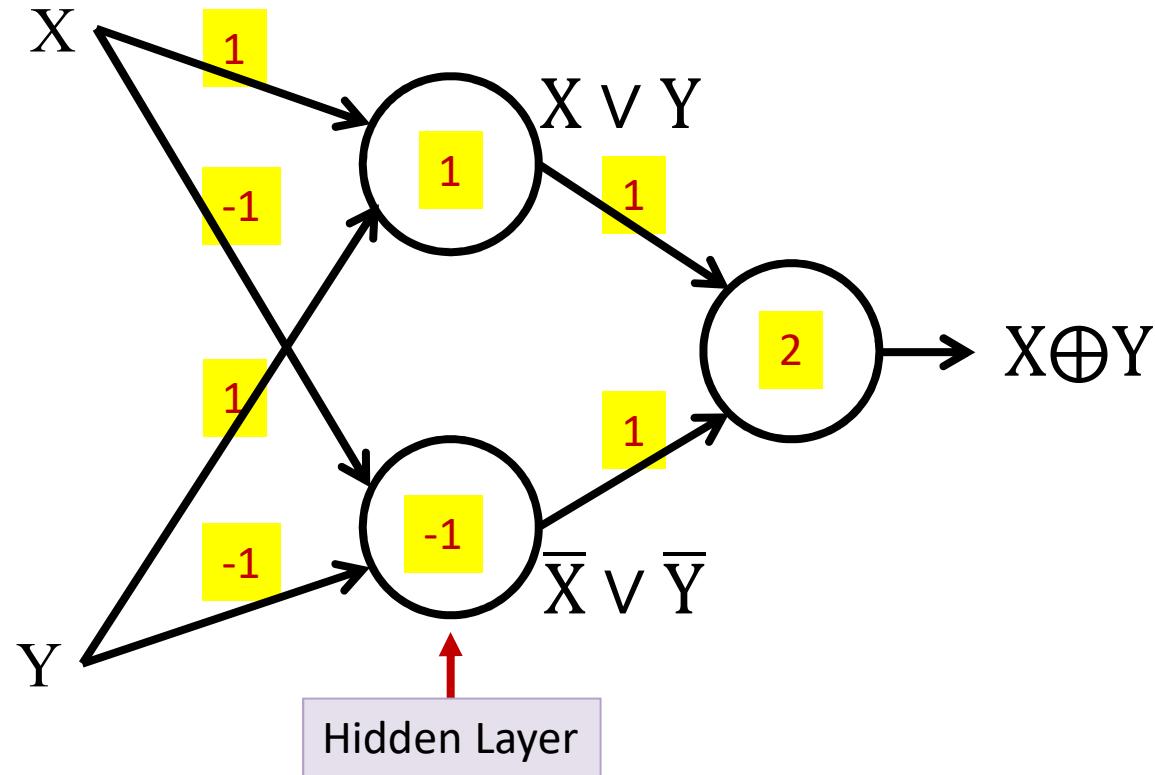


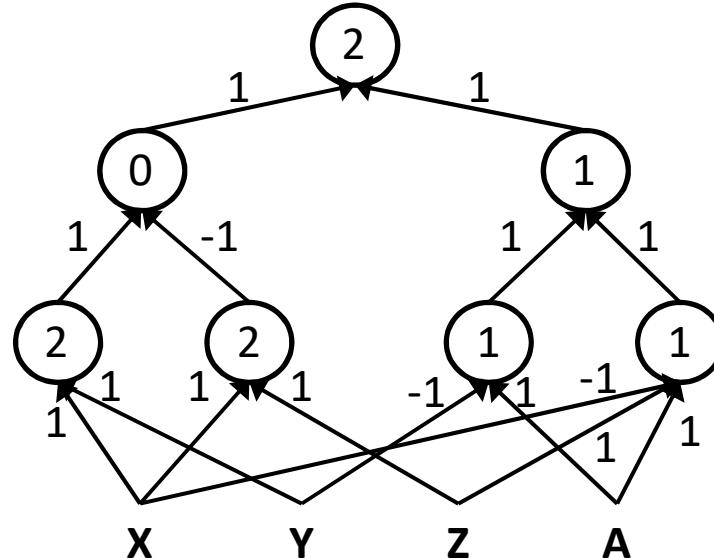
Multi-layer perceptron



- MLPs can compute the XOR

Multi-layer perceptron

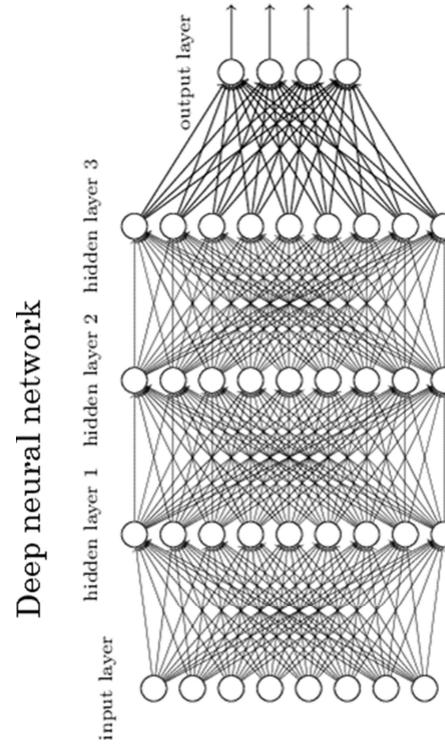
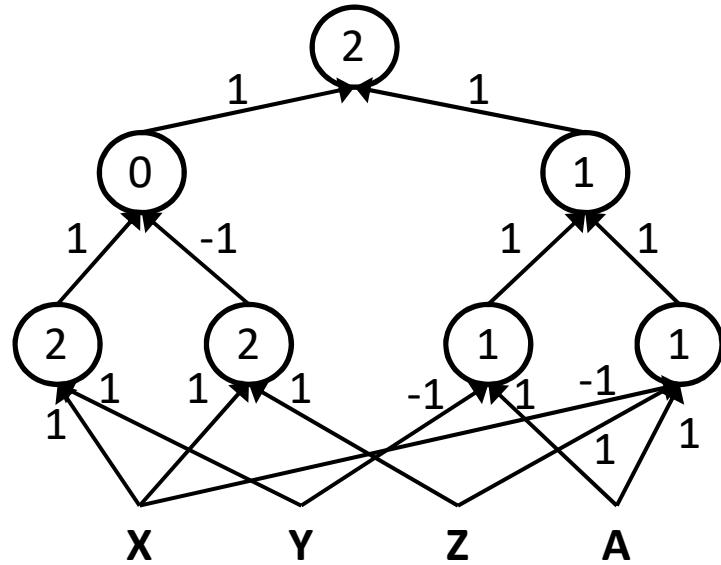
$$((A \& \bar{X} \& Z) | (A \& \bar{Y})) \& ((X \& Y) | \bar{(X \& Z)})$$



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

MLP as Boolean Functions

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



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

How many layers for a Boolean MLP?

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

Truth Table

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

- *A Boolean function is just a truth table*

How many layers for a Boolean MLP?

Truth Table

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

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

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

- Expressed in disjunctive normal form

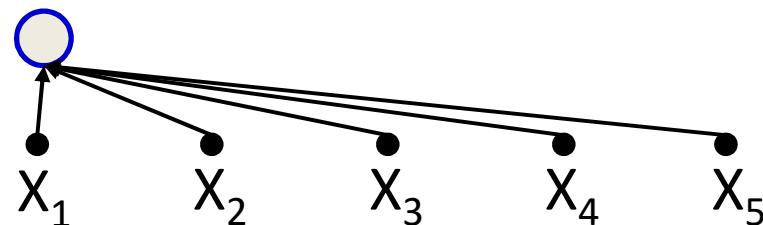
How many layers for a Boolean MLP?

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

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



- Expressed in disjunctive normal form

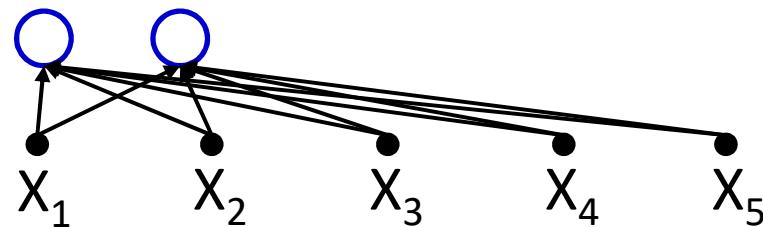
How many layers for a Boolean MLP?

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

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



- Expressed in disjunctive normal form

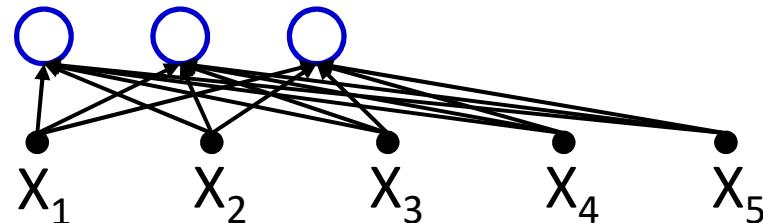
How many layers for a Boolean MLP?

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

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



- Expressed in disjunctive normal form

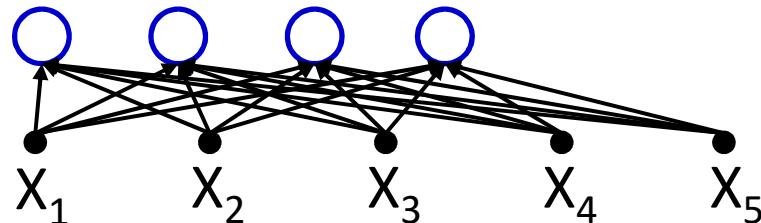
How many layers for a Boolean MLP?

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

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



- Expressed in disjunctive normal form

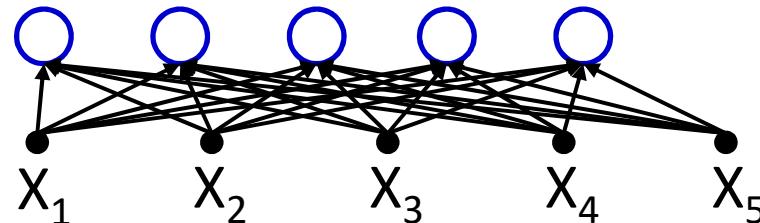
How many layers for a Boolean MLP?

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

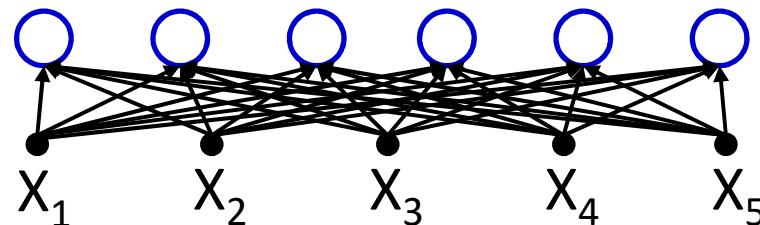
How many layers for a Boolean MLP?

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

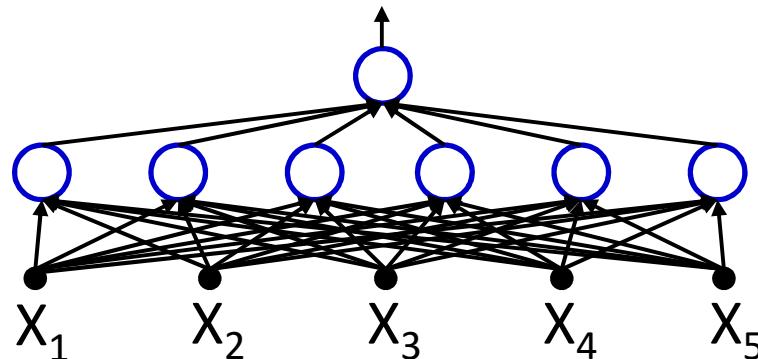
How many layers for a Boolean MLP?

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

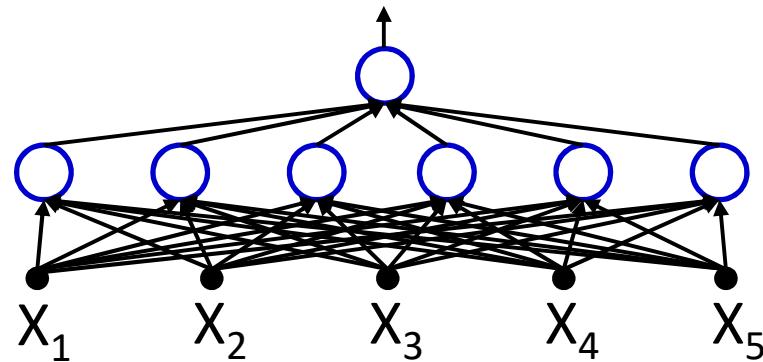
How many layers for a Boolean MLP?

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

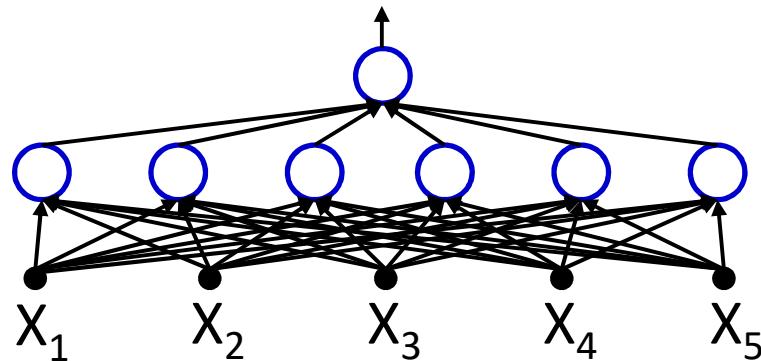
How many layers for a Boolean MLP?

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

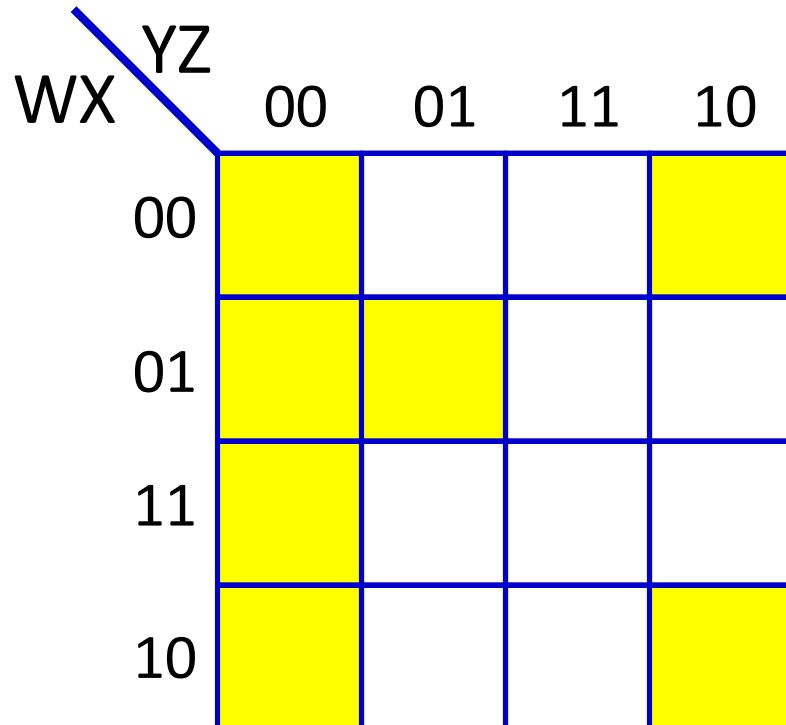
$$Y = \bar{X}_1 \bar{X}_2 X_3 X_4 \bar{X}_5 + \bar{X}_1 X_2 \bar{X}_3 X_4 X_5 + \bar{X}_1 X_2 X_3 \bar{X}_4 \bar{X}_5 + X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 X_5 + X_1 \bar{X}_2 X_3 X_4 X_5 + X_1 X_2 \bar{X}_3 \bar{X}_4 X_5$$



- 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



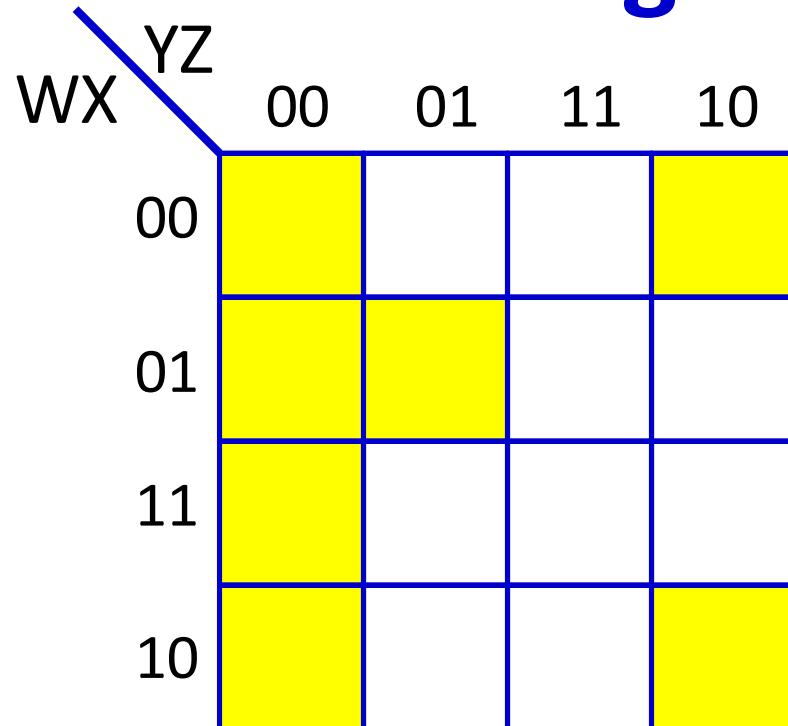
This is a "Karnaugh Map"

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

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

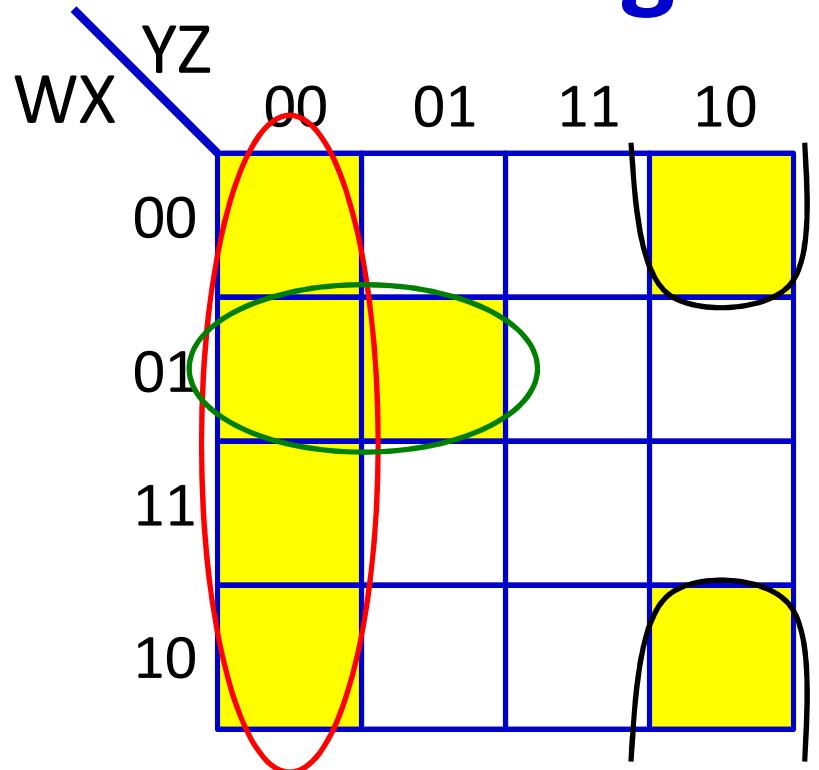
- DNF form:
 - Find groups
 - Express as reduced DNF

Reducing a Boolean Function



Basic DNF formula will require 7 terms

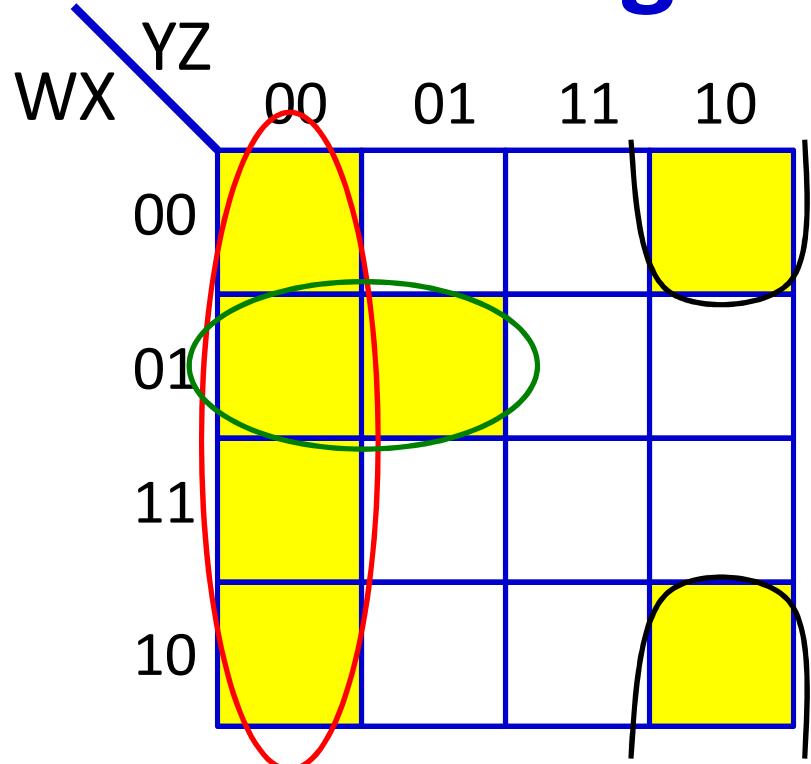
Reducing a Boolean Function



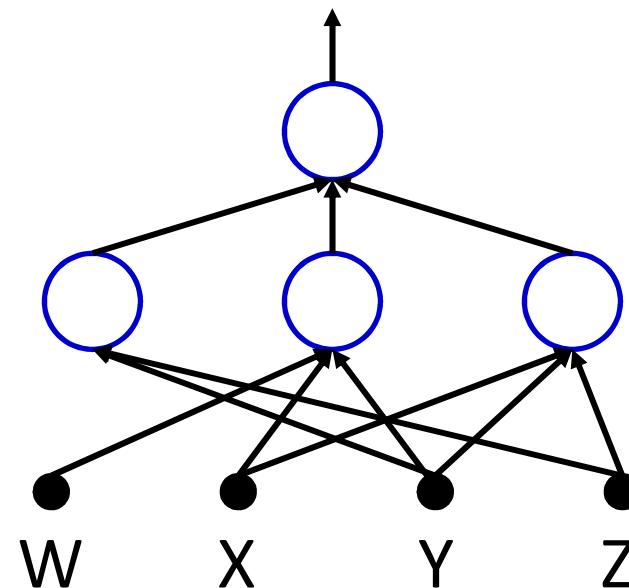
$$O = \bar{Y}\bar{Z} + \bar{W}X\bar{Y} + \bar{X}Y\bar{Z}$$

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

Reducing a Boolean Function

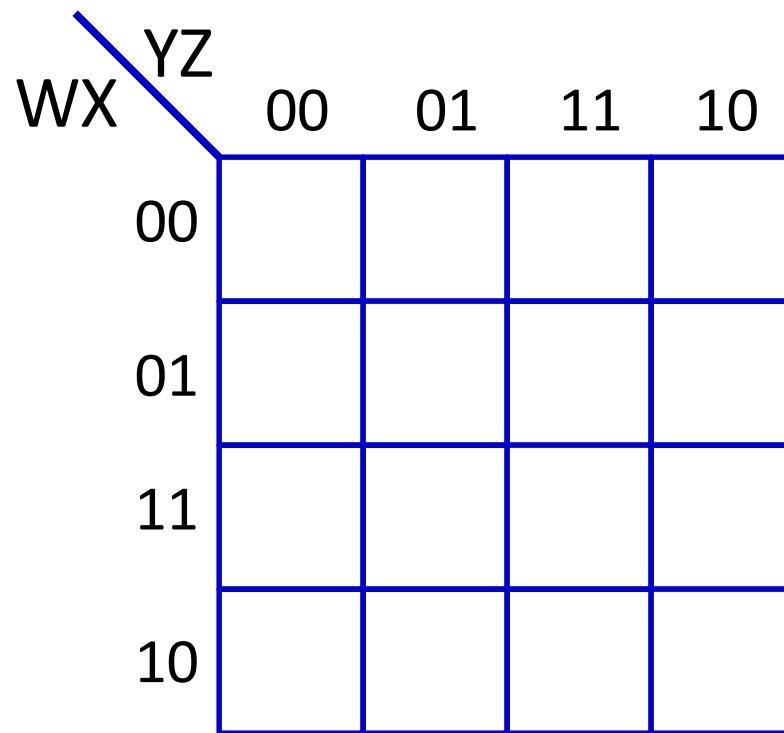


$$O = \bar{Y}\bar{Z} + \bar{W}X\bar{Y} + \bar{X}YZ$$



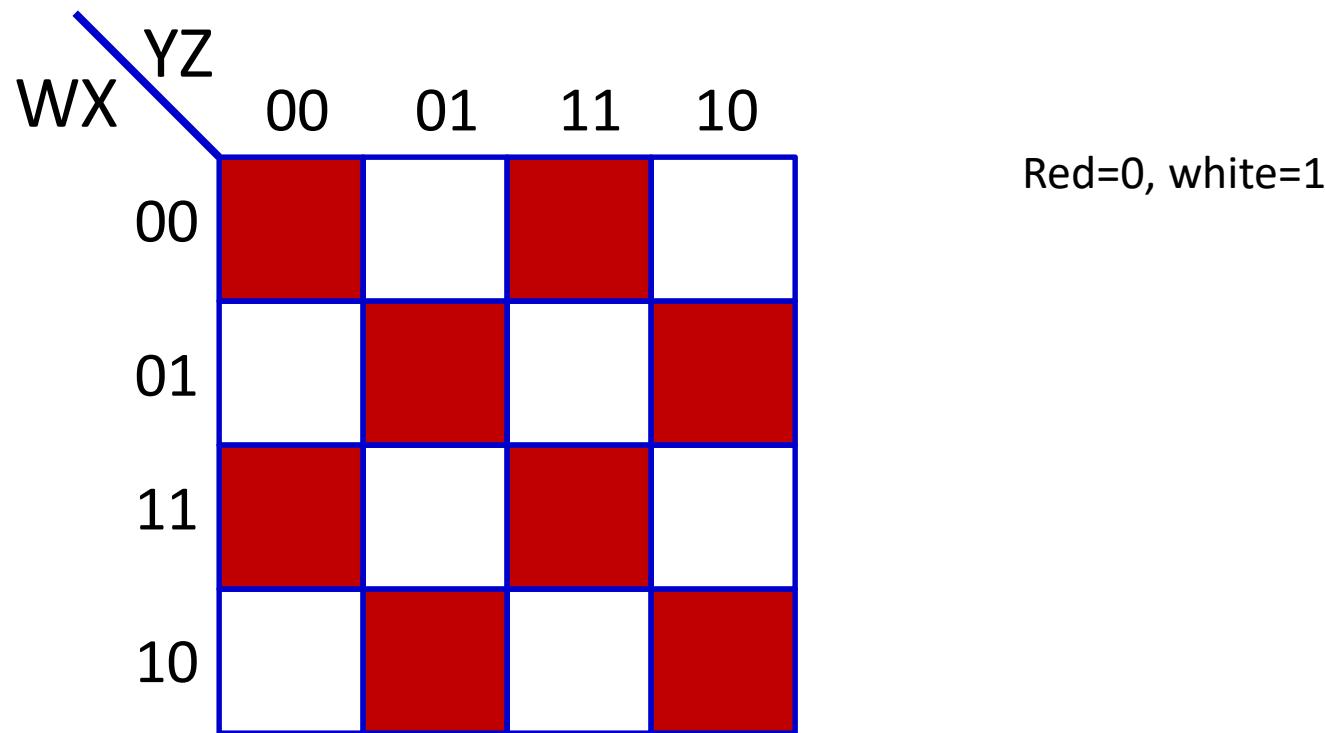
- *Reduced DNF form:*
 - Find groups
 - Express as *reduced* DNF
 - Boolean network for this function needs only 3 hidden units
 - Reduction of the DNF reduces the size of the one-hidden-layer network

Largest irreducible DNF?



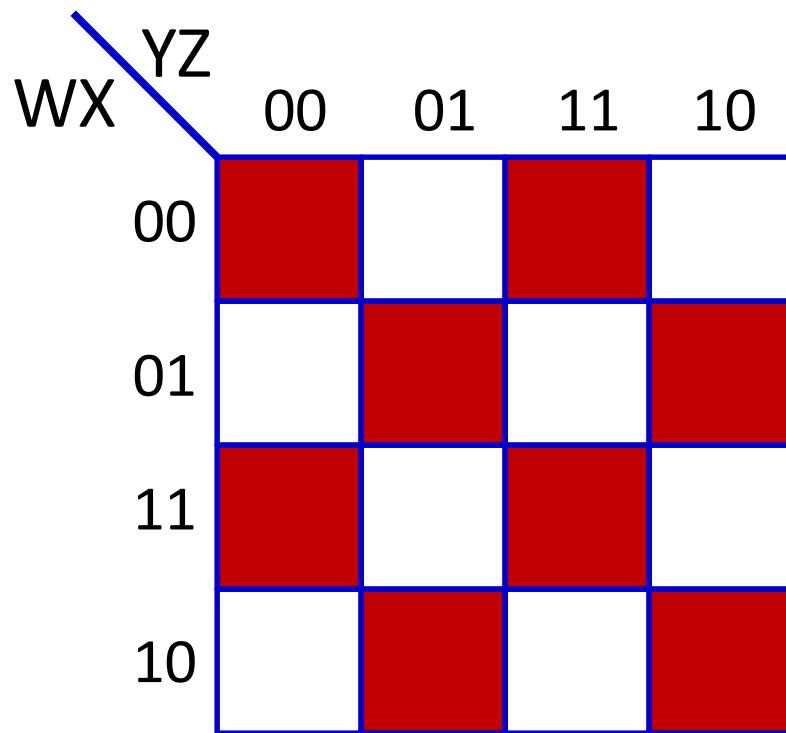
- What arrangement of ones and zeros simply cannot be reduced further?

Largest irreducible DNF?



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

Largest irreducible DNF?

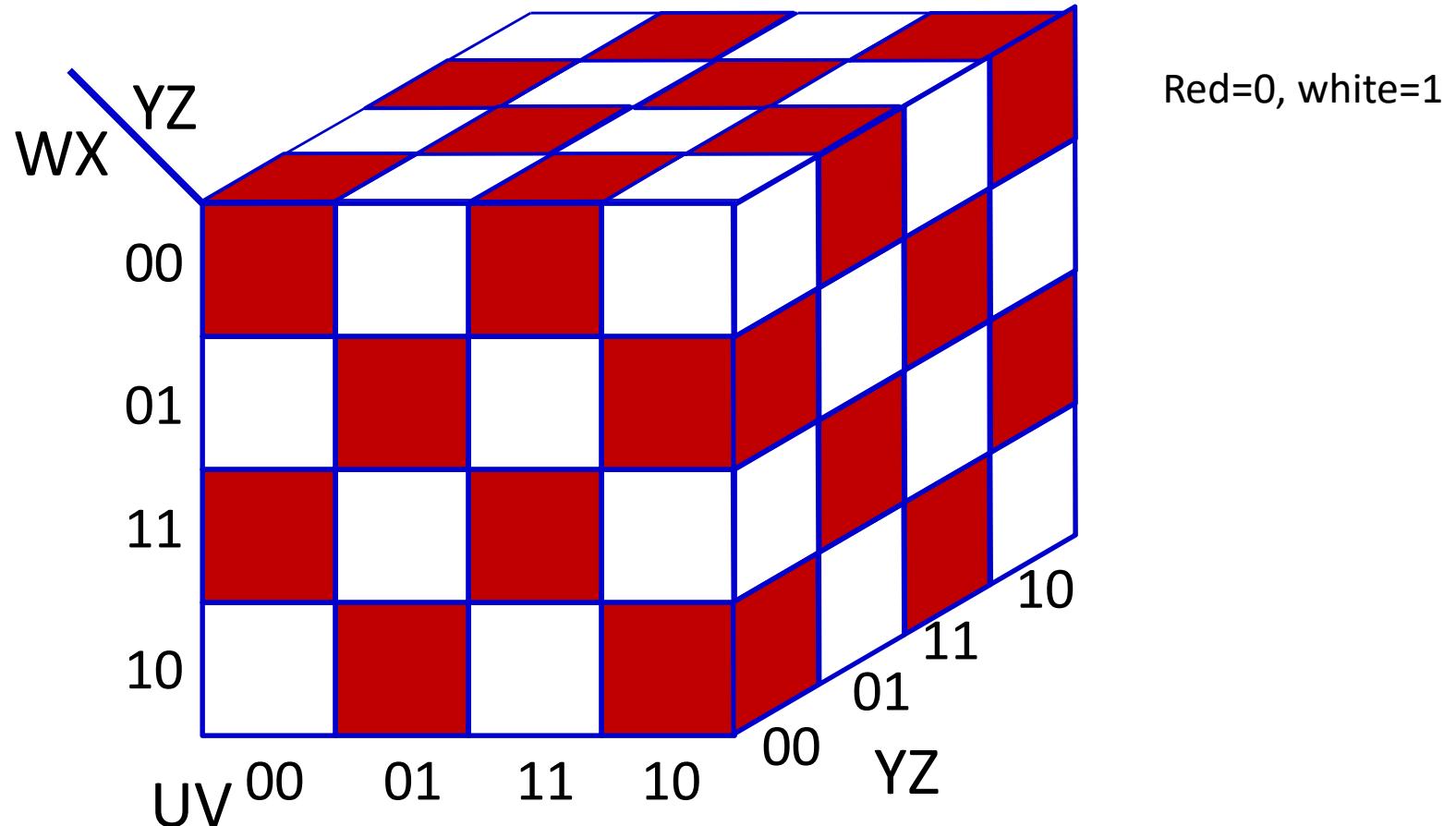


	00	01	11	10
00	Red	White	Red	White
01	White	Red	White	Red
11	Red	White	Red	White
10	White	Red	White	Red

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 single-layer Boolean MLP



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

Width of a single-layer Boolean MLP

WX
YZ
00

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

10
UV 00 01 11 10 00 10
01 11 YZ

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

Width of a single-layer Boolean MLP

WX
YZ
00

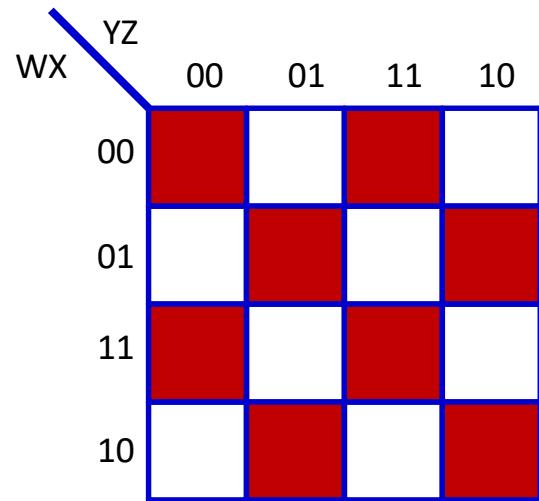
Can be generalized: Will require 2^{N-1} perceptrons in hidden layer
Exponential in N

10
UV 00 01 11 10 00 10
01 11
YZ

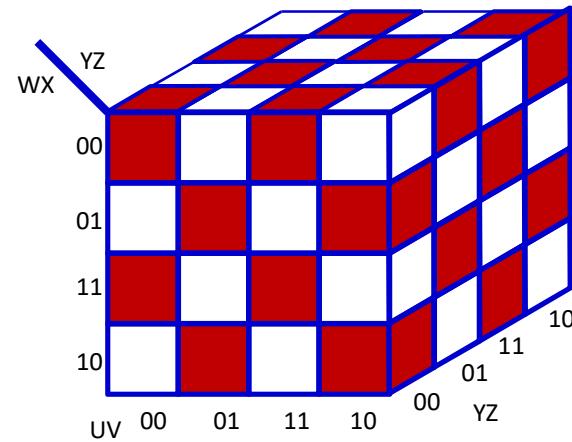
How many units if we use *multiple layers*?

layer) MLP for this Boolean function

Width of a deep MLP



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

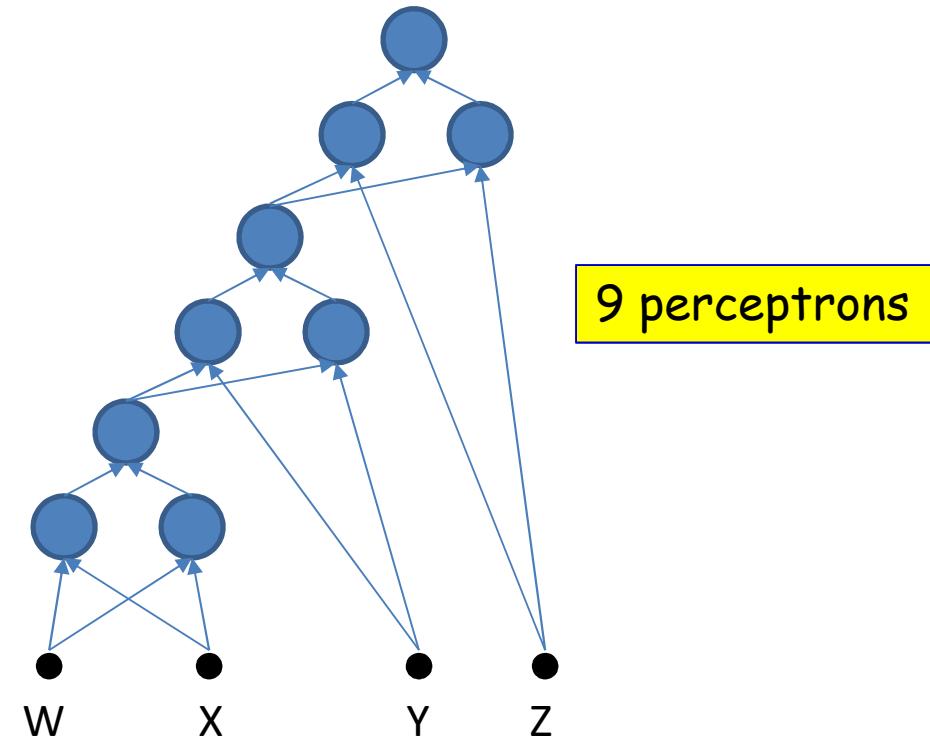


$$O = U \oplus V \oplus W \oplus X \oplus Y \oplus Z$$

Width of a deep MLP

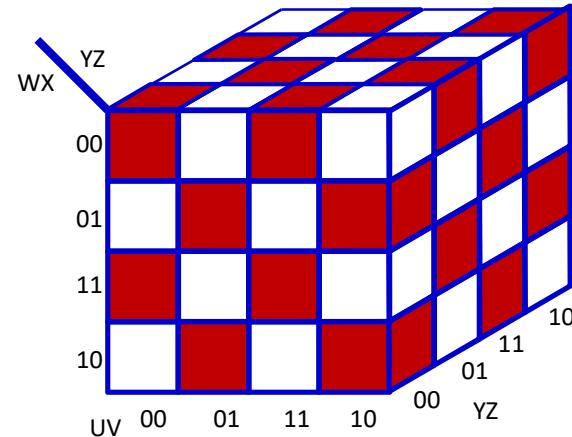
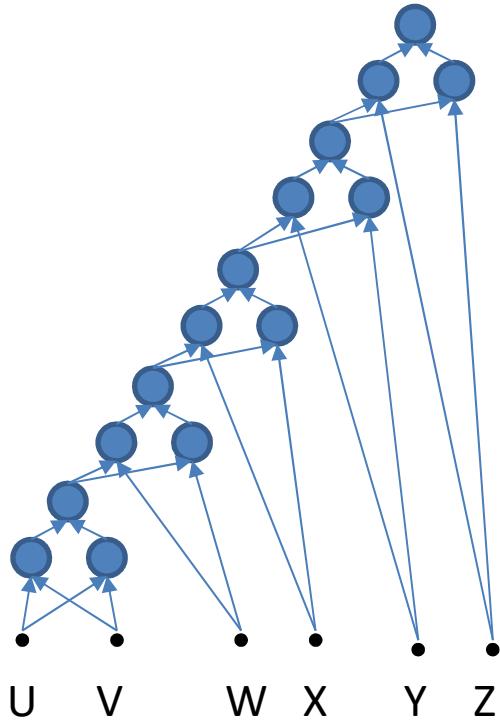
	00	01	11	10
00	red	white	red	white
01	white	red	white	red
11	red	white	red	white
10	white	red	white	red

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



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

Width of a deep MLP

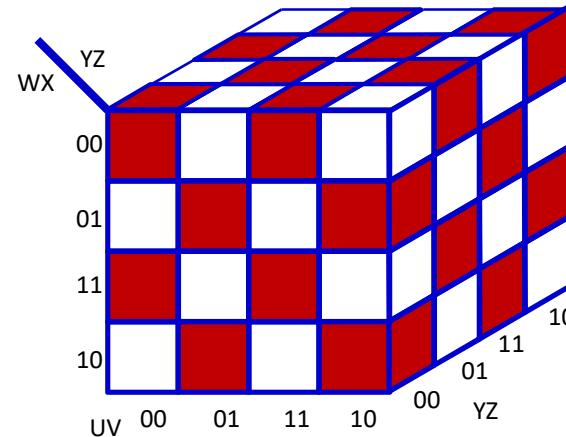
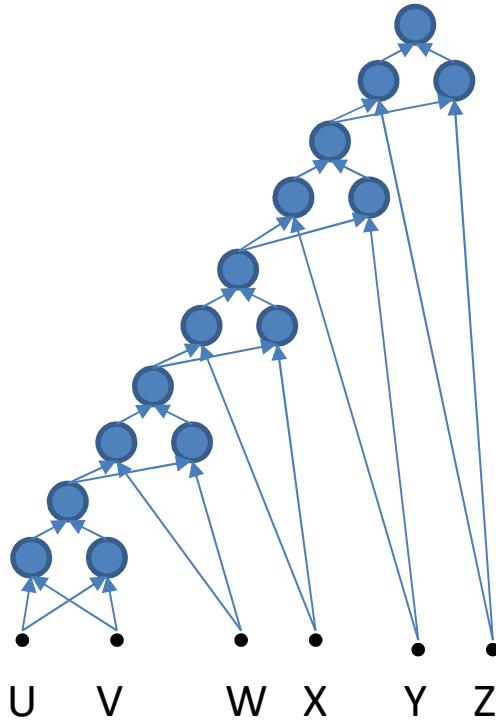


$$O = U \oplus V \oplus W \oplus X \oplus Y \oplus Z$$

15 perceptrons

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

Width of a deep MLP

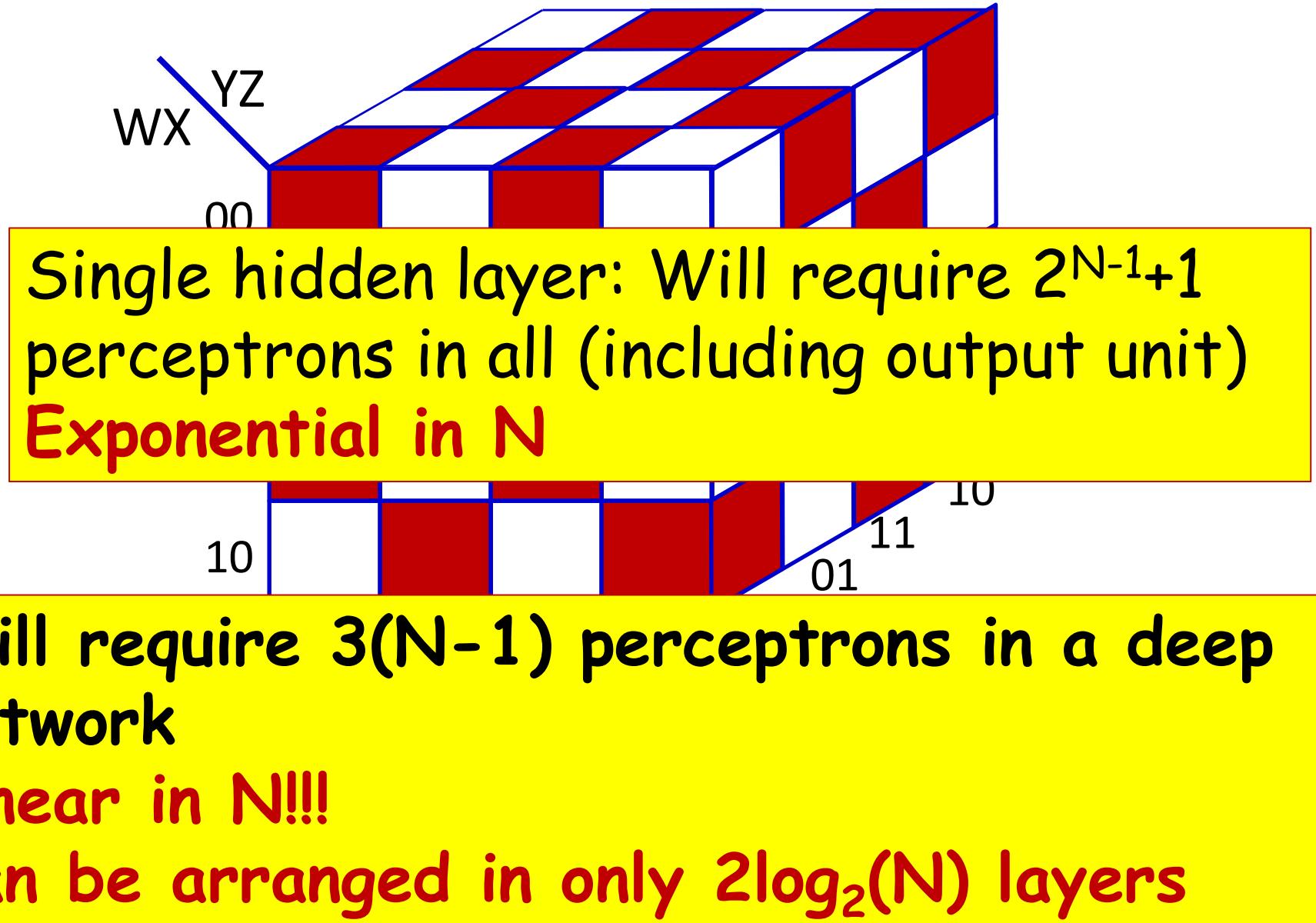


$$O = U \oplus V \oplus W \oplus X \oplus Y \oplus Z$$

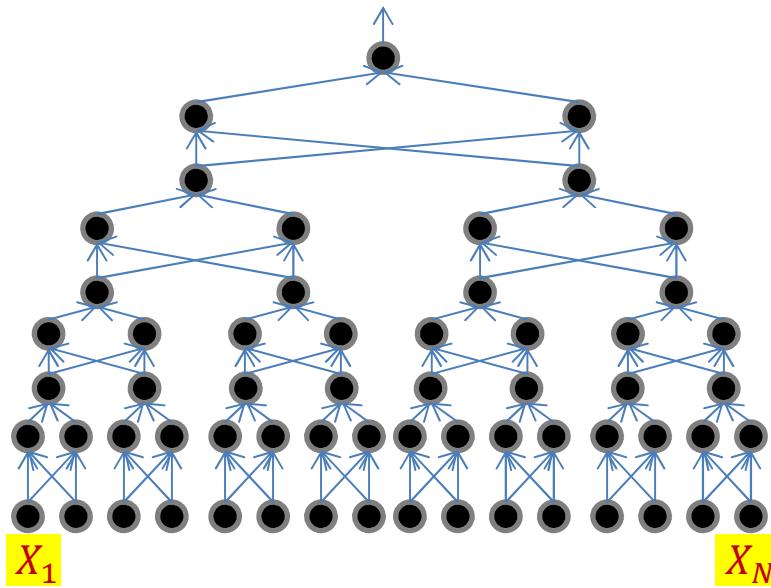
More generally, the XOR of N variables will require $3(N-1)$ perceptrons!!

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

Width of a single-layer Boolean MLP



A better representation

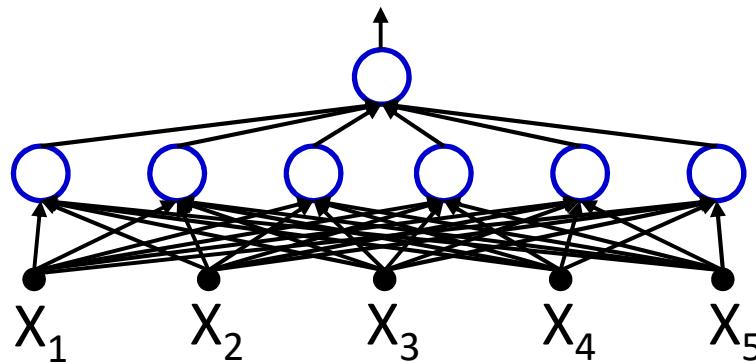


$$O = X_1 \oplus X_2 \oplus \cdots \oplus X_N$$

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

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

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 or *superexponential* number of weights..

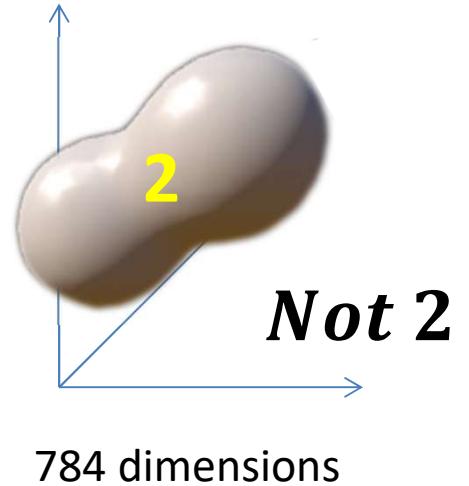
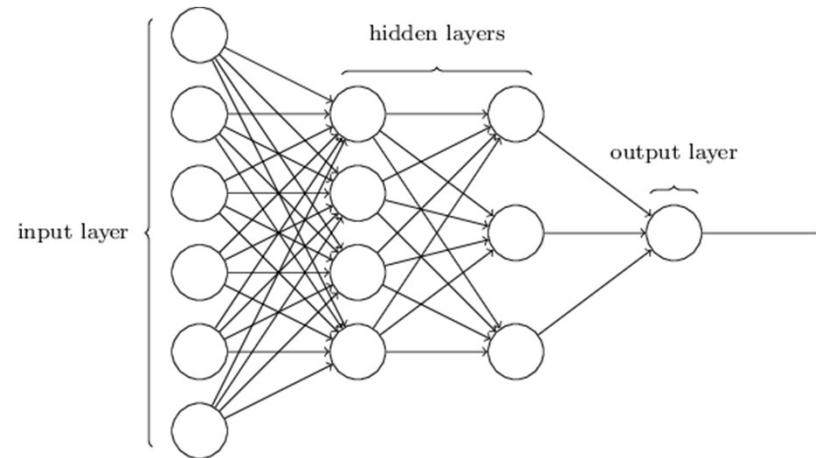
Network size: summary

- 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

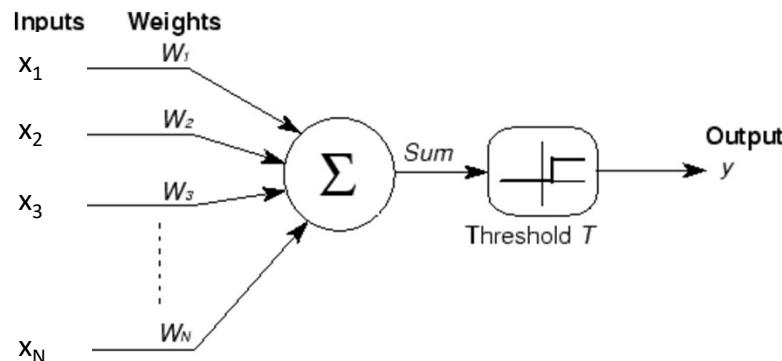
- Multi-layer perceptrons are *Universal Boolean Machines*
- Even a network with a *single* hidden layer is a universal Boolean machine
 - But a single-layer network may require an exponentially large number of perceptrons
- Deeper networks may require far fewer neurons than shallower networks to express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

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*

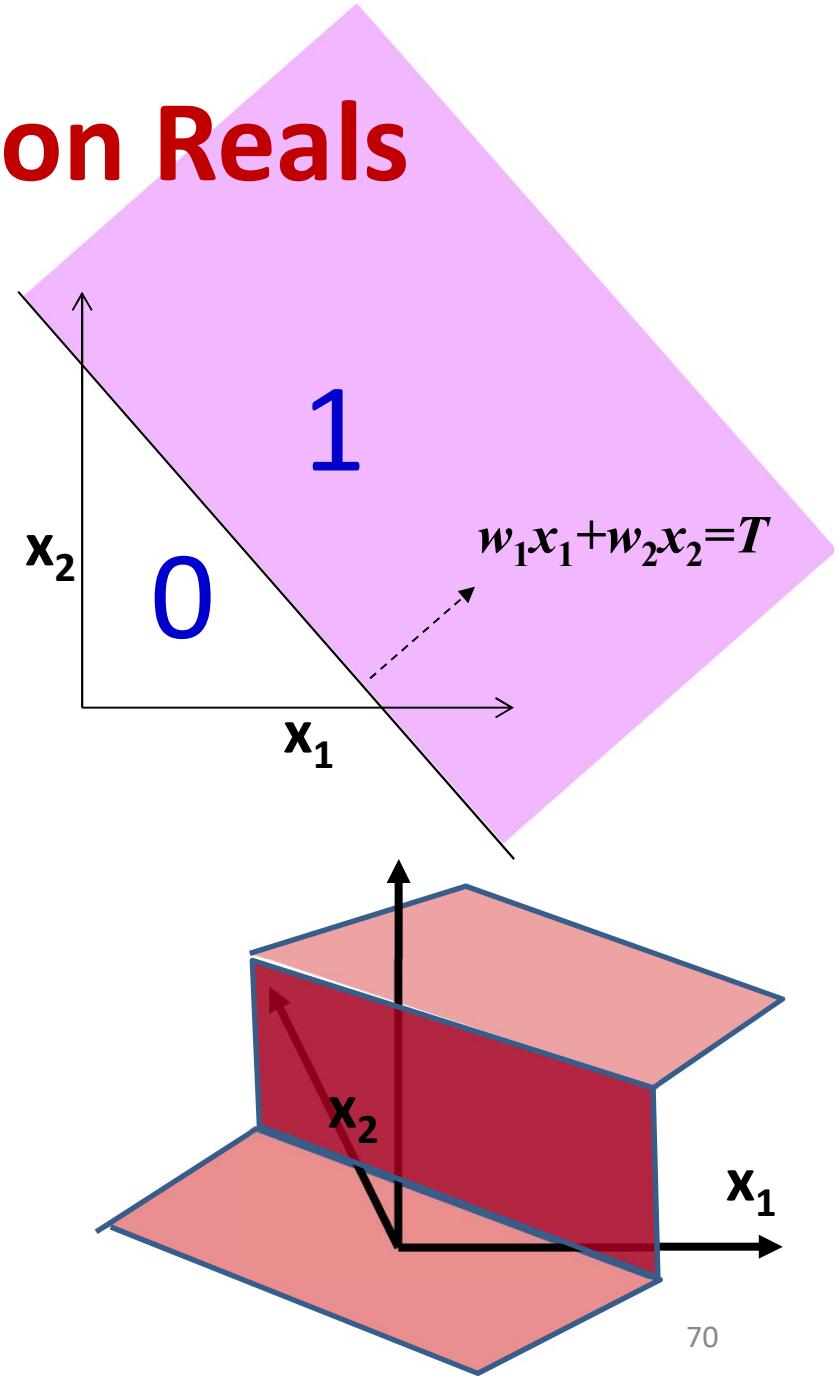
A Perceptron on Reals



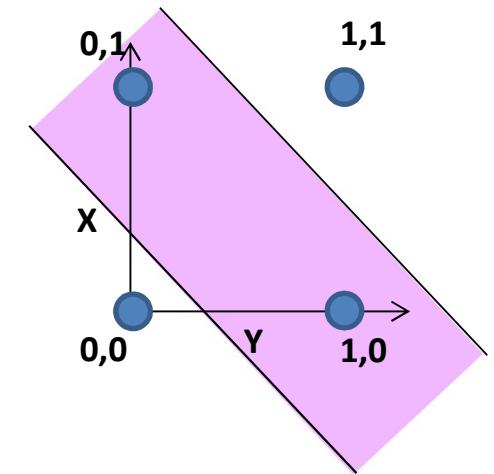
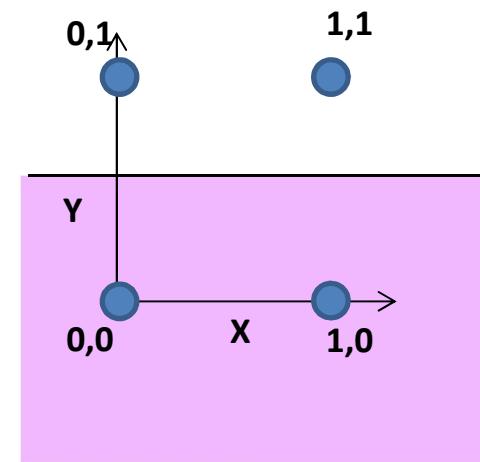
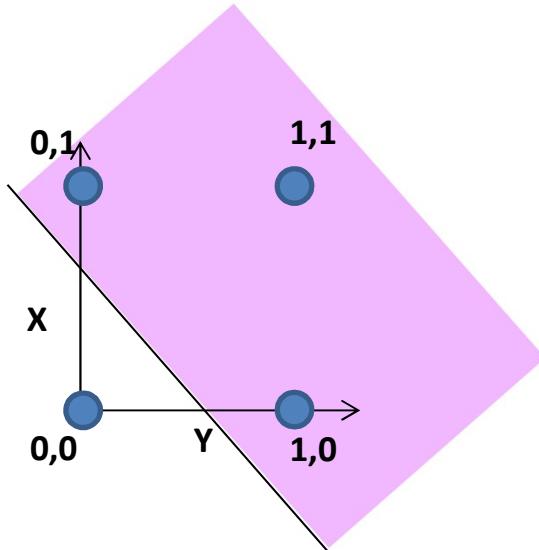
$$y = \begin{cases} 1 & \text{if } \sum_i w_i x_i \geq T \\ 0 & \text{else} \end{cases}$$

- A perceptron operates on *real-valued* vectors

— This is a *linear classifier*

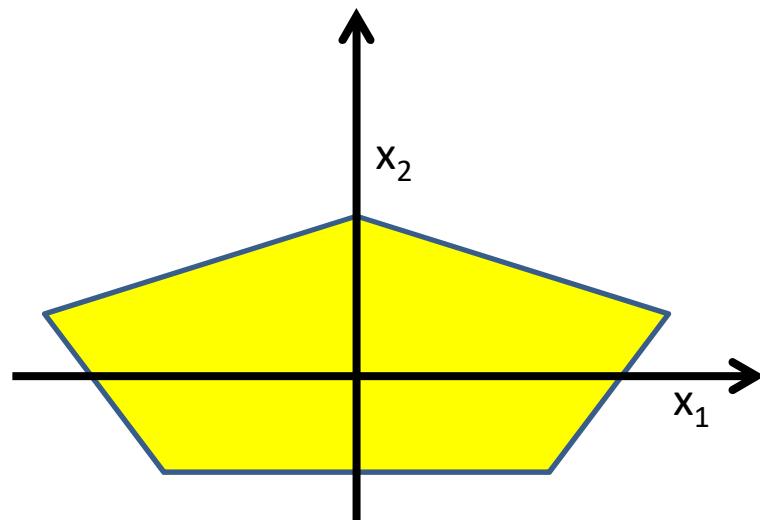


Boolean functions with a real perceptron



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

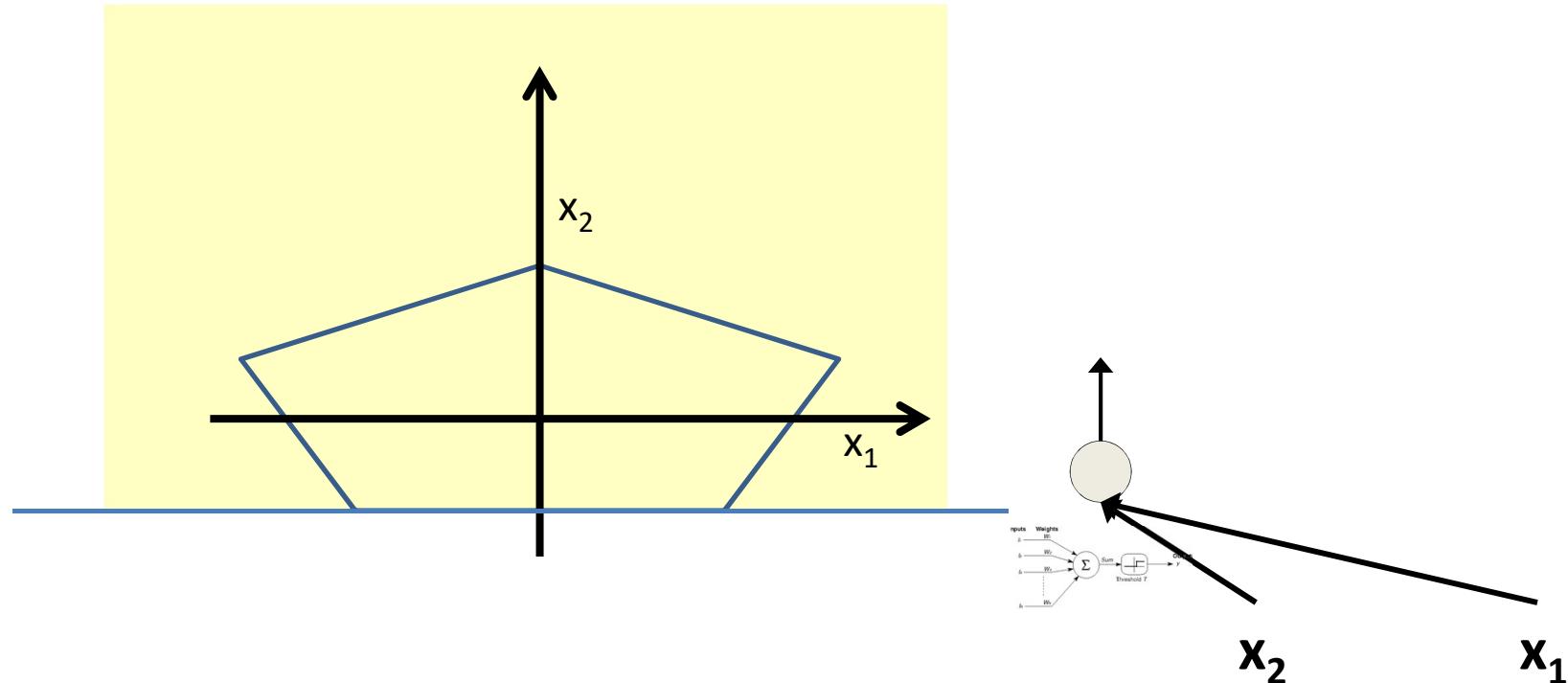
Composing complicated “decision” boundaries



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

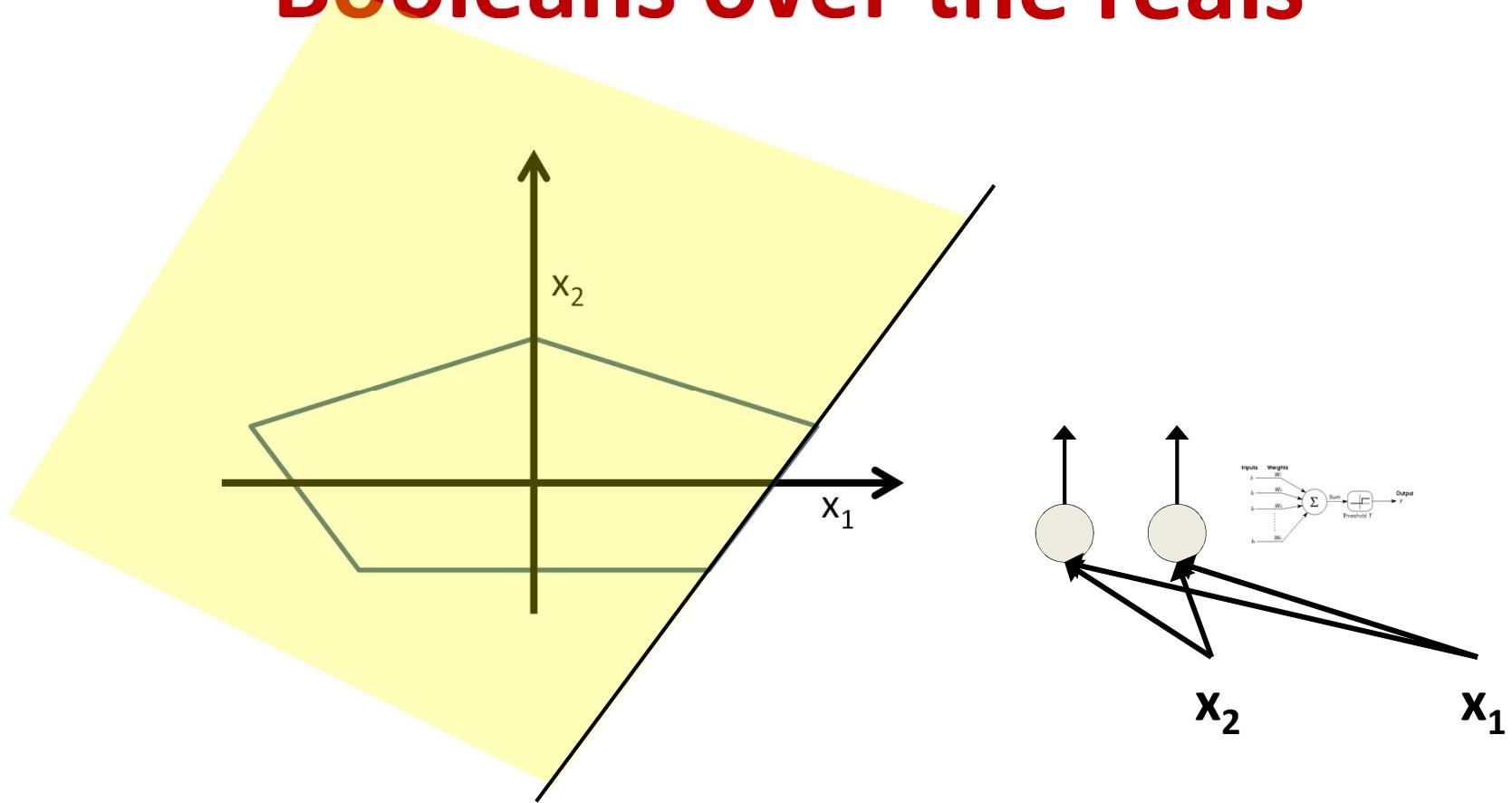
- Build a network of units with a single output that fires if the input is in the coloured area

Booleans over the reals



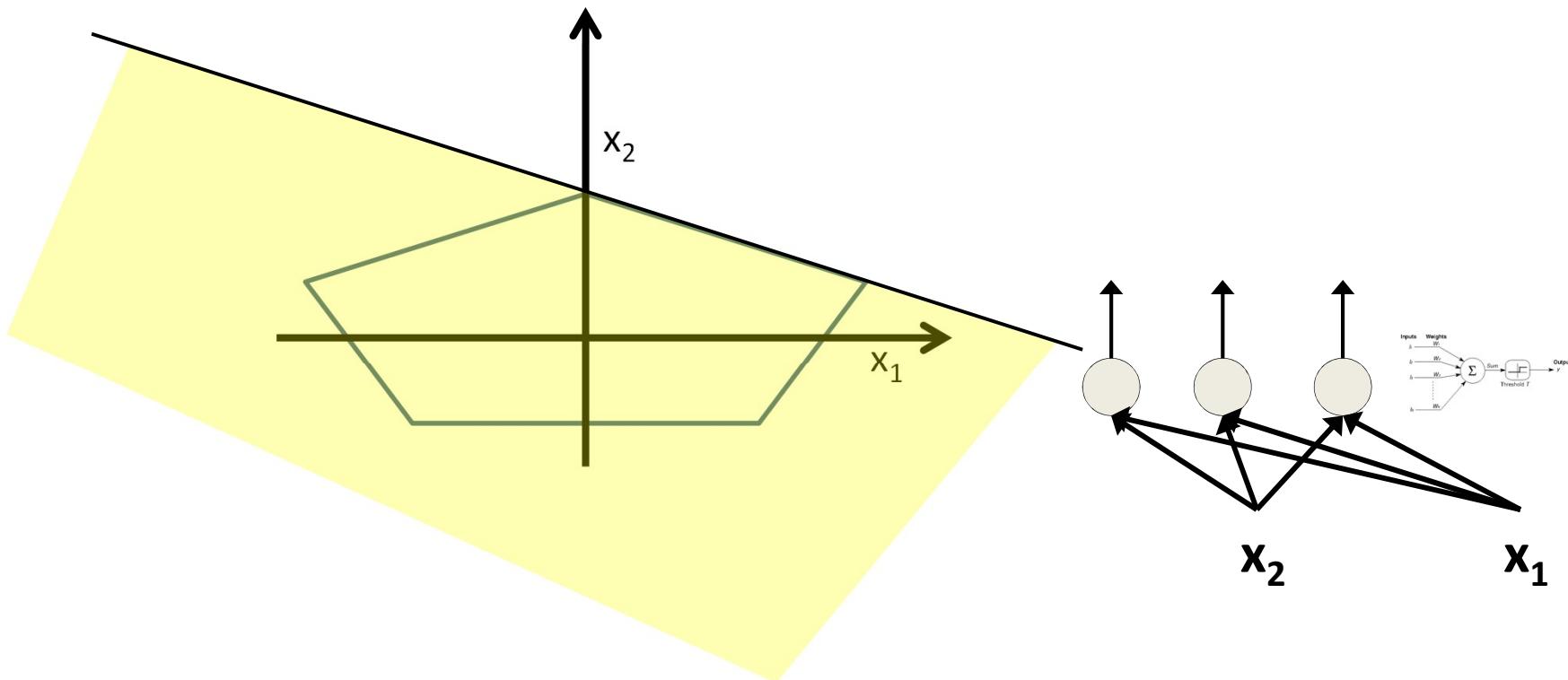
- The network must fire if the input is in the coloured area

Booleans over the reals



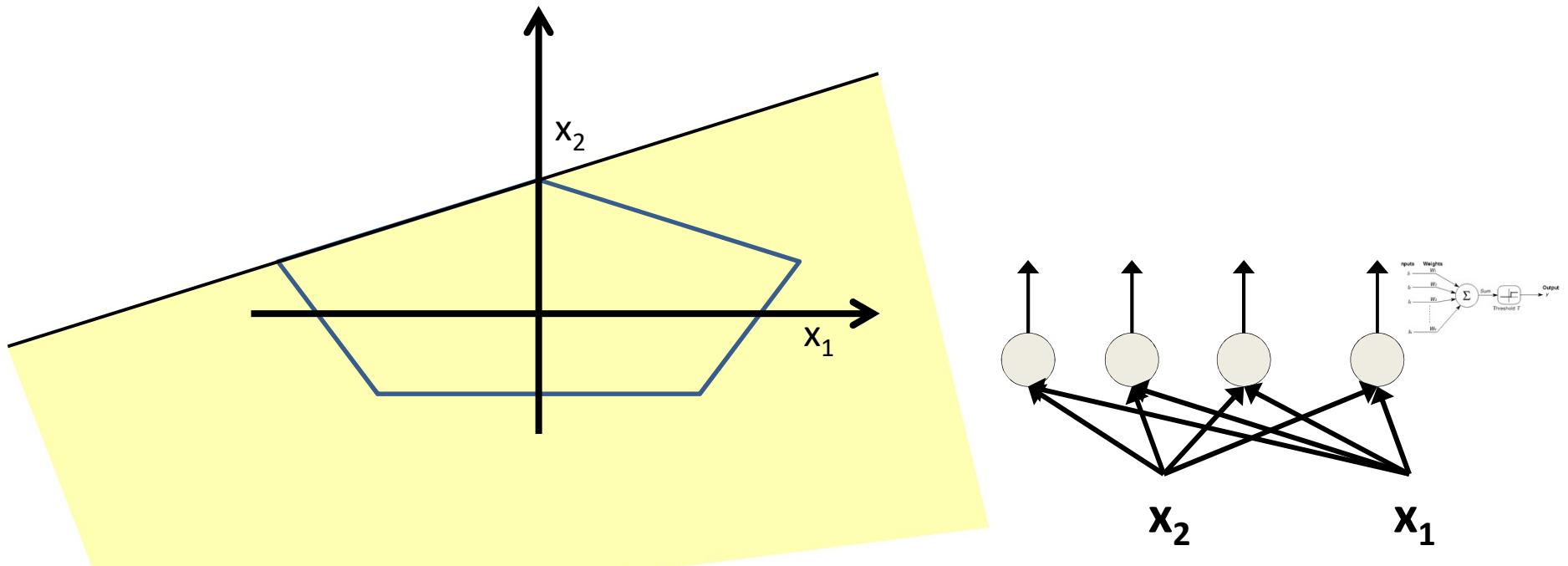
- The network must fire if the input is in the coloured area

Booleans over the reals



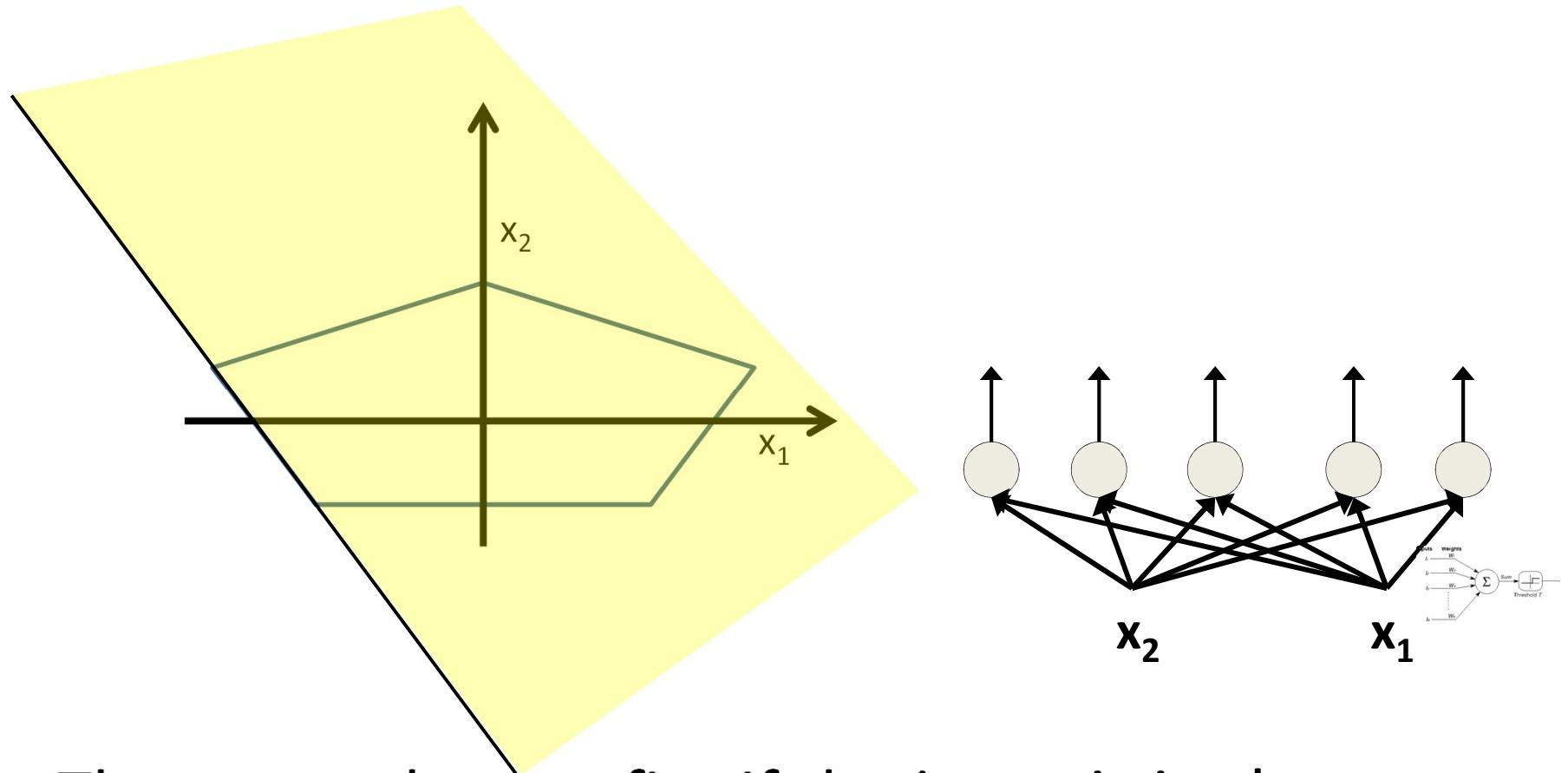
- The network must fire if the input is in the coloured area

Booleans over the reals



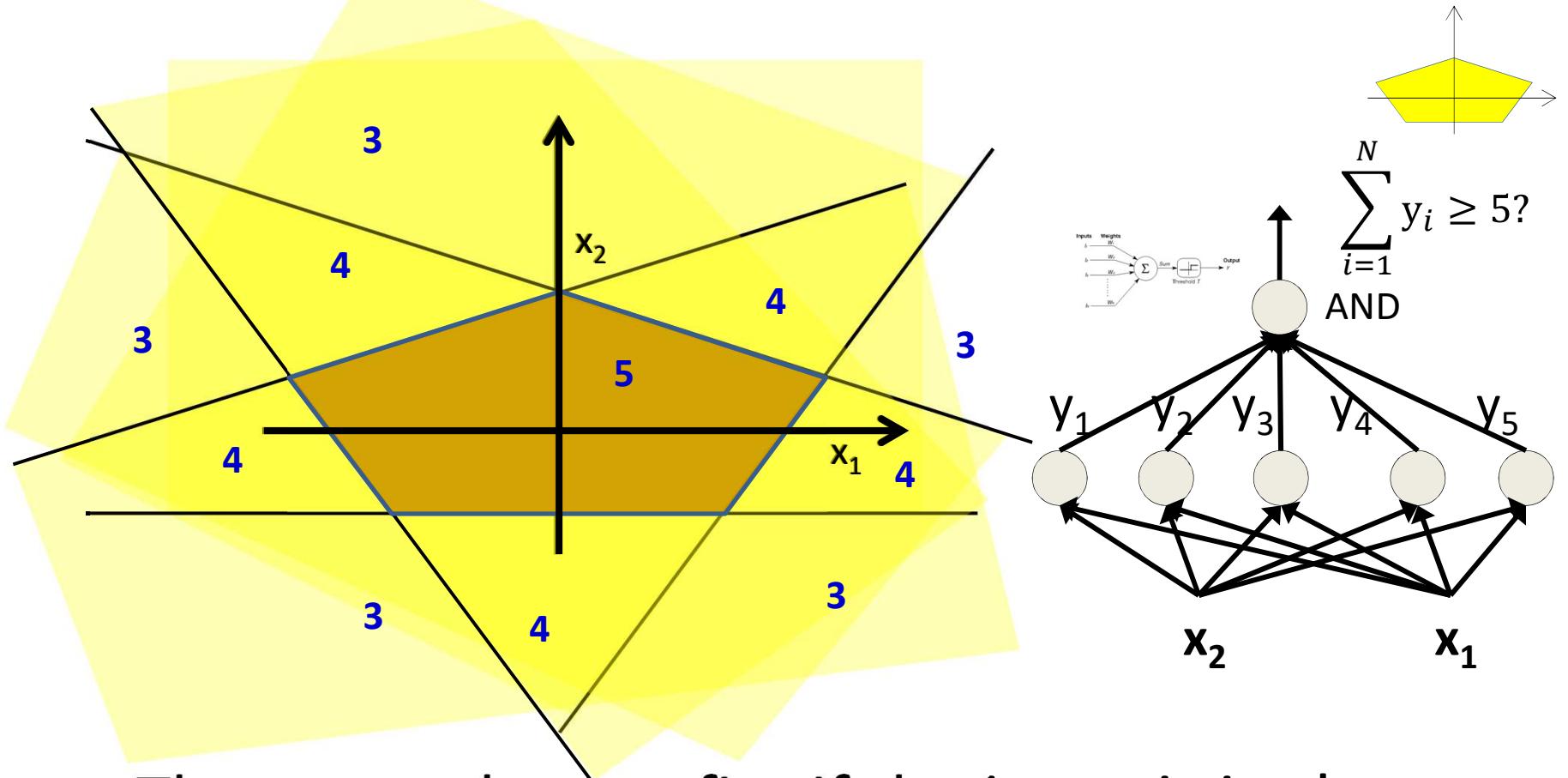
- The network must fire if the input is in the coloured area

Booleans over the reals



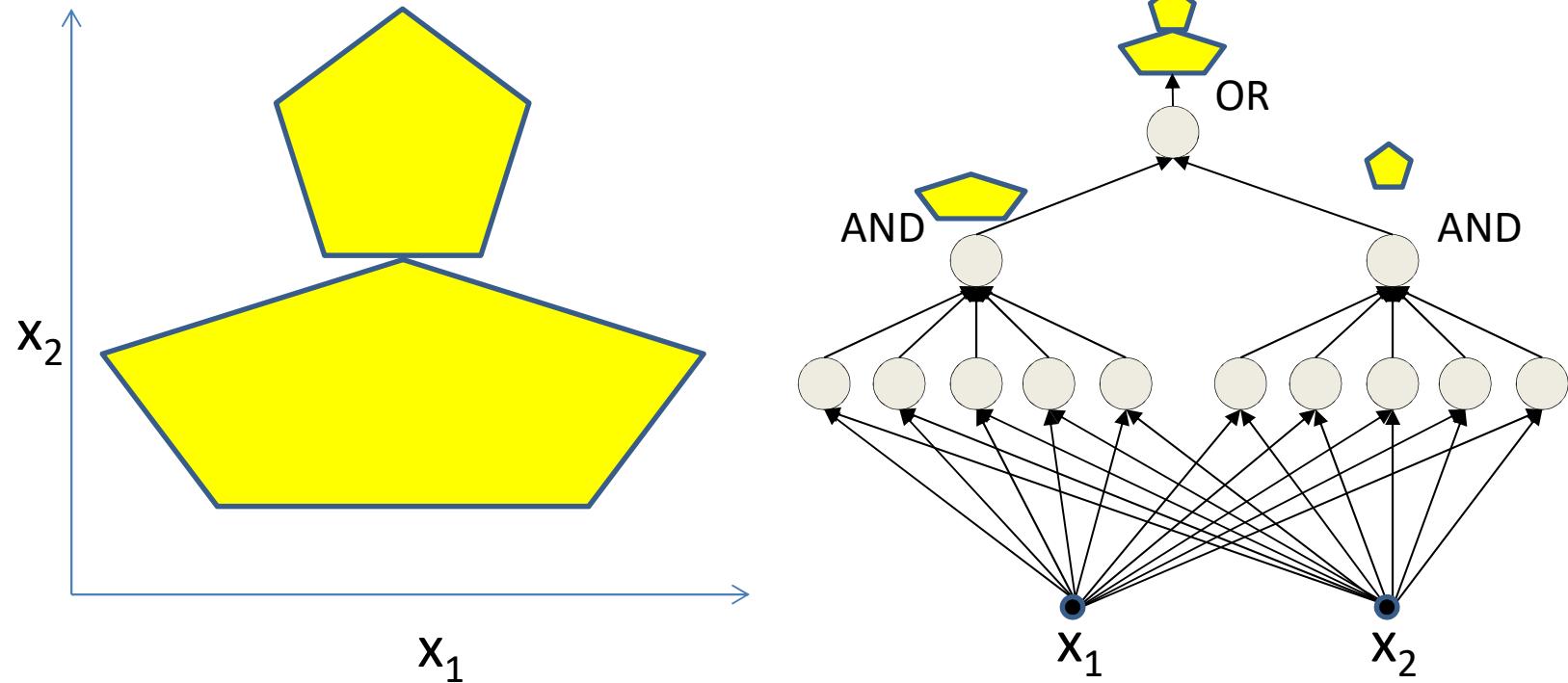
- The network must fire if the input is in the coloured area

Booleans over the reals



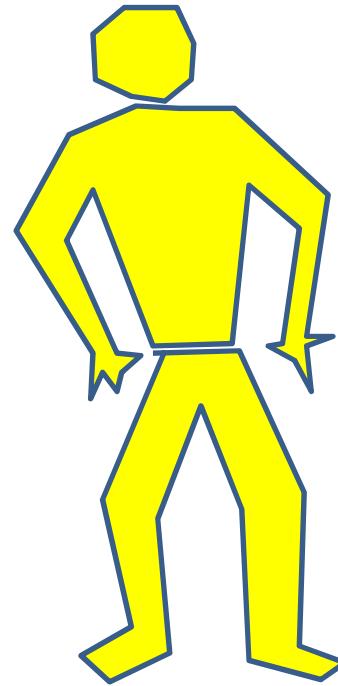
- The network must fire if the input is in the coloured area

More complex decision boundaries



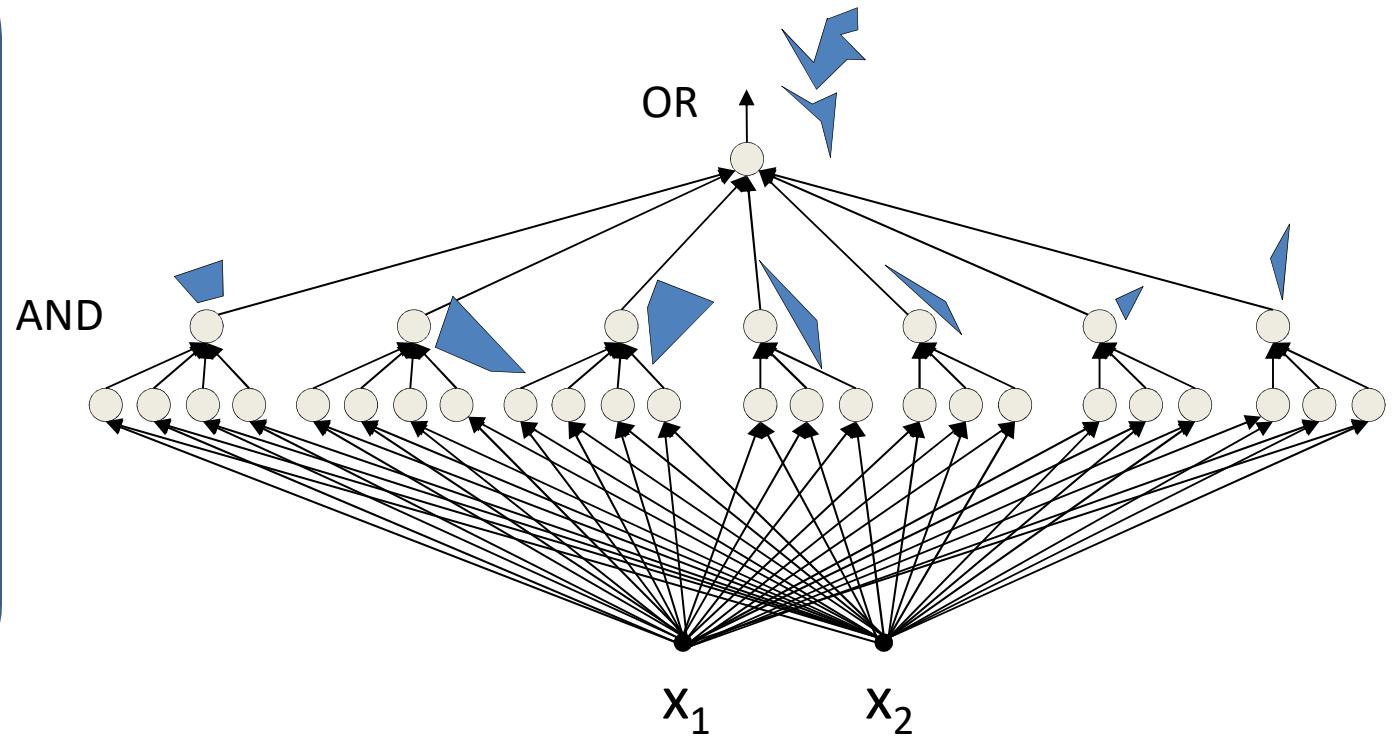
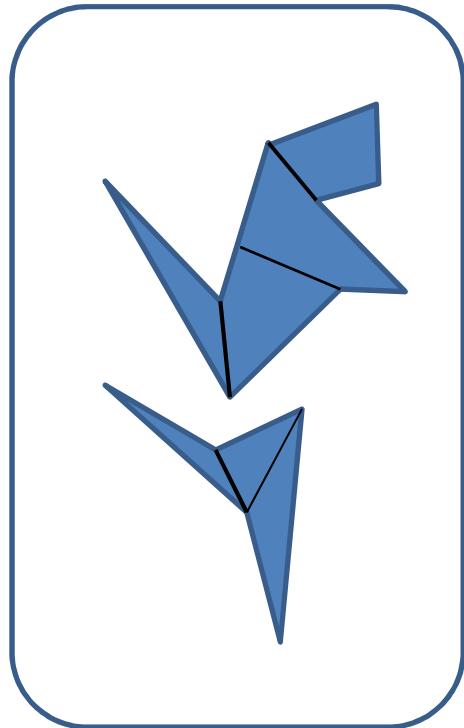
- Network to fire if the input is in the yellow area
 - “OR” two polygons
 - A third layer is required

Complex decision boundaries



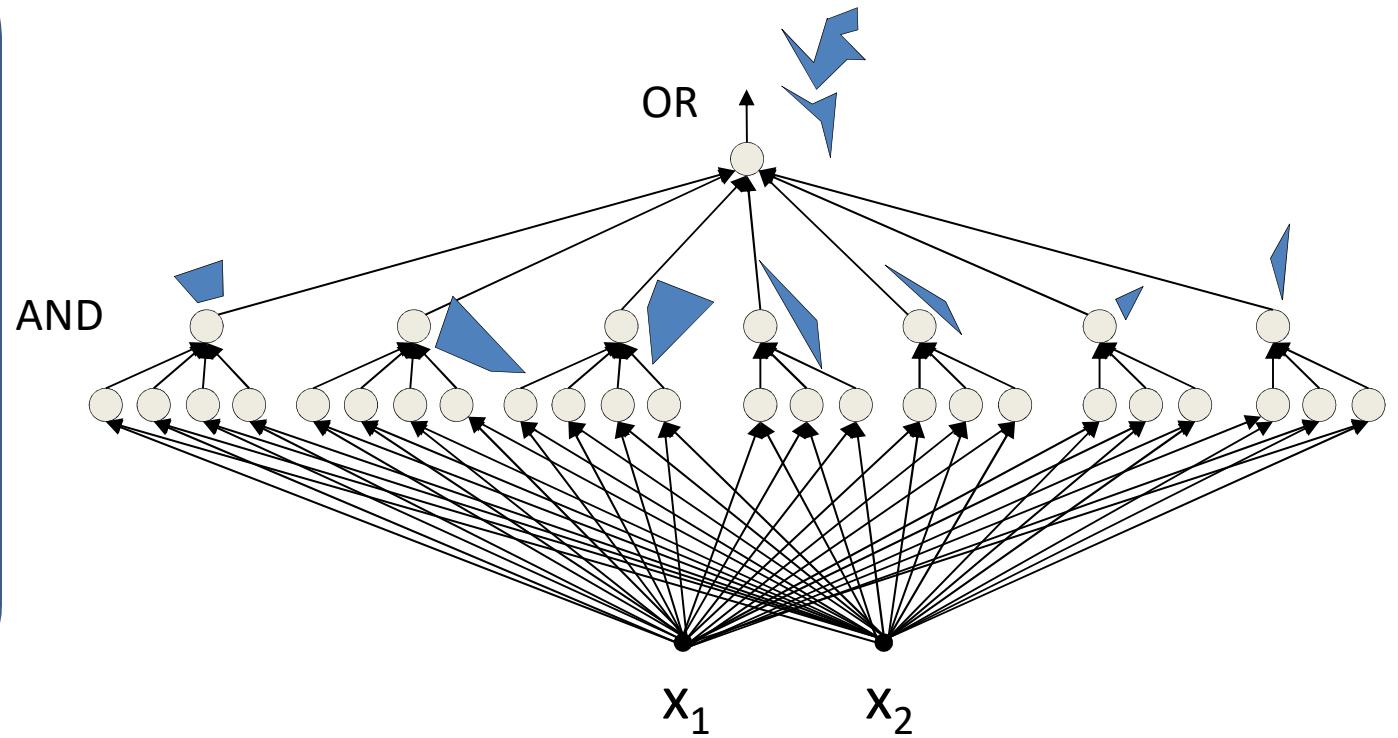
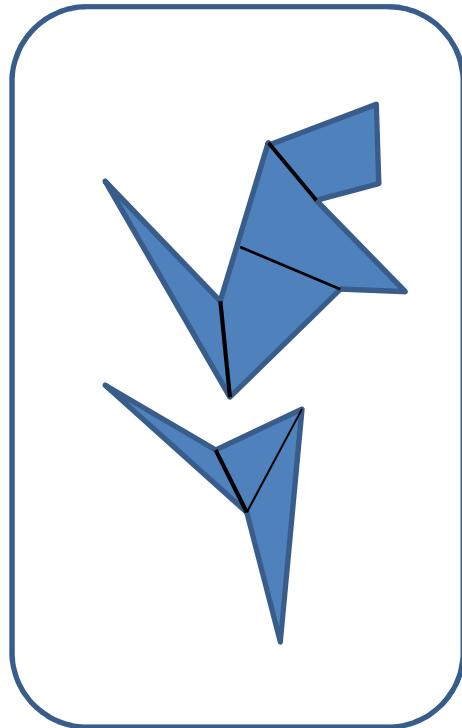
- Can compose *arbitrarily* complex decision boundaries

Complex decision boundaries



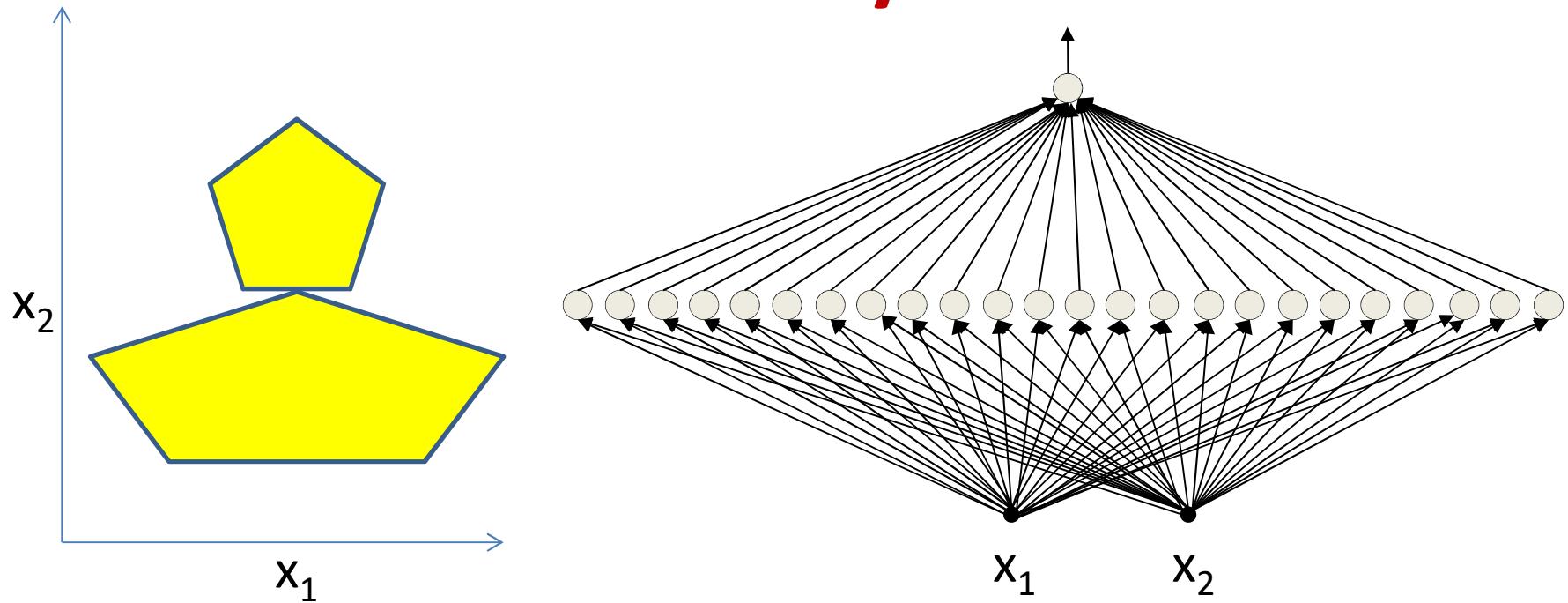
- Can compose *arbitrarily* complex decision boundaries

Complex decision boundaries



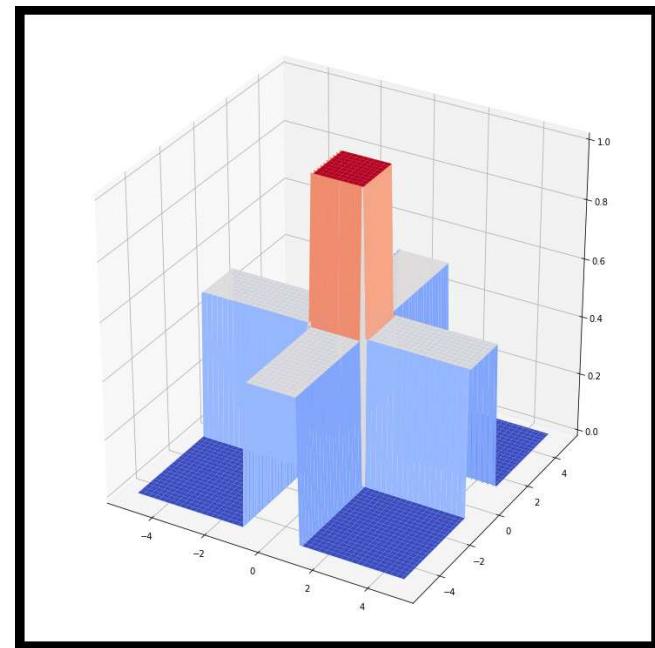
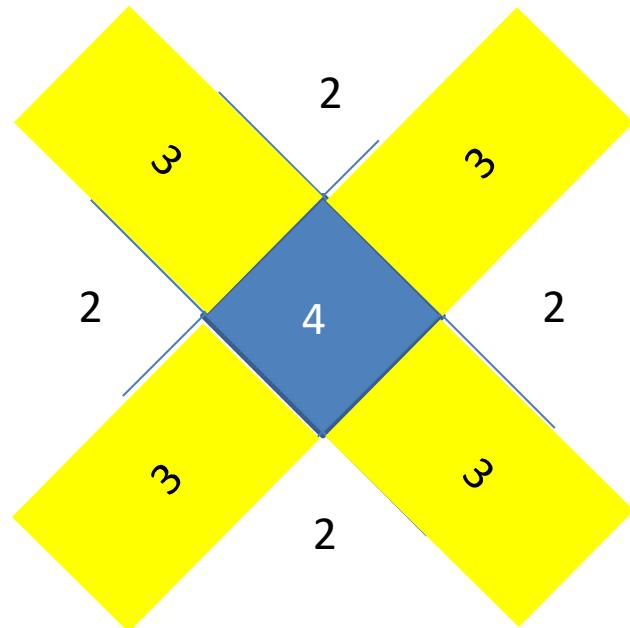
- Can compose *arbitrarily* complex decision boundaries
 - With *only one hidden layer!*
 - **How?**

Exercise: compose this with one hidden layer

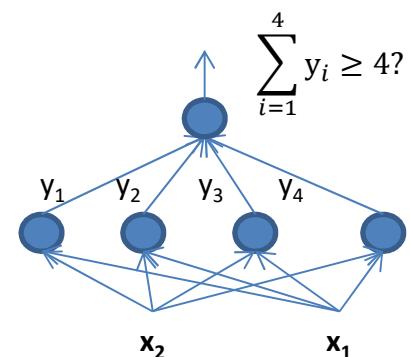


- How would you compose the decision boundary to the left with only *one* hidden layer?

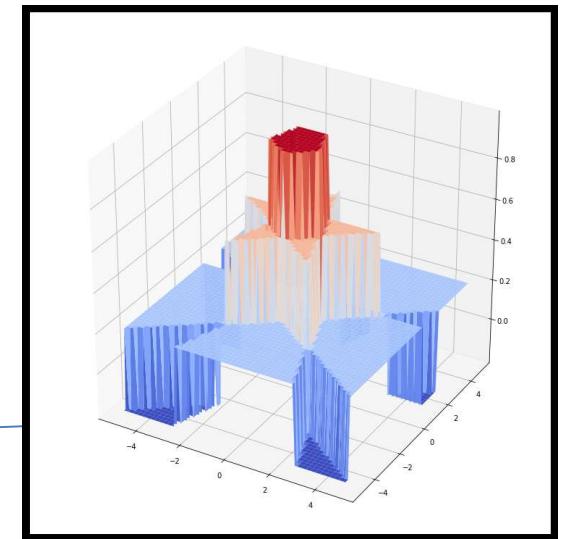
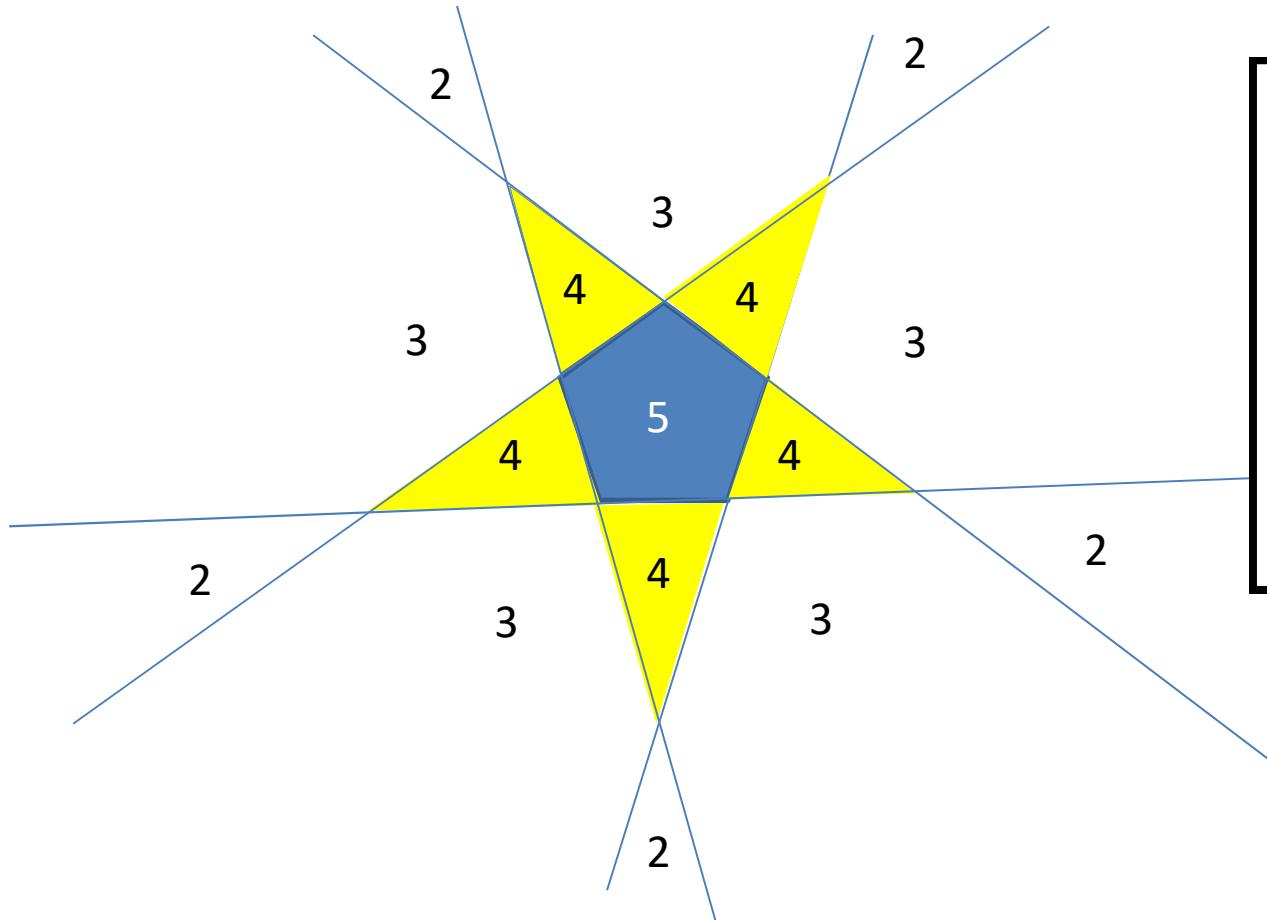
Composing a Square decision boundary



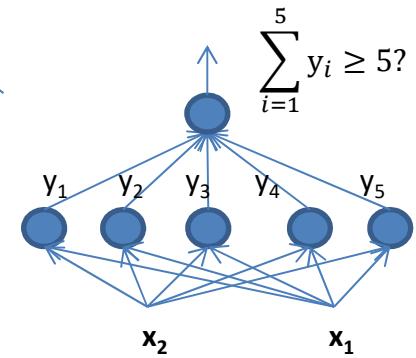
- The polygon net



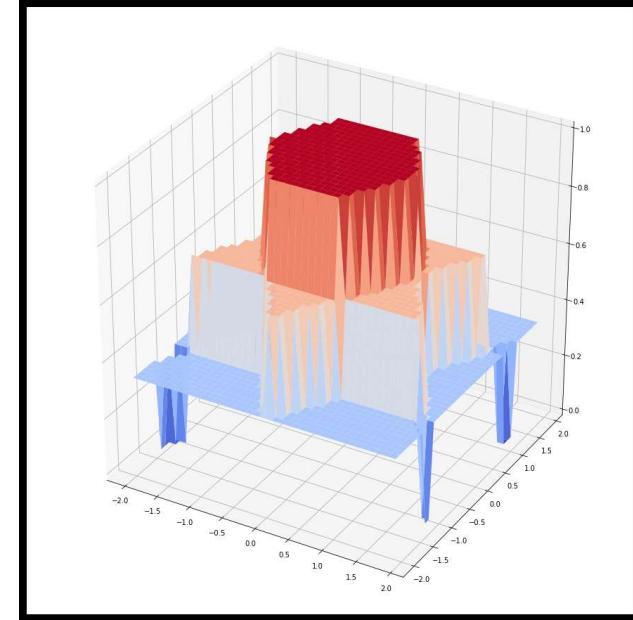
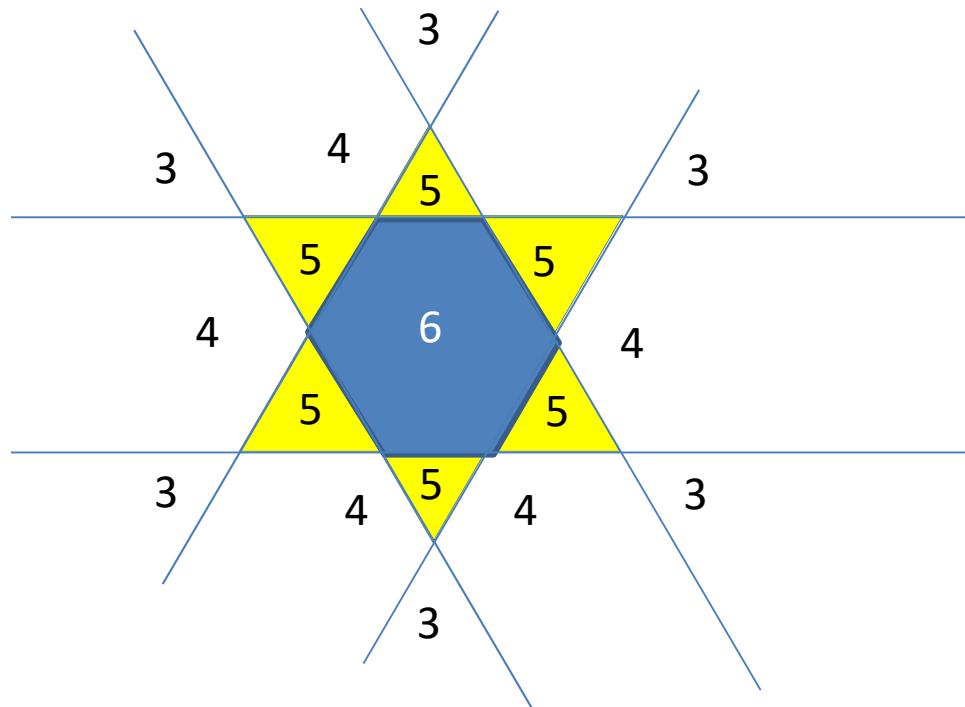
Composing a pentagon



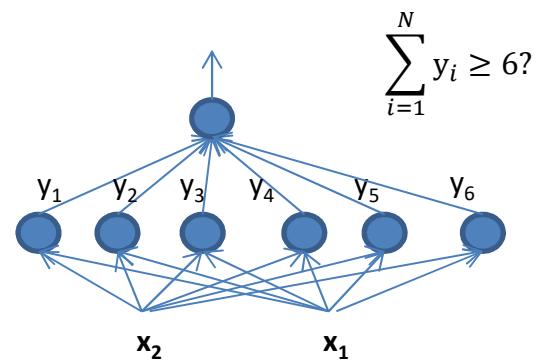
- The polygon net



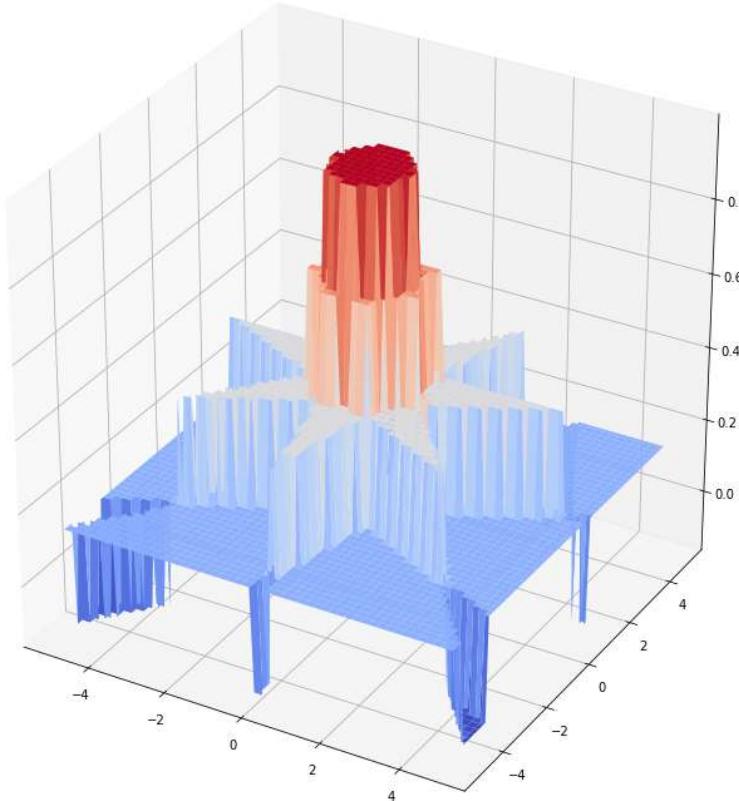
Composing a hexagon



- The polygon net

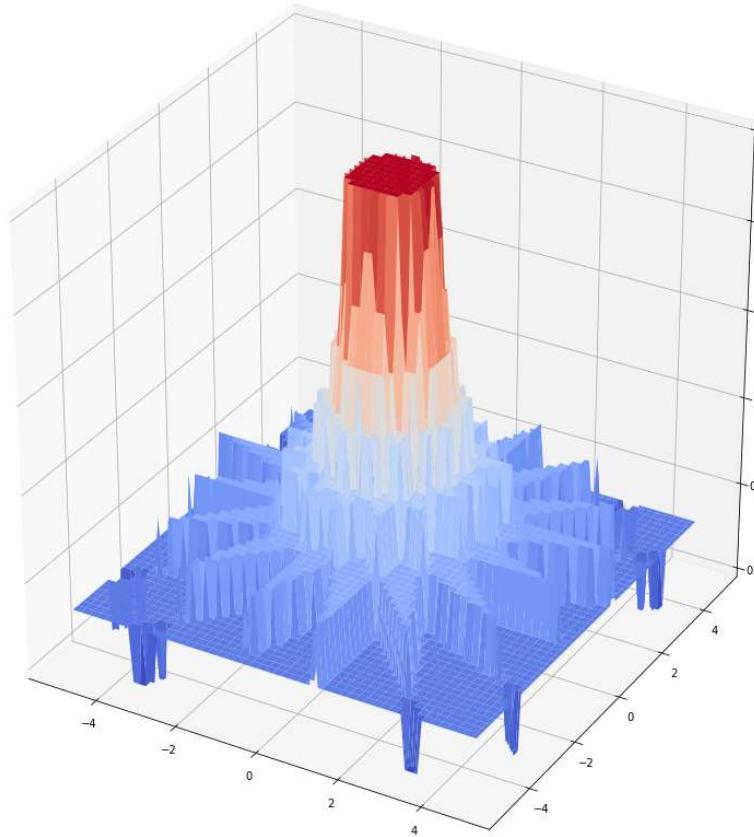


How about a heptagon



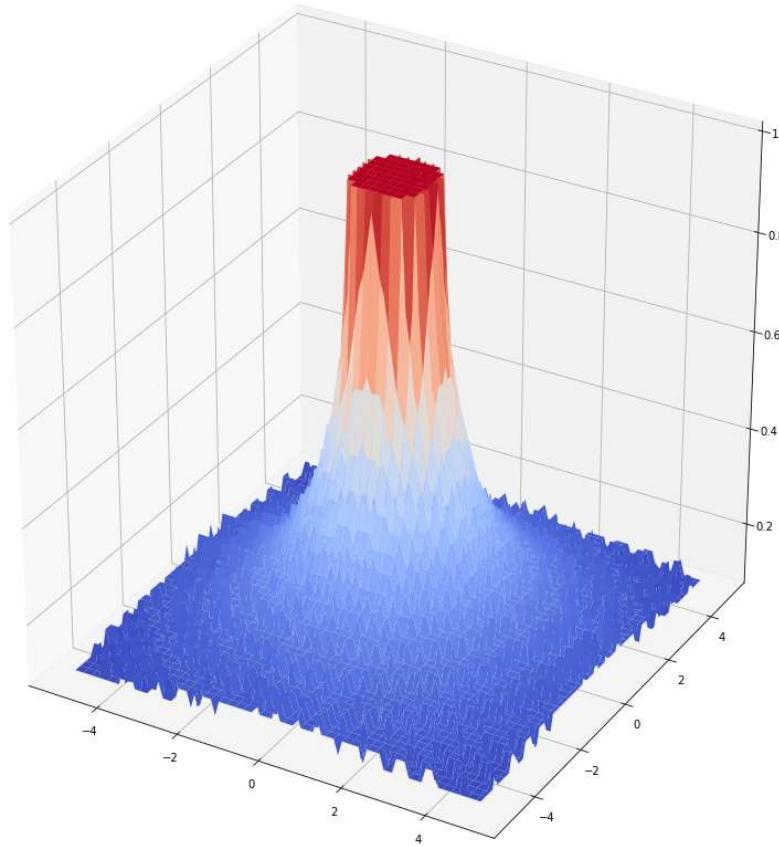
- What are the sums in the different regions?
 - A pattern emerges as we consider $N > 6..$

16 sides



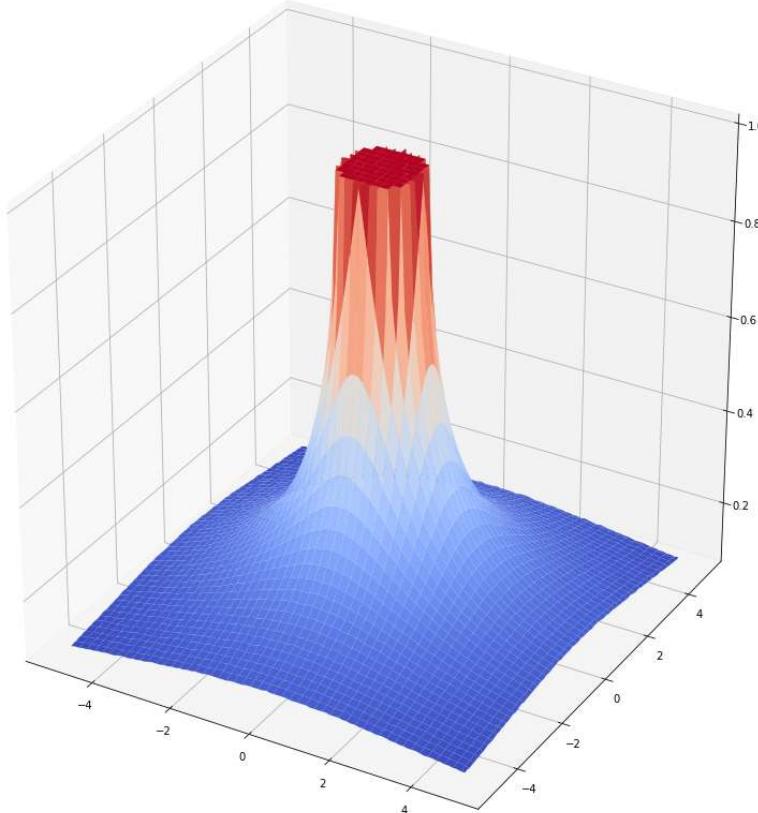
- What are the sums in the different regions?
 - A pattern emerges as we consider $N > 6..$

64 sides



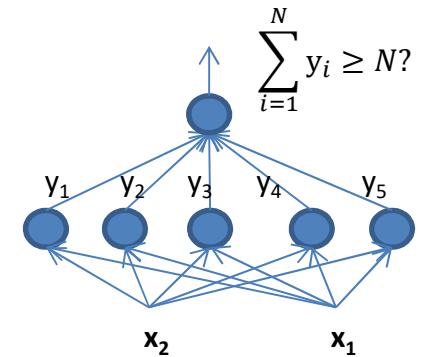
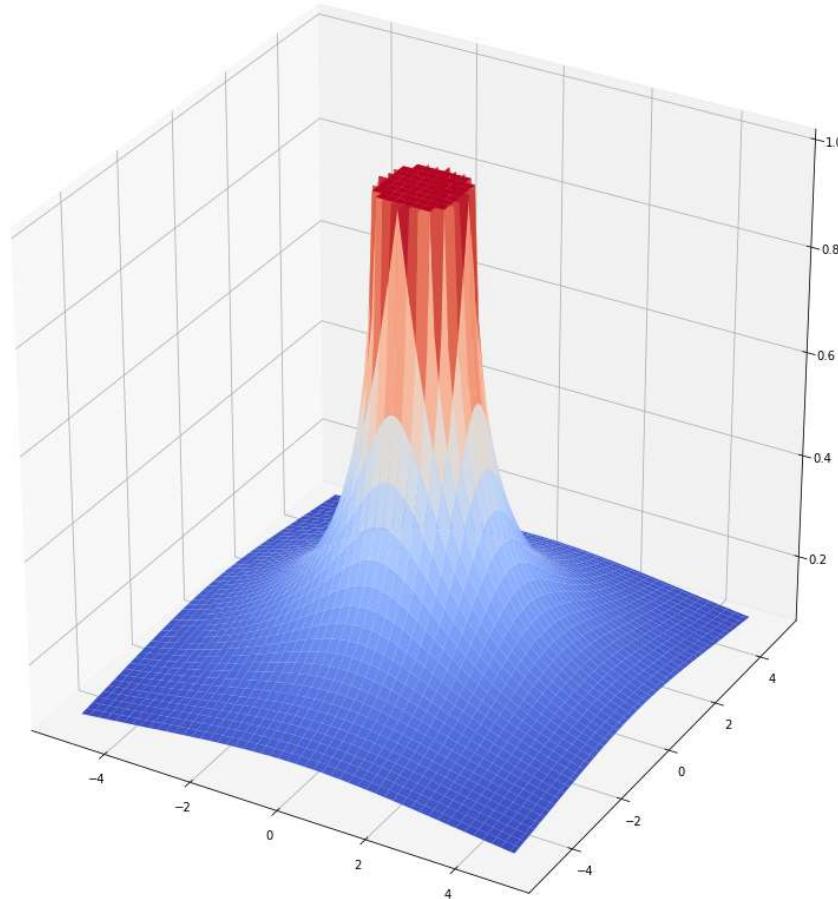
- What are the sums in the different regions?
 - A pattern emerges as we consider $N > 6..$

1000 sides



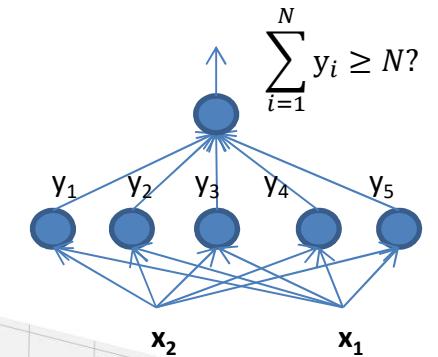
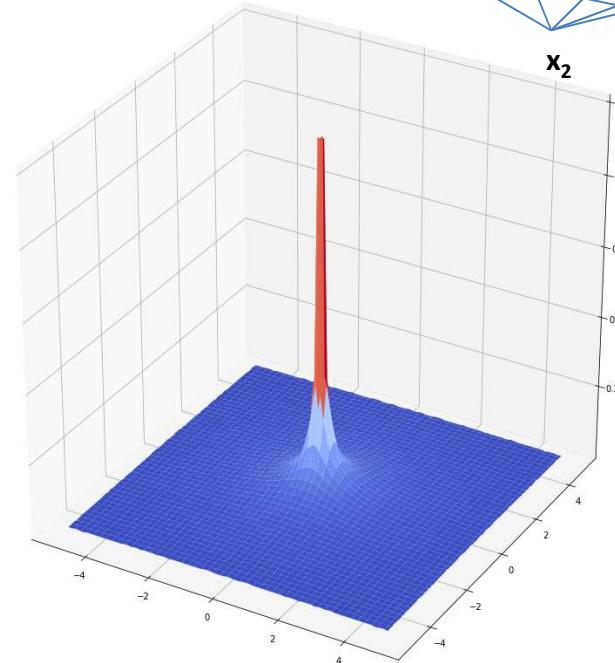
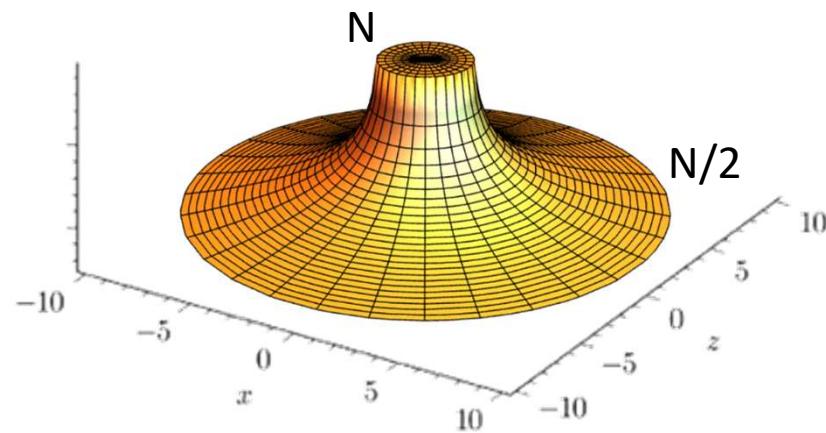
- What are the sums in the different regions?
 - A pattern emerges as we consider $N > 6..$

Polygon net



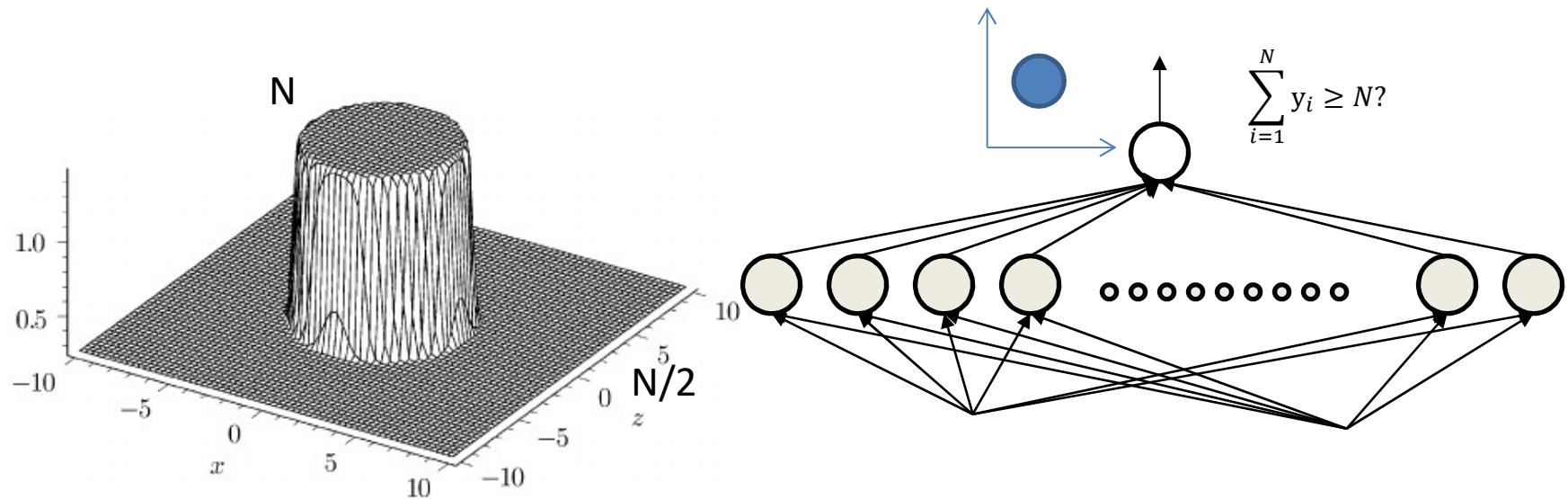
- Increasing the number of sides reduces the area outside the polygon that have $N/2 < \text{Sum} < N$

In the limit



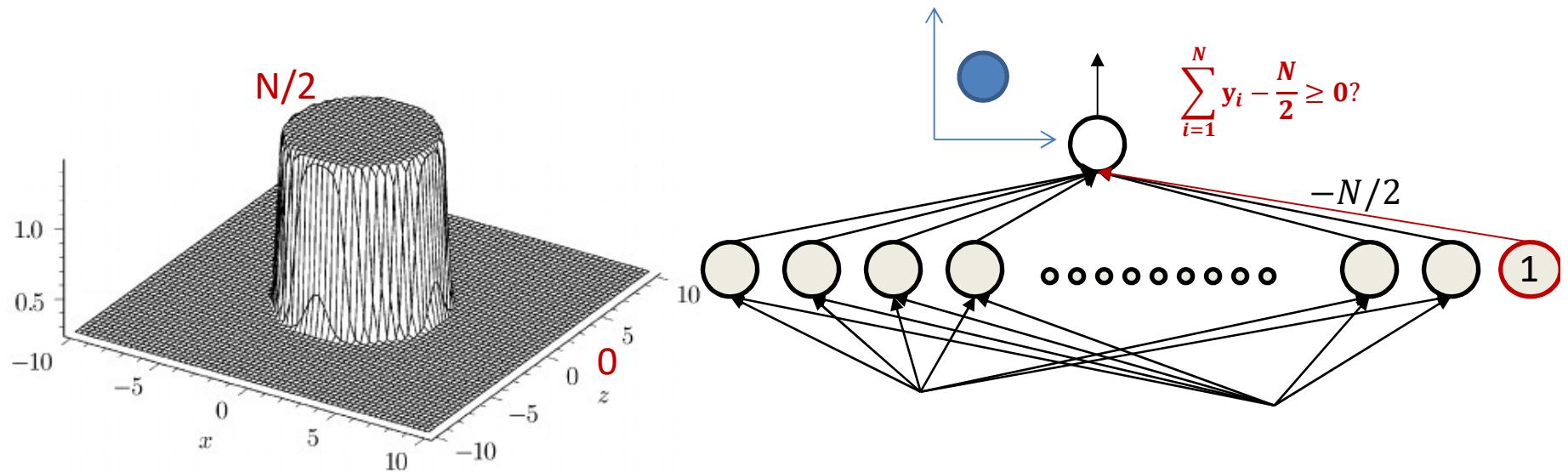
- $\sum_i y_i = N \left(1 - \frac{1}{\pi} \arccos \left(\min \left(1, \frac{\text{radius}}{|\mathbf{x}-\text{cent}|} \right) \right) \right)$
- For small radius, it's a near perfect cylinder
 - N in the cylinder, N/2 outside

Composing a circle



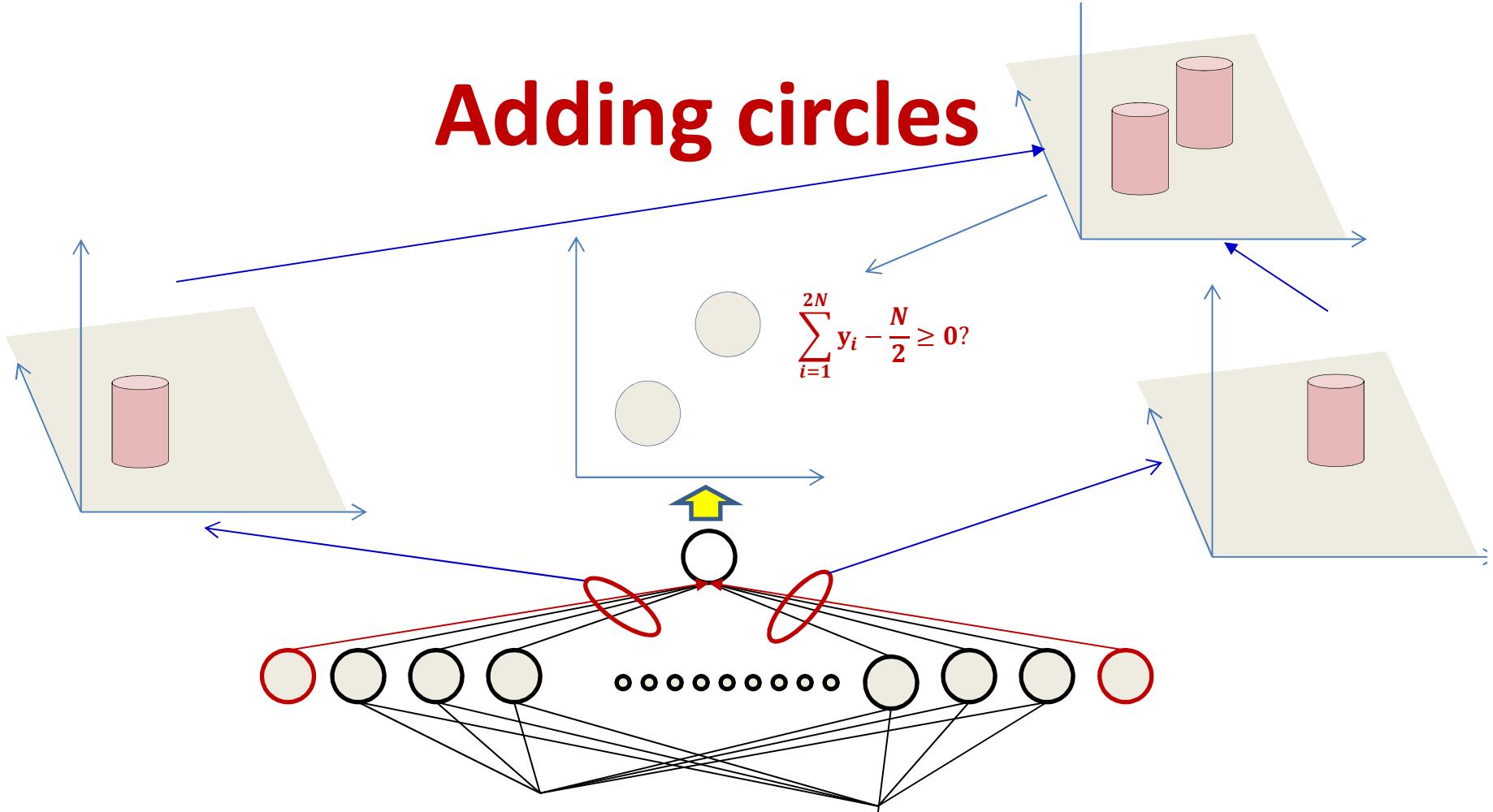
- The circle net
 - Very large number of neurons
 - *Sum is N inside the circle, $N/2$ outside almost everywhere*
 - Circle can be at any location

Composing a circle



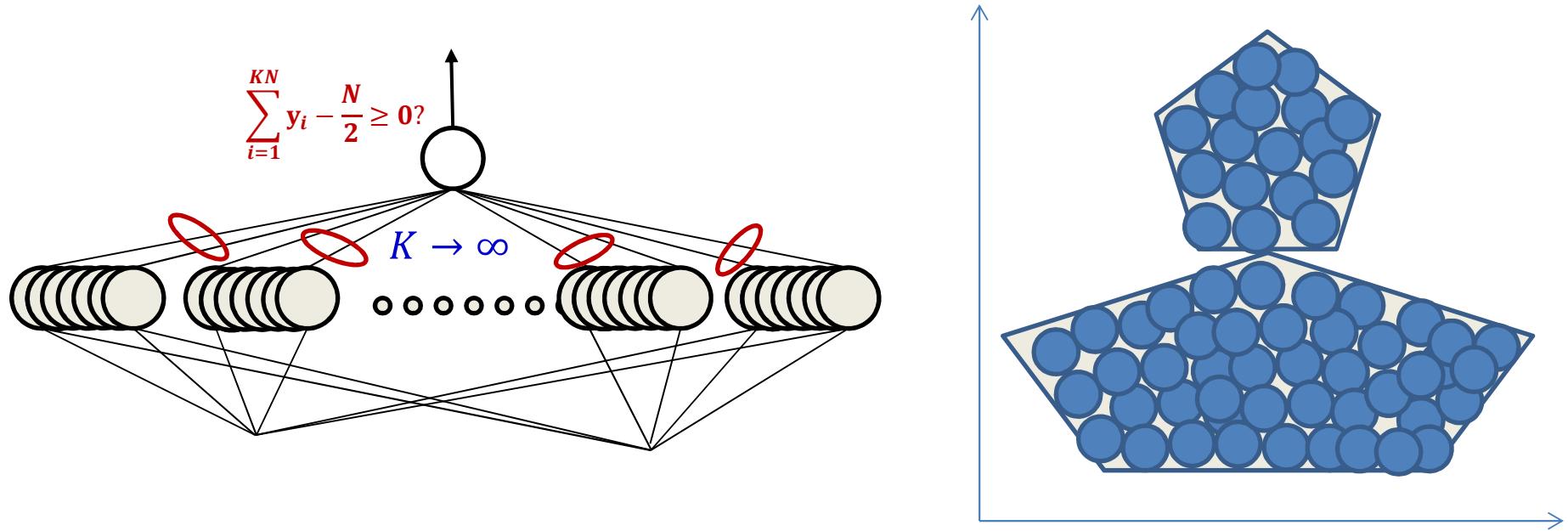
- The circle net
 - Very large number of neurons
 - *Sum is $N/2$ inside the circle, 0 outside almost everywhere*
 - Circle can be at any location

Adding circles



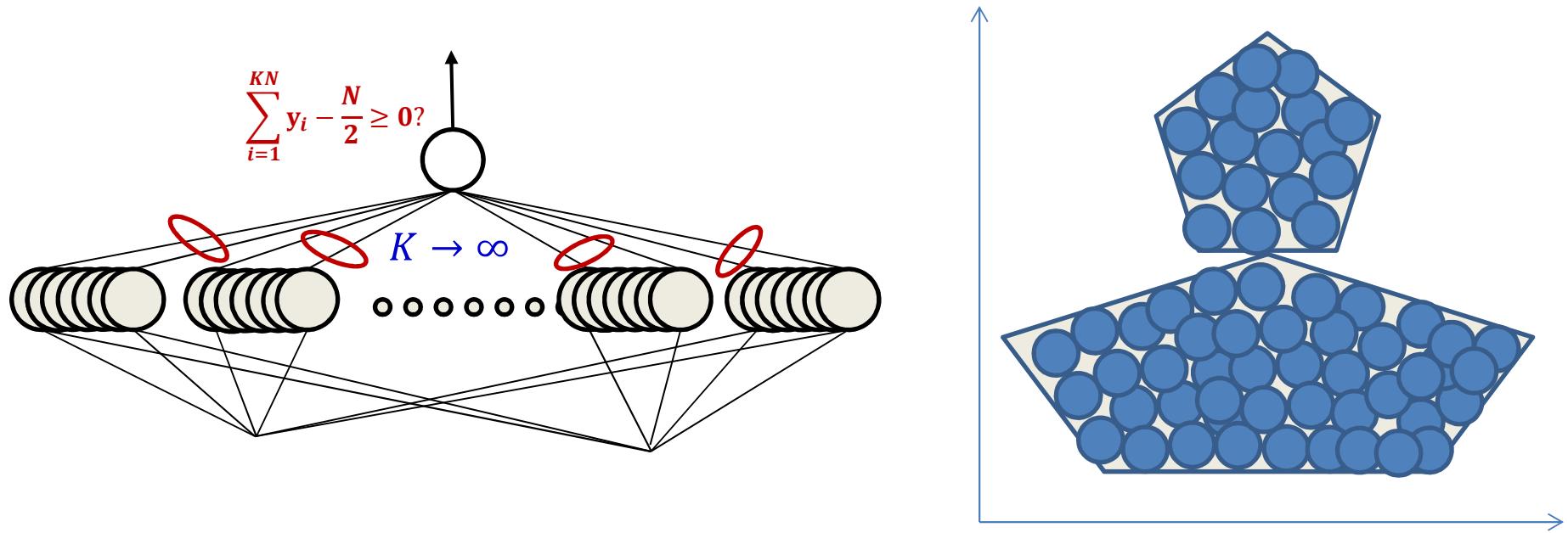
- The “sum” of two circles sub nets is exactly $N/2$ inside either circle, and 0 almost everywhere outside

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-layer MLP* can model any classification boundary
- *MLPs are universal classifiers*

Any **continuous function** defined in the **n-dimensional unit hypercube** may be approximated by a **finite sum** of the type:

$$\sum_{j=1}^N v_j \varphi \left(\overrightarrow{\omega}^{(j)} \cdot \overrightarrow{x} + b_j \right),$$

wherein $v_j, b_j \in \mathbb{R}$, $\overrightarrow{\omega}^{(j)} \in \mathbb{R}^n$, and φ is a **continuous discriminatory function**.

- Feedforward neural network models with one hidden layer of sigmoidal units implement functions on \mathbb{R}^d of the form

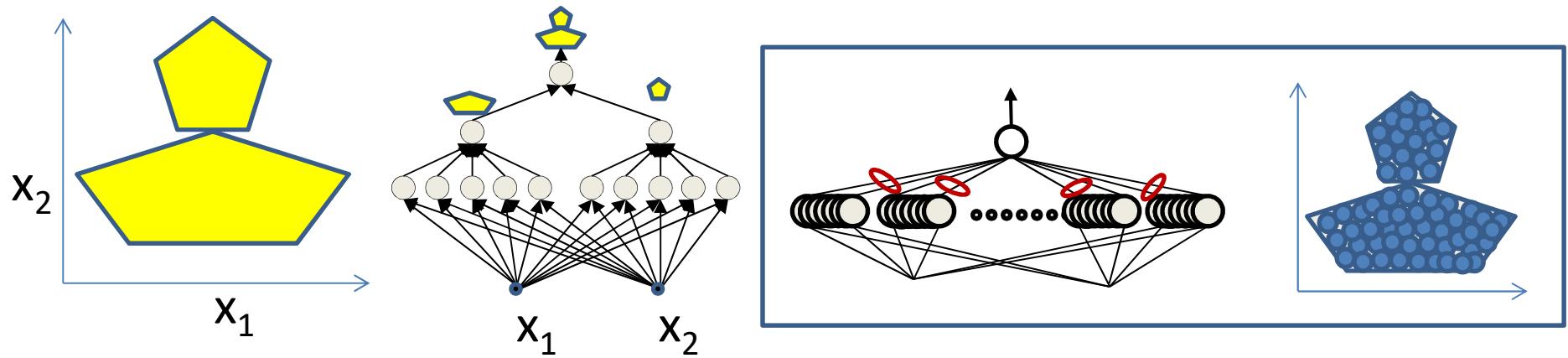
$$f_n(\mathbf{x}) = \sum_{i=1}^n c_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i) + c_0$$

- Universal Approximation Theorem**¹: For every function f and every $n \geq 1$, there exists a linear combination of sigmoidal functions $f_n(\mathbf{x})$, such that

$$\int_{B_r} (f(\mathbf{x}) - f_n(\mathbf{x}))^2 \mu(d\mathbf{x}) \leq \frac{c'_f}{n}$$

¹Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

Depth and the universal classifier

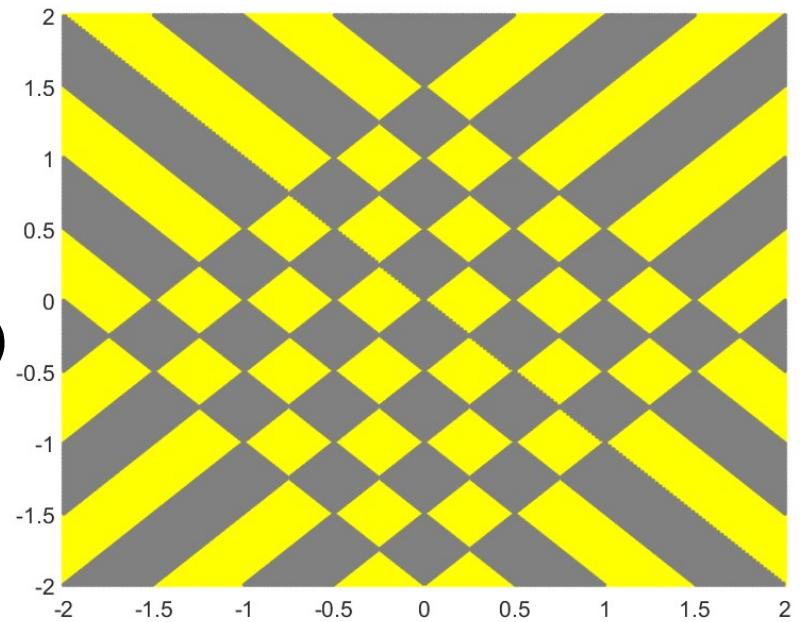
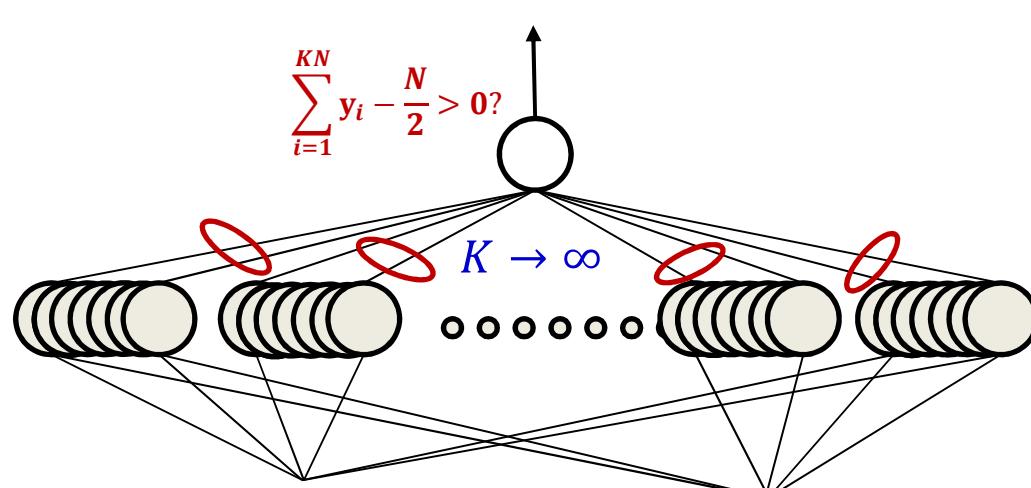


- Deeper networks can require far fewer neurons

Optimal depth in *generic* nets

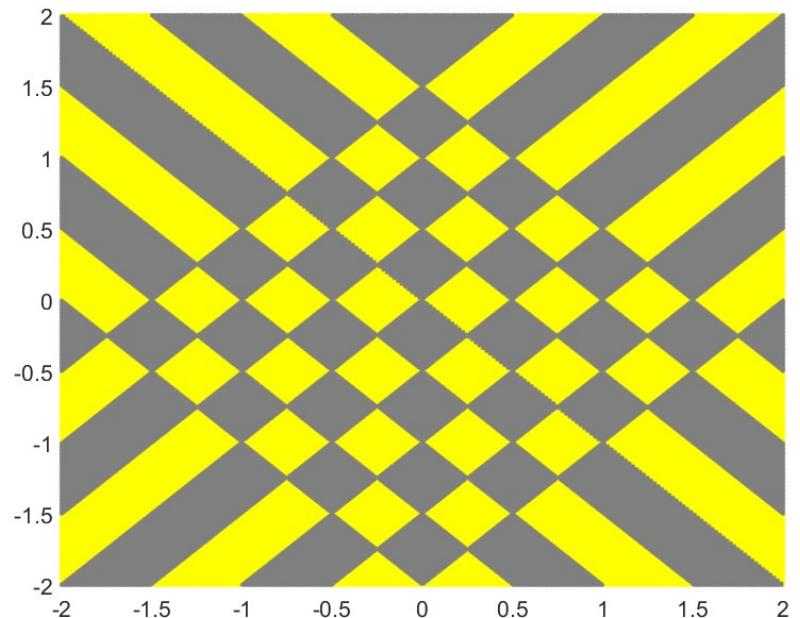
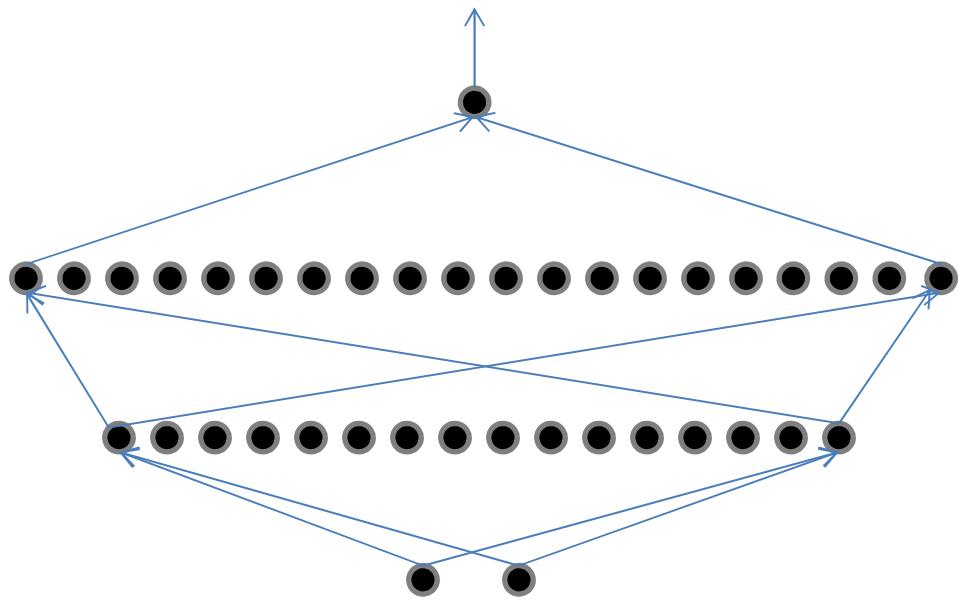
- We look at a different pattern:
 - “worst case” decision boundaries
- For *threshold-activation* networks
 - Generalizes to other nets

Optimal depth



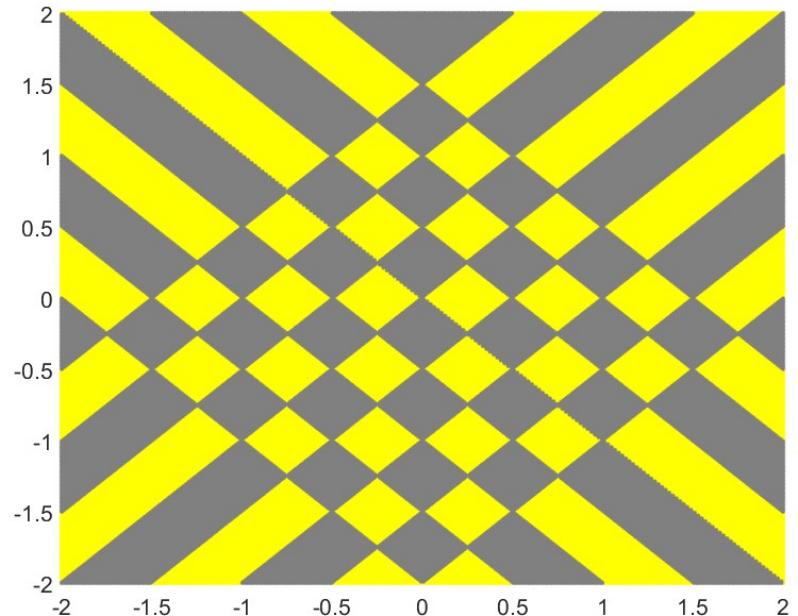
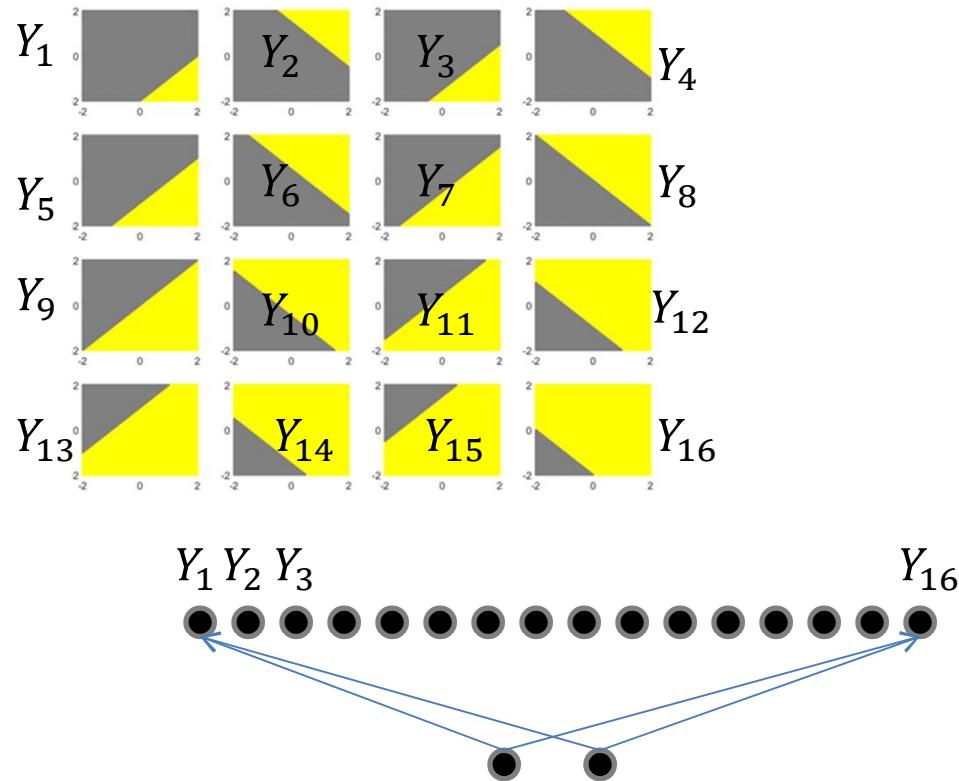
- A naïve one-hidden-layer neural network will require infinite hidden neurons

Optimal depth



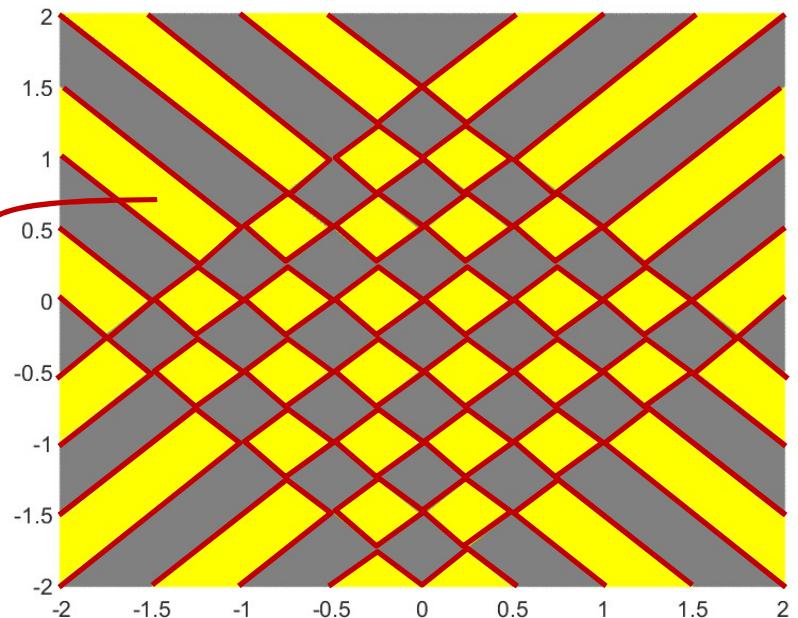
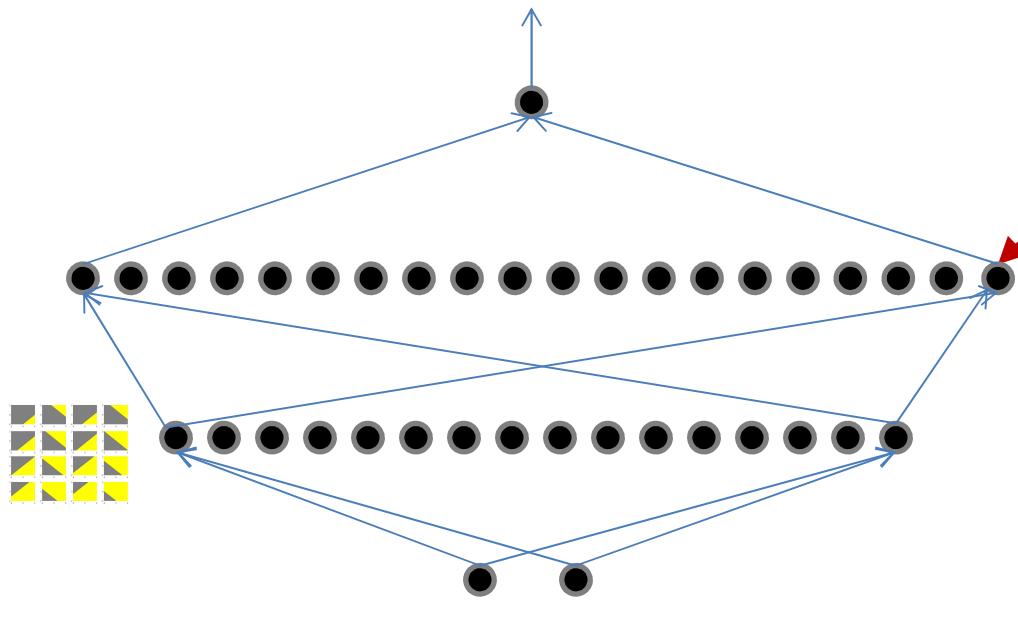
- Two hidden-layer network: 56 hidden neurons

Optimal depth



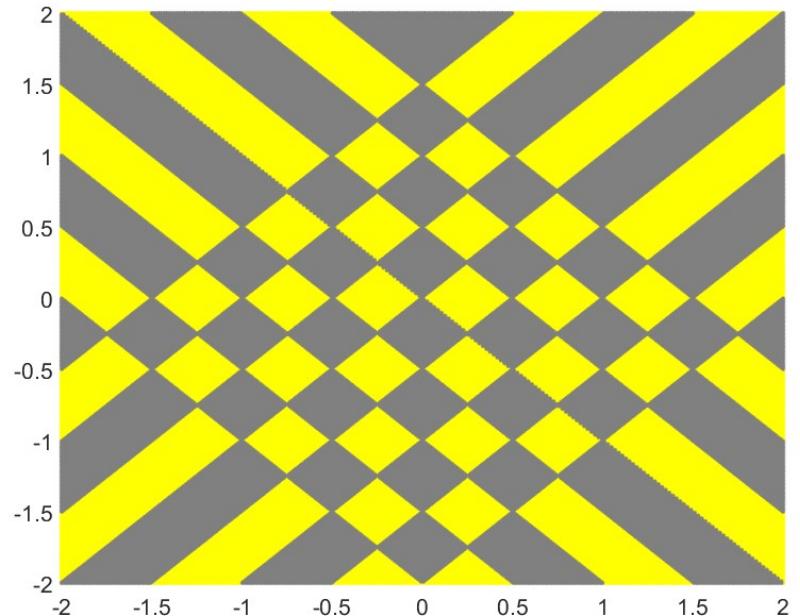
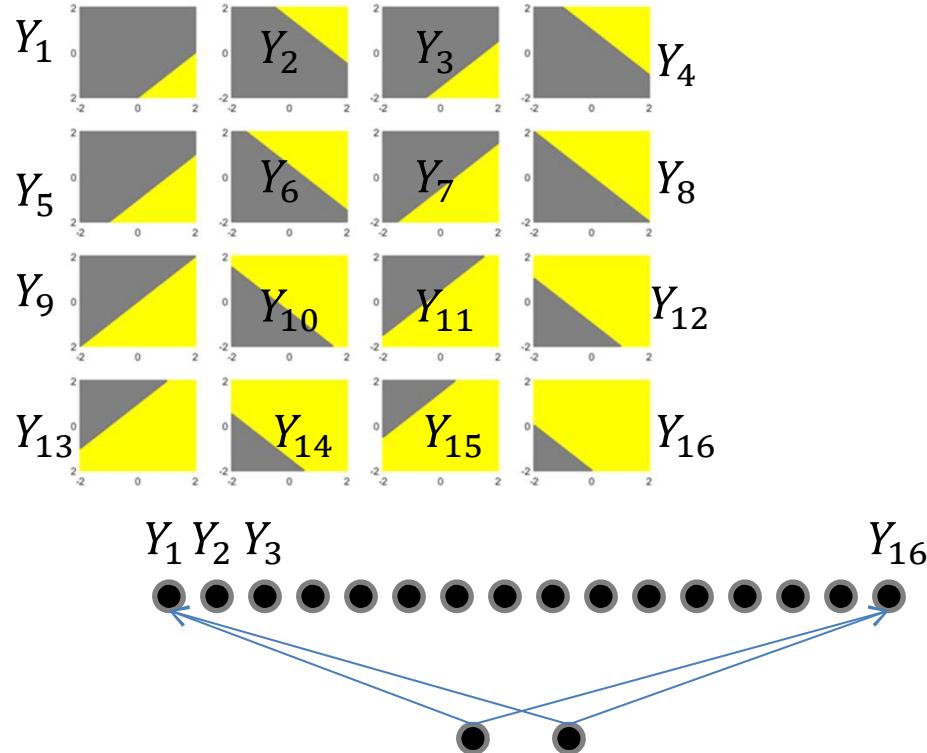
- Two layer network: 56 hidden neurons
 - 16 neurons in hidden layer 1

Optimal depth



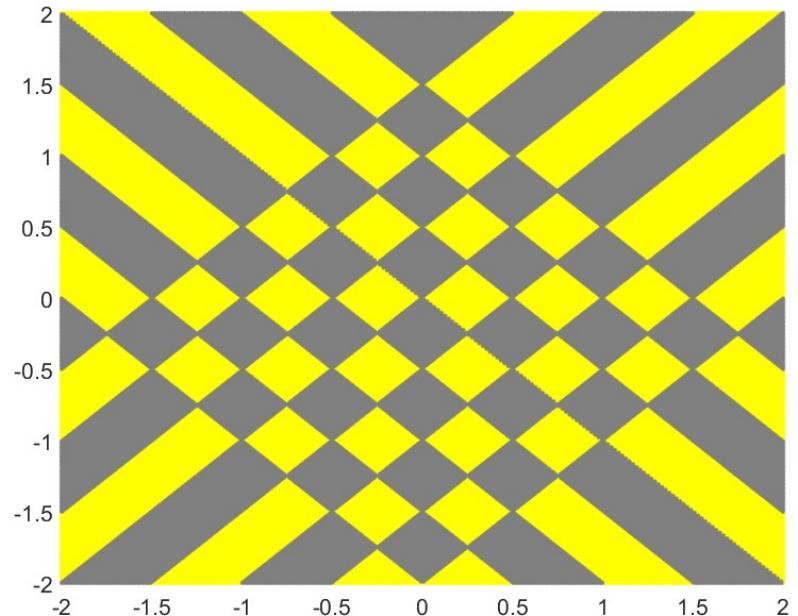
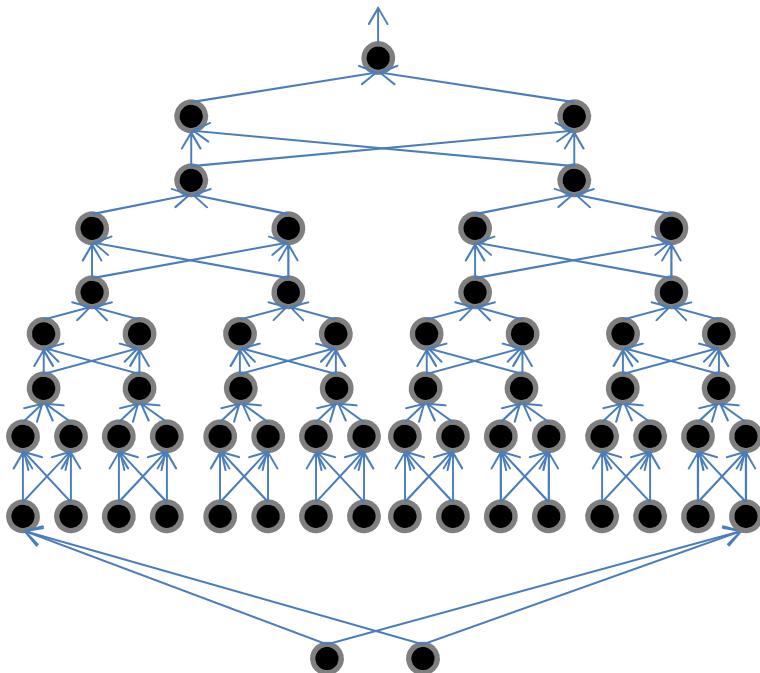
- Two-layer network: 56 hidden neurons
 - 16 in hidden layer 1
 - 40 in hidden layer 2
 - 57 total neurons, including output neuron

Optimal depth



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

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-gate XOR