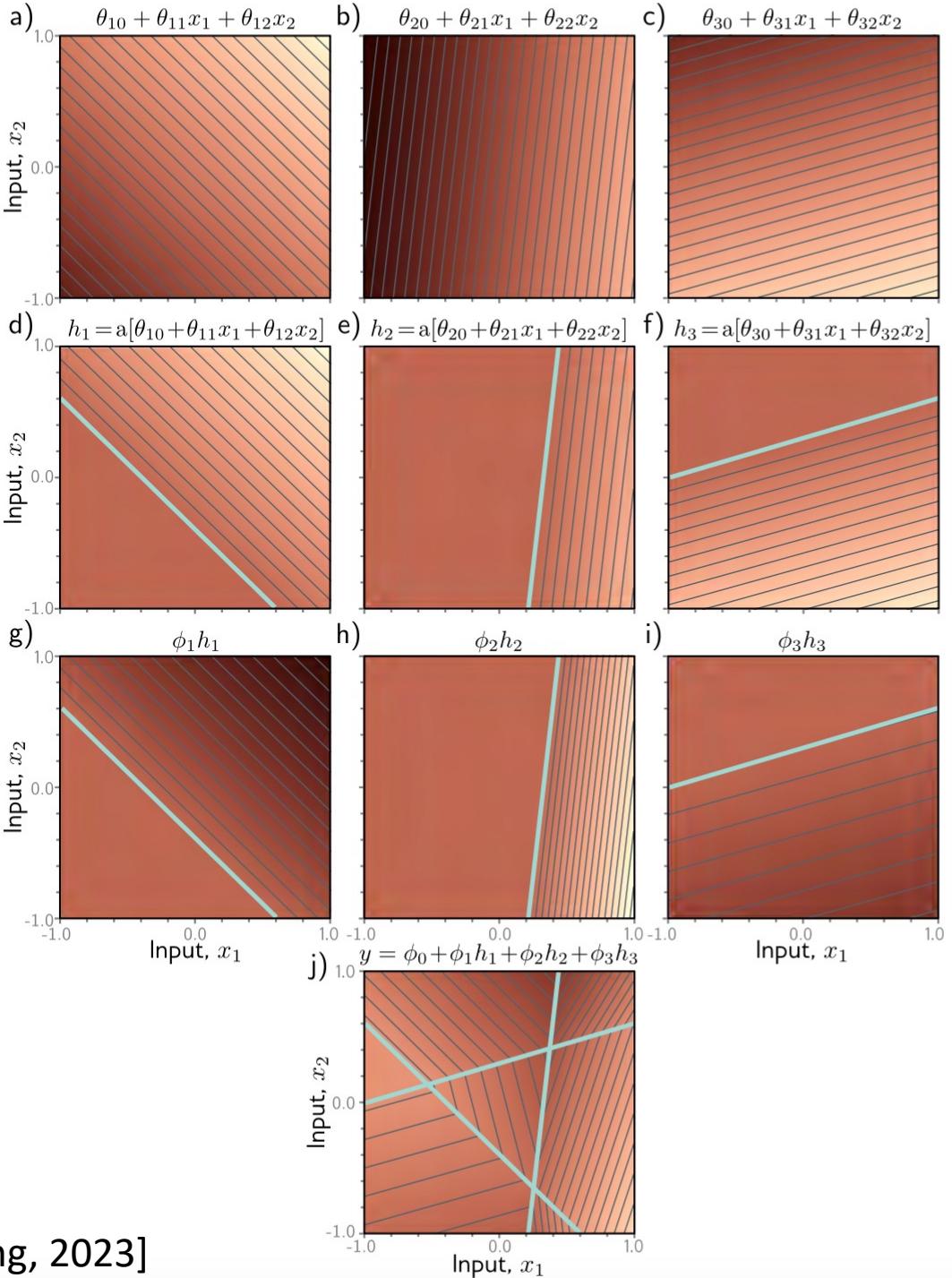
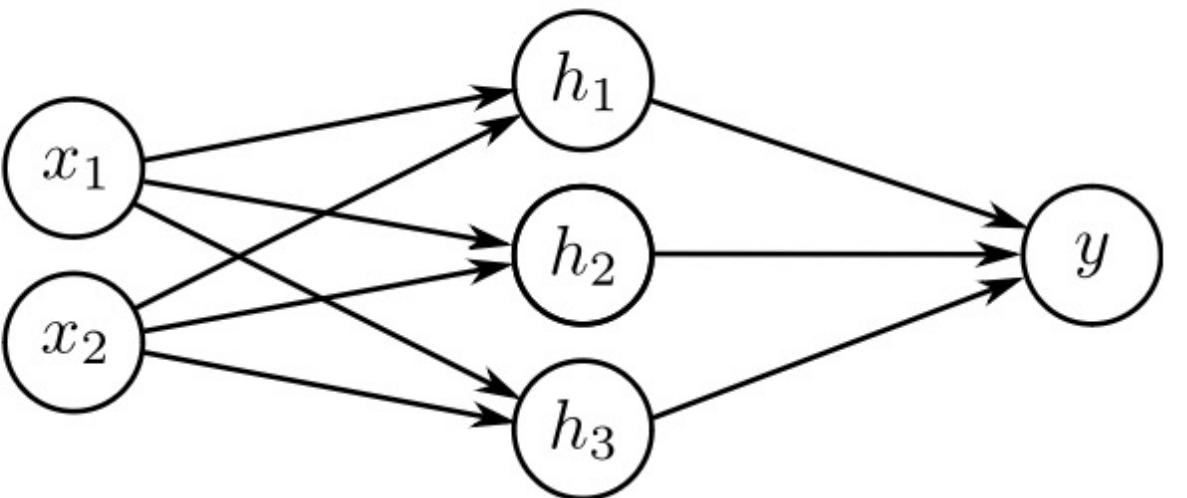


- Output neuron may have actual “activation”
  - Threshold, sigmoid, tanh, softplus, rectifier, etc.
- What is the property of such networks?

# MLP with ReLU Activation

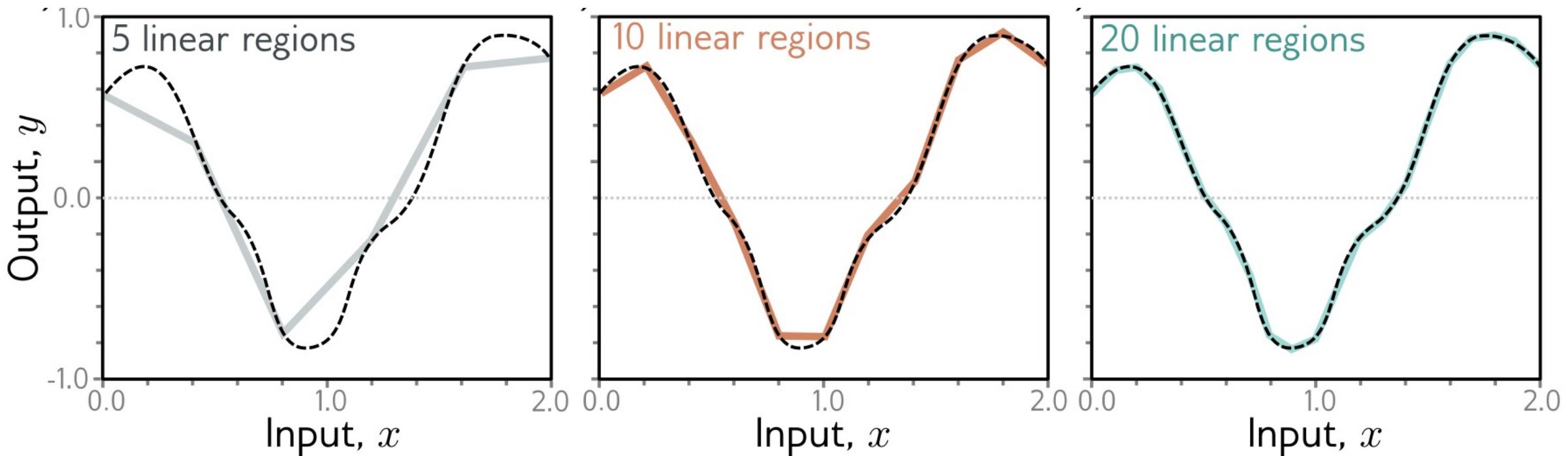


# Multivariate Inputs

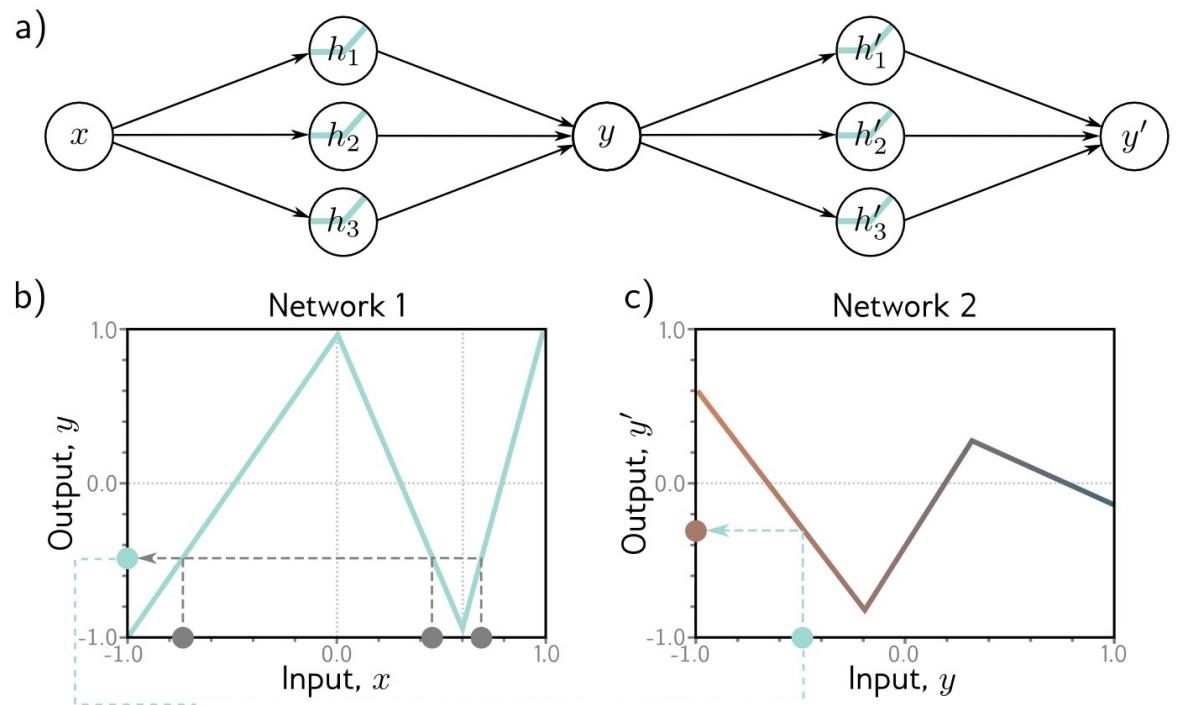


# Approximating by MLP with ReLUs

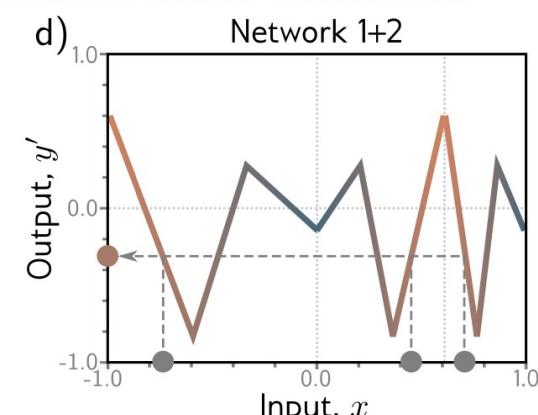
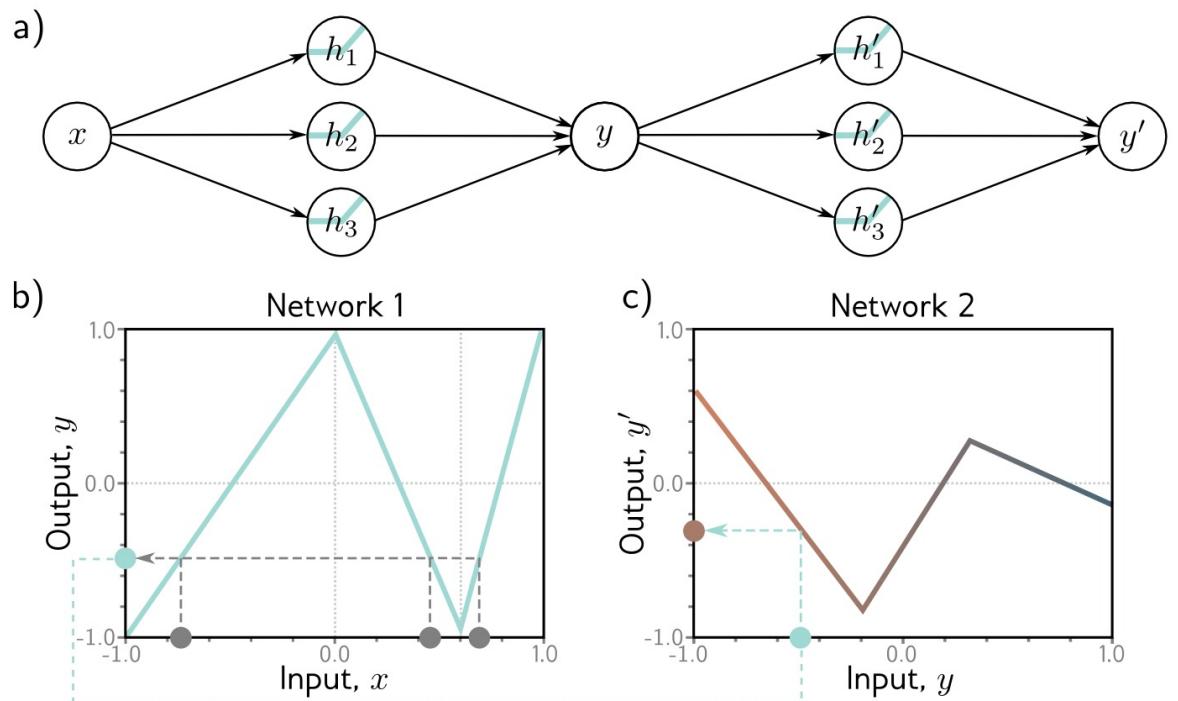
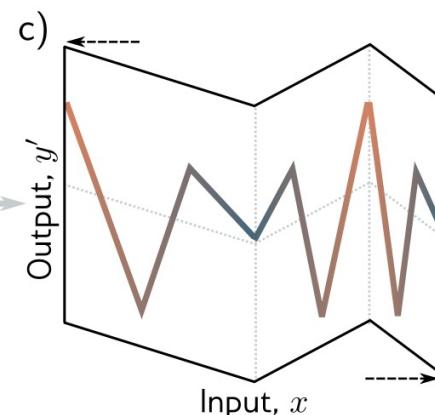
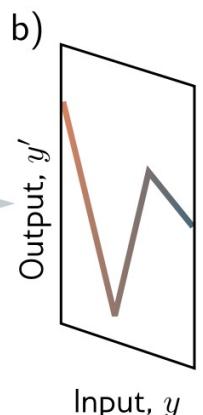
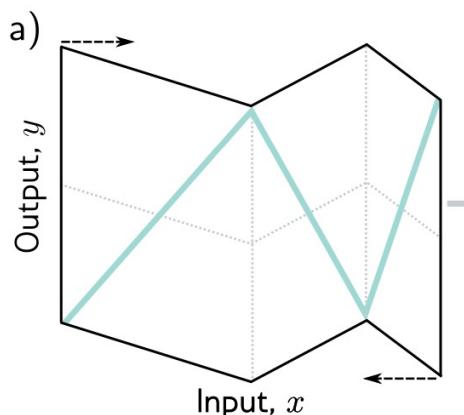
- Approximation by piecewise linear functions



# Composing Networks

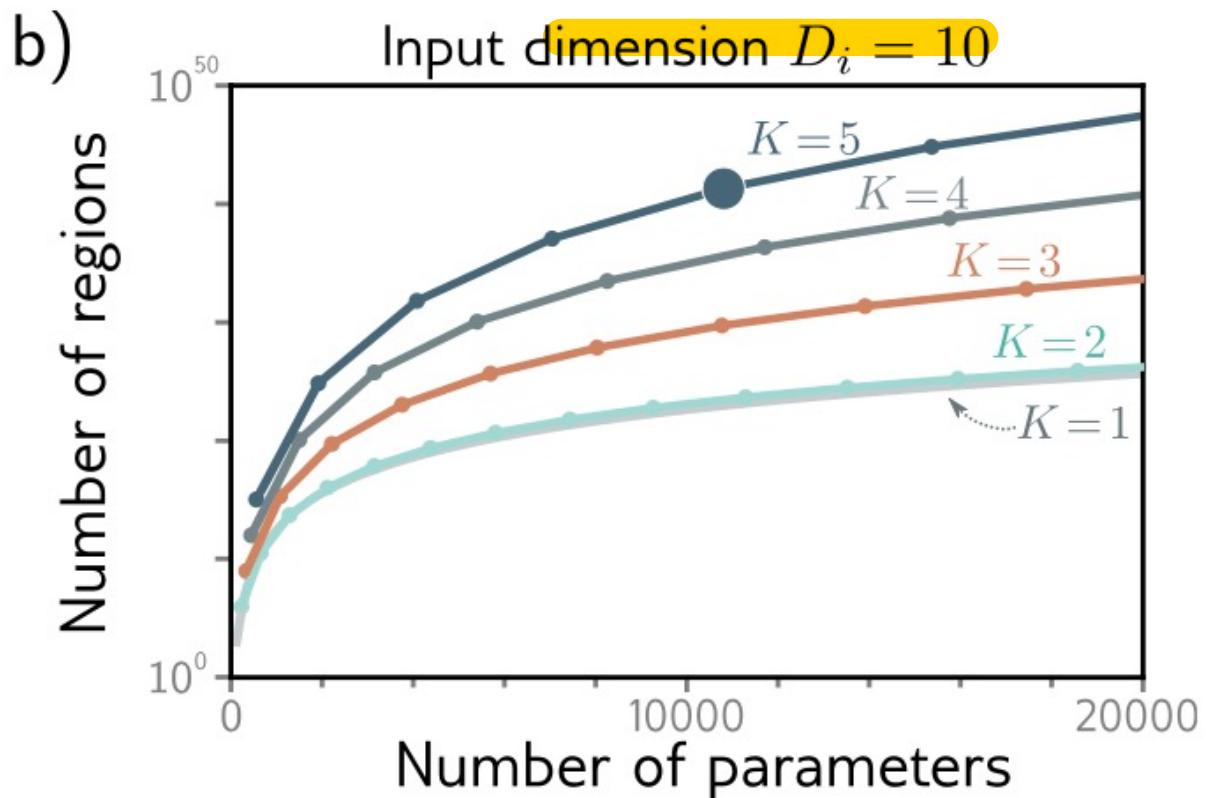
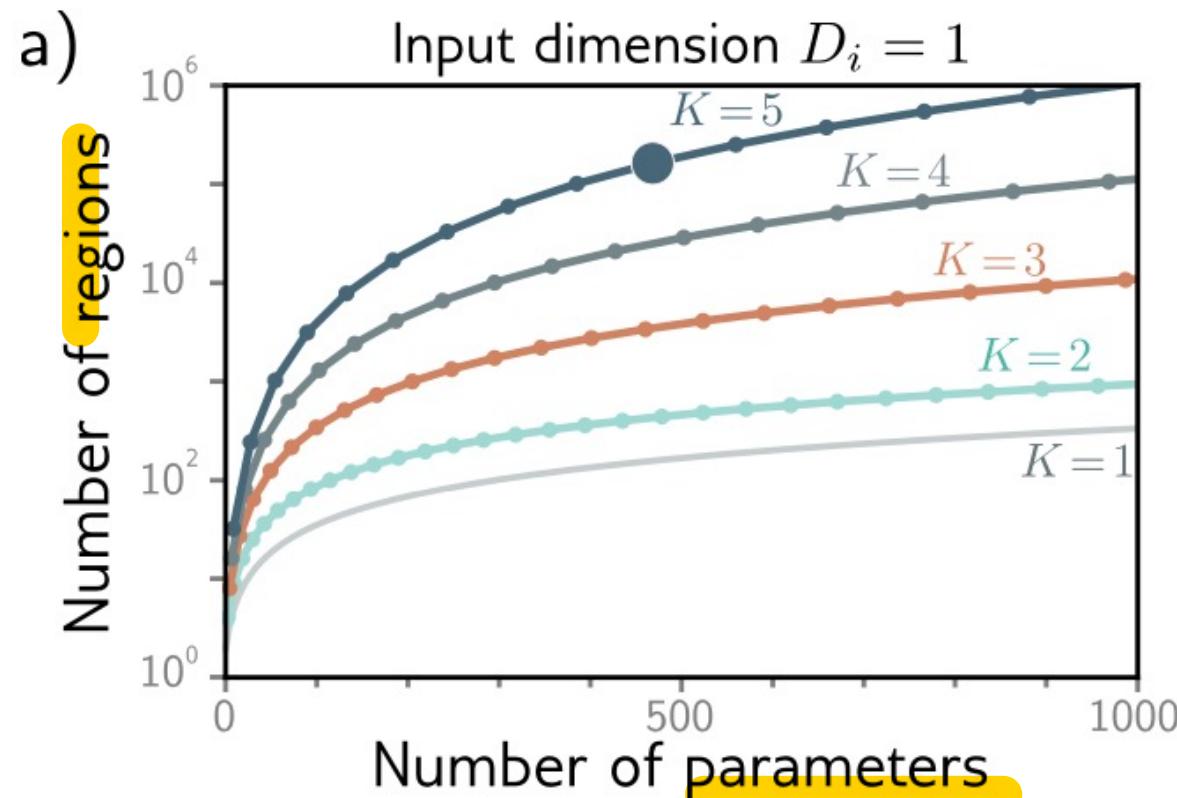


# Composing Networks



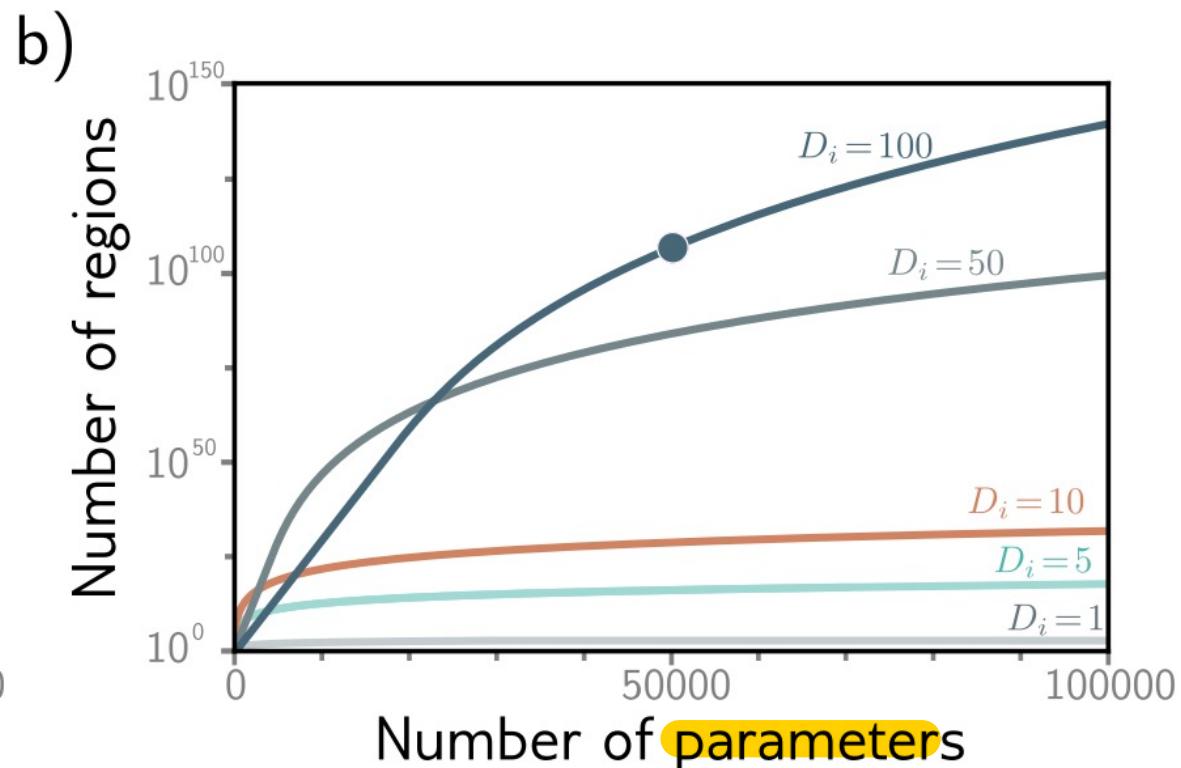
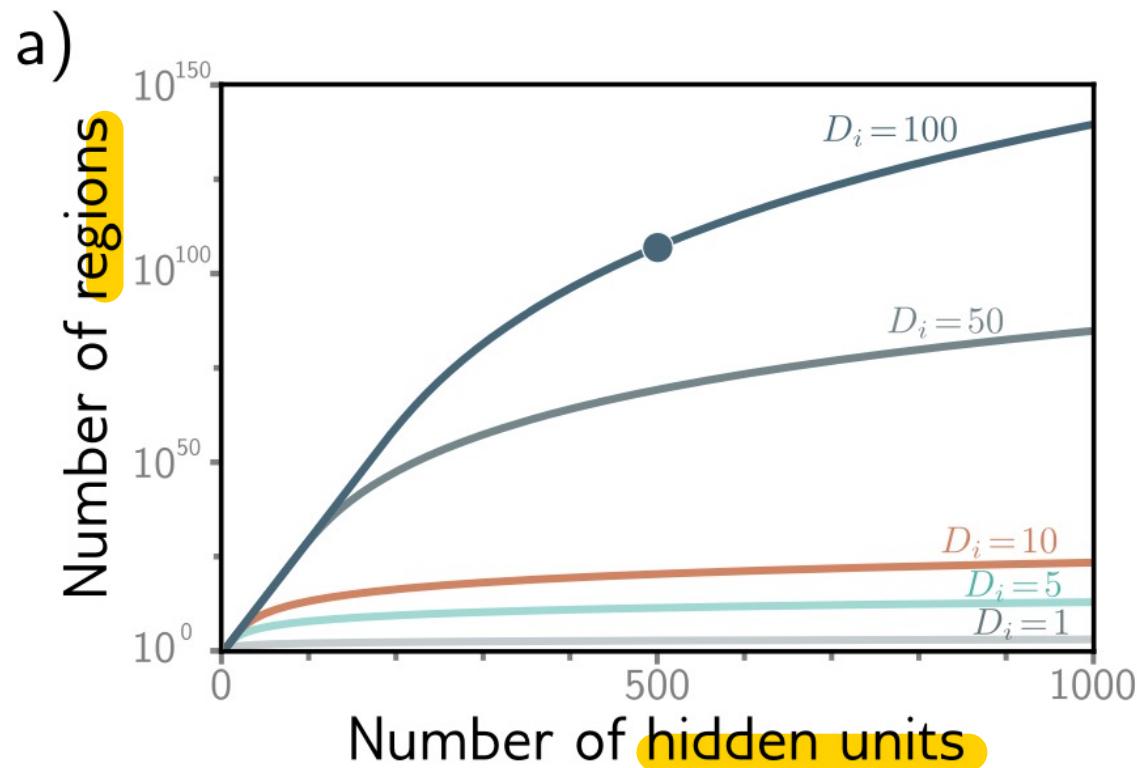
# Max no of Regions

- Max number of linear regions increases as a function of the number of parameters for networks mapping scalar input  $x$  to scalar output  $y$ .
- For  $K$  layers:



# Max no of Regions

- Maximum possible regions as a function of the number of hidden units and the number of parameters for five different input dimensions



# Summary

- 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