Why Deep?

CSE 849 Deep Learning

Spring 2025

Zijun Cui

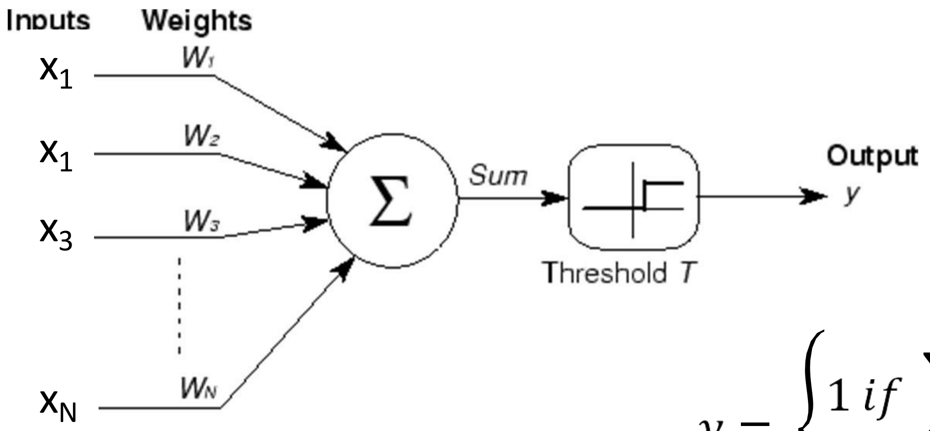
Slides are adapted from CMU 11-785 Introduction to Deep Learning by Bhiksha Raj

Today’s Topic

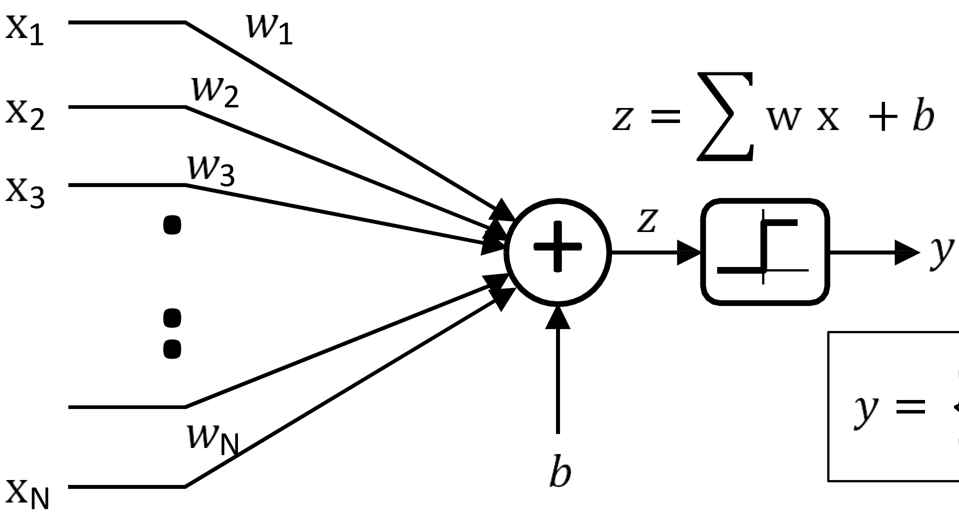
Why we need **DEEP** networks?

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

Recap: the perceptron



$$y = \begin{cases} 1 \; if \; \sum w \, \mathrm{x} \; \geq T \\ 0 \; else \end{cases}$$



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

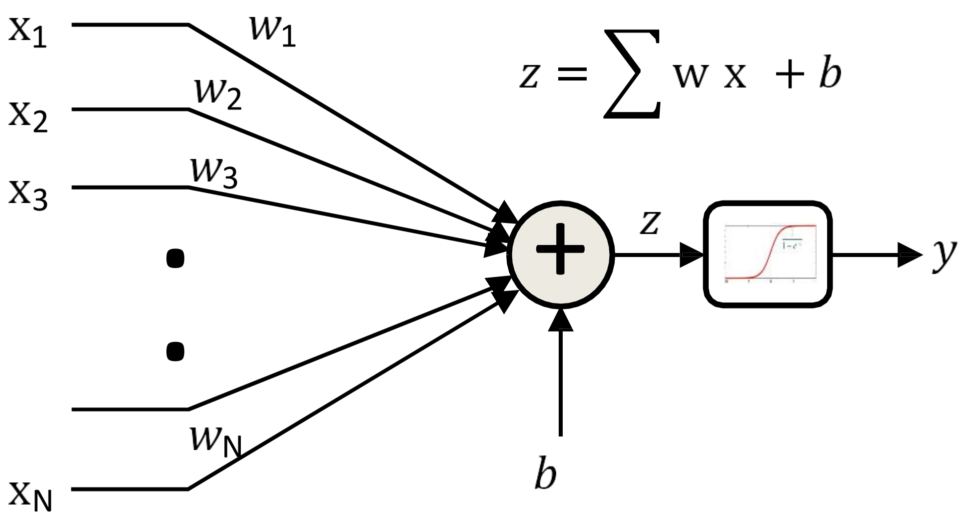
- A threshold unit
  - "Fires" if the weighted sum of inputs exceeds a threshold
  - Electrical engineers will call this a ***threshold gate***
    - A basic unit of Boolean circuits

"Fires" if the affine function of inputs is positive

The bias is the negative of the threshold T

Today's Star

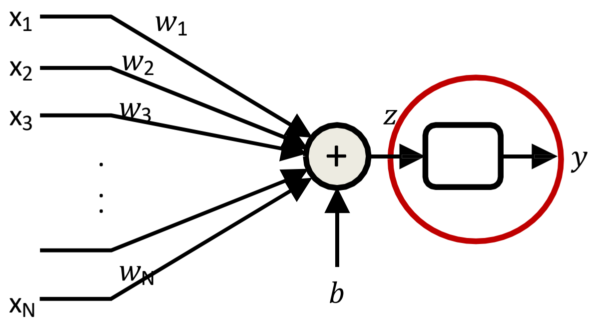
The “soft” perceptron (logistic)



$$z = \sum w\,x \; + b$$

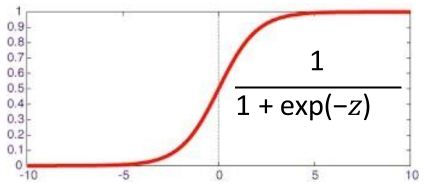
$$\boxed{y = \frac{1}{1 + exp(-z)}}$$

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

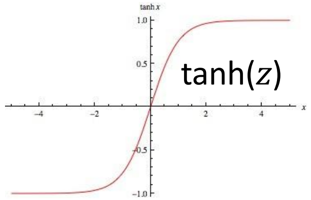


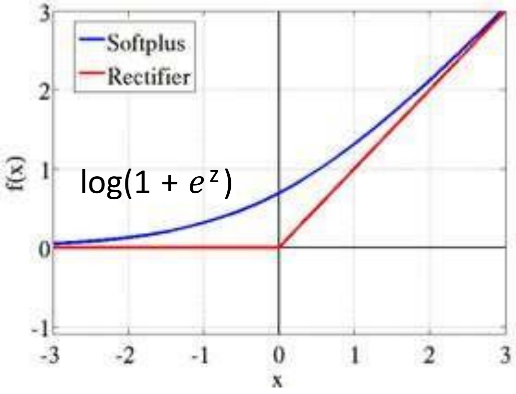
Other “activations”

sigmoid

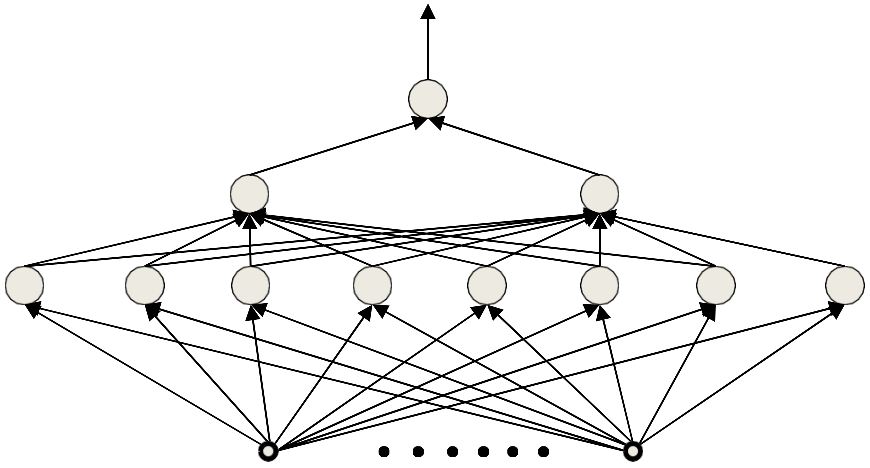


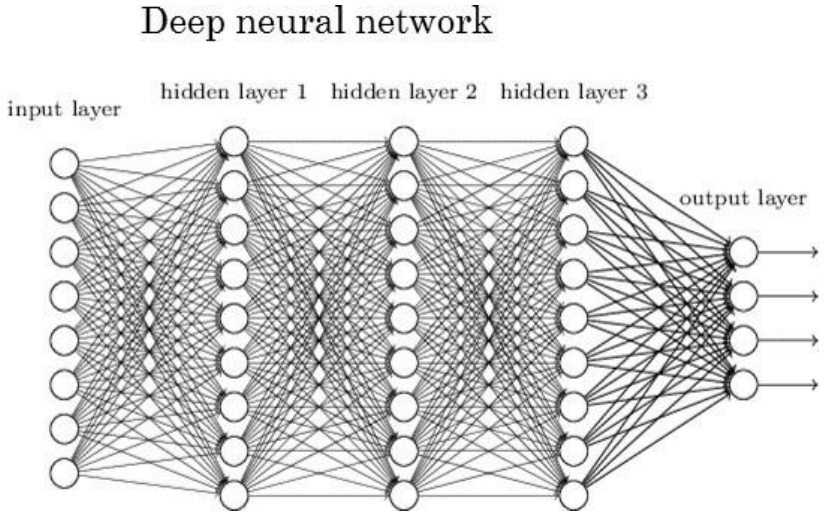
tanh





The multi-layer perceptron



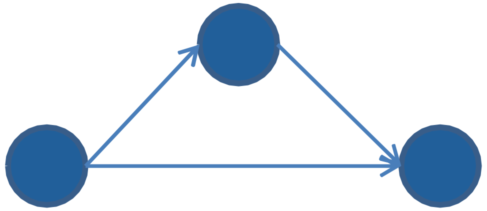


- A network of perceptrons
  - Perceptrons “feed” other perceptron
  - “formal” definition of a layer?

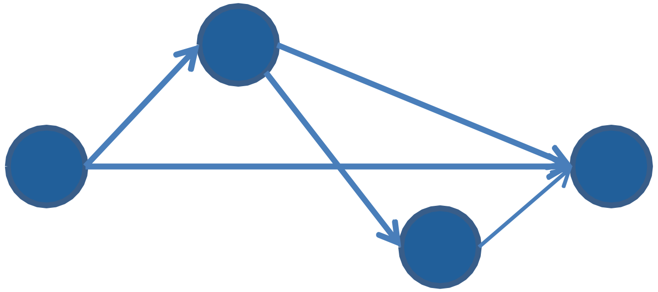
- Defining “depth”
- What is a “deep” network

Deep Structures

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

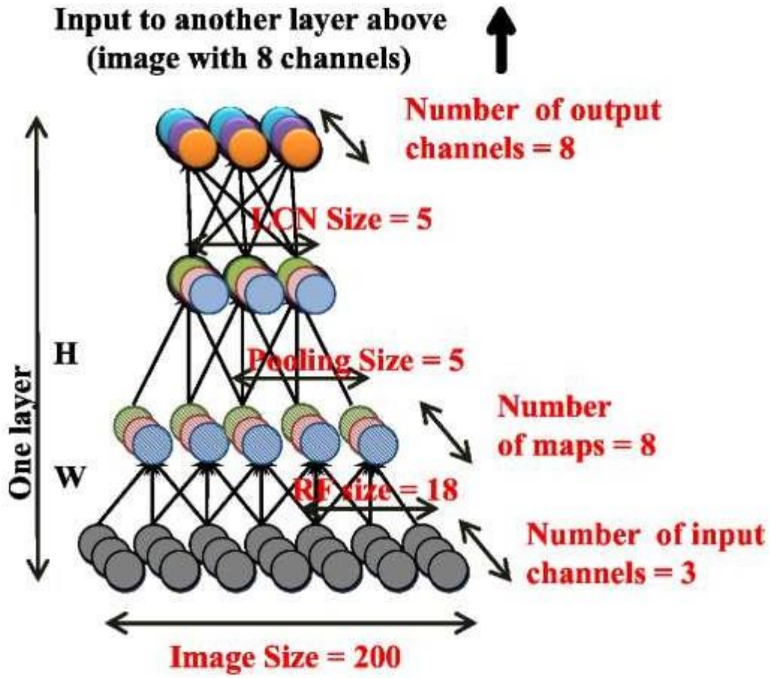


Depth = 2



Depth = 3

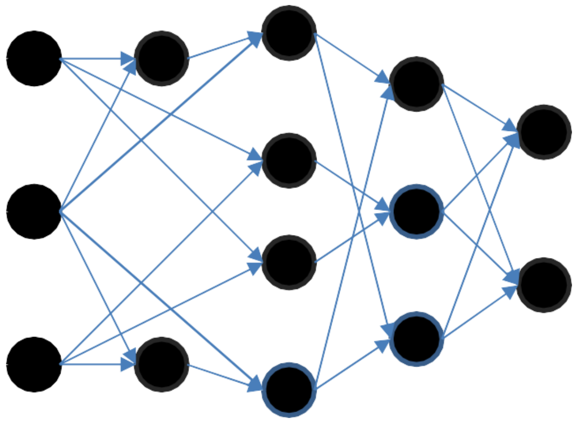
- *Deep structure*
  - *The input is the “source”,*
  - *The output nodes are “sinks”*

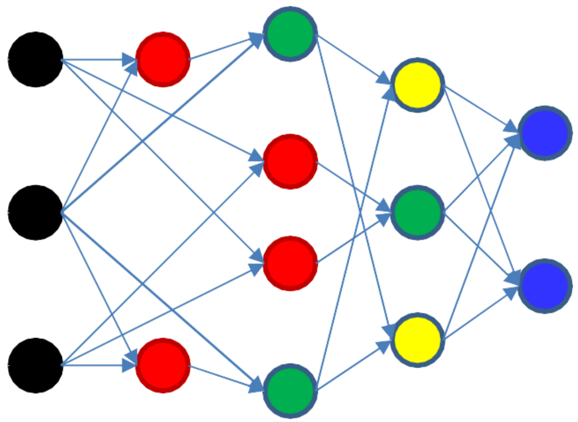


“Deep”: Depth of output neurons is greater than 2

What is a layer?

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

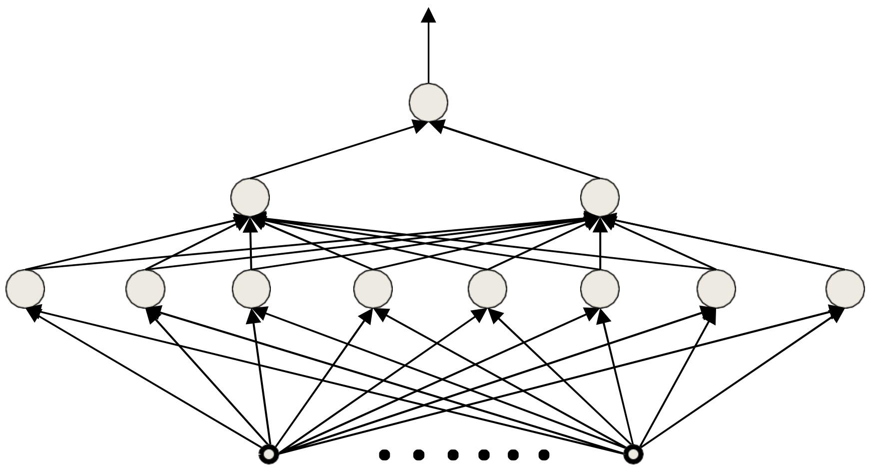


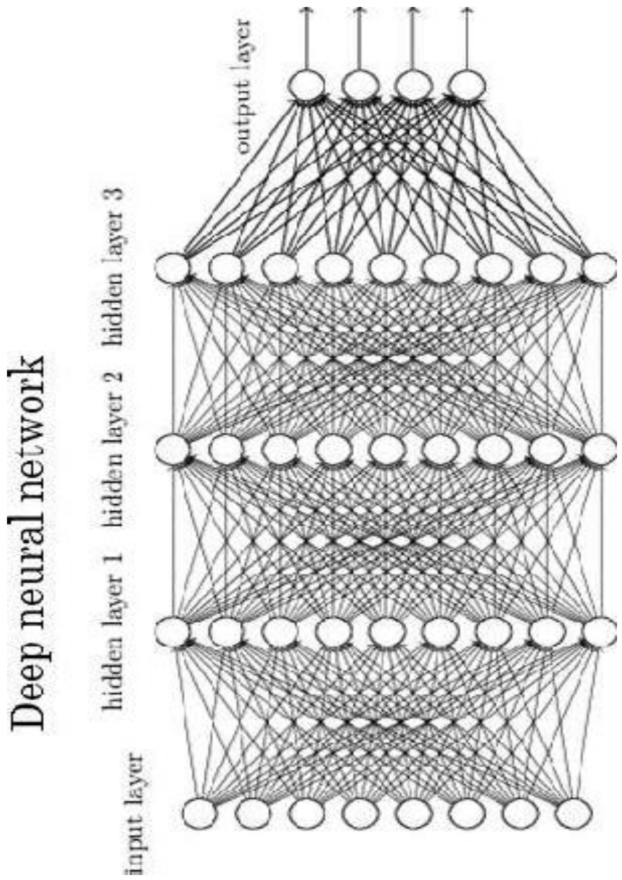


Input: Black
Layer 1: Red
Layer 2: Green
Layer 3: Yellow
Layer 4: Blue

“Deep” : At least 3 layers
– Output layer depth is at least 3

The multi-layer perceptron



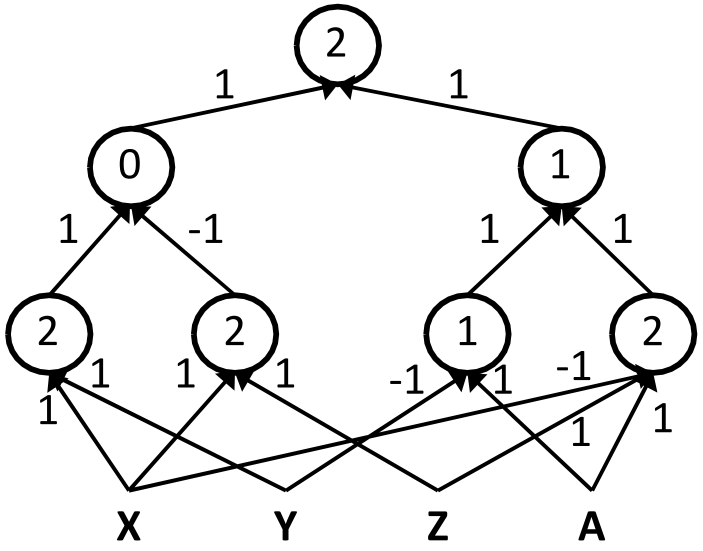


- Inputs are real or Boolean values
- Outputs are real or Boolean values
  – Can have multiple outputs for a single input
- What can this network compute?
  - MLPs can compose Boolean functions and real-valued functions
- What are the limitations?

--- the need for depth

MLP as Boolean Functions

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



- MLPs are universal Boolean functions
  - Any function over any number of inputs and any number of outputs

- **But how many "layers" will they need?**

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

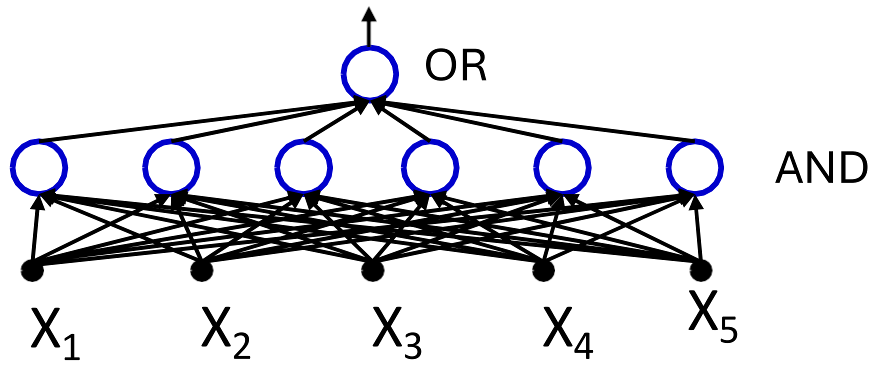
How many layers for a Boolean MLP?

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

Truth Table

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

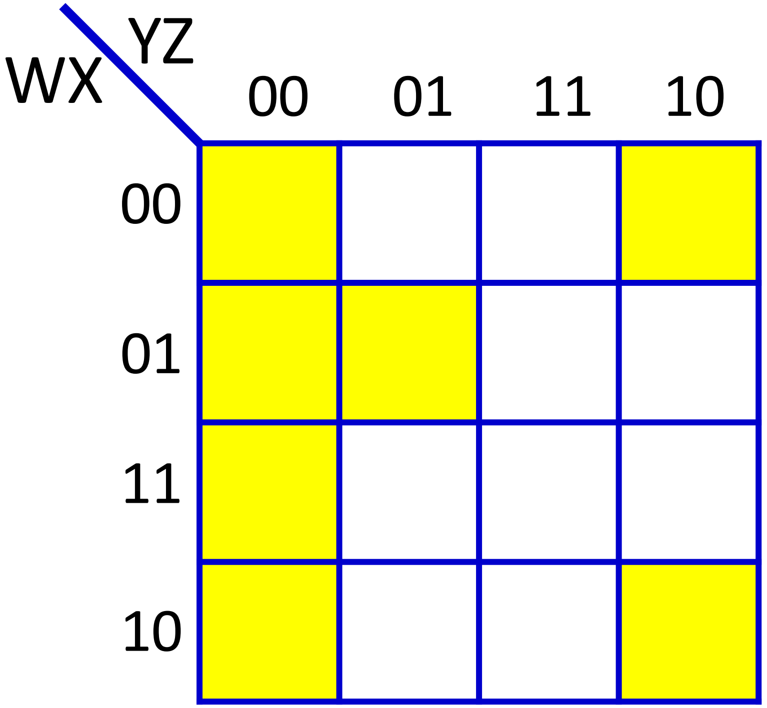
$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



- Expressed in disjunctive normal form (DNF)
- 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 perceptron required in the single hidden layer for an N-input-variable function?

Reducing a Boolean Function



This is a "Karnaugh Map"

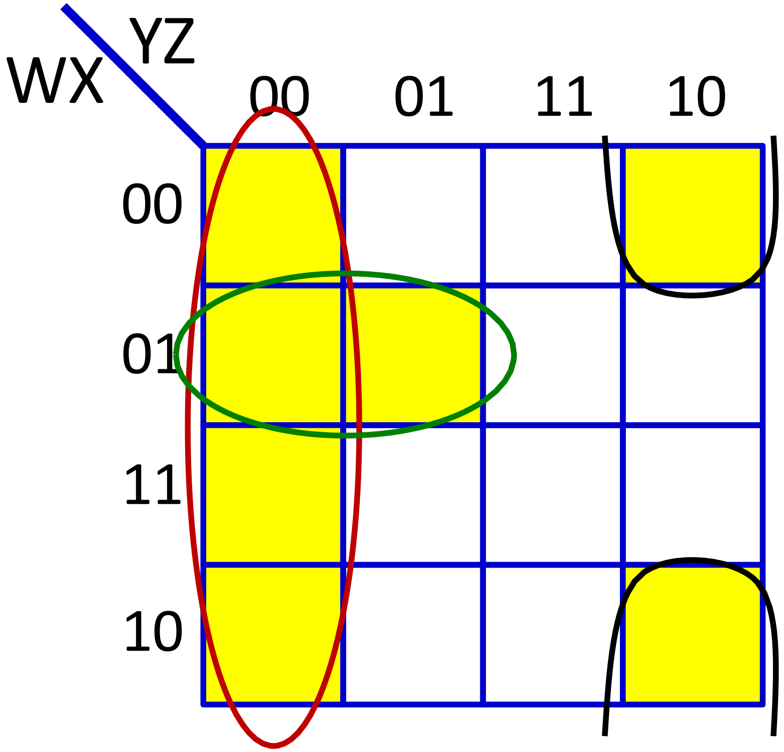
It represents a truth table as a grid Filled boxes represent input combinations for which output is 1; blank boxes have output 0

- Disjunctive Normal Form (DNF) form:
  - Find groups
  - Express as reduced DNF

Today's Star

Basic disjunctive normal form (DNF) formula will require 7 terms

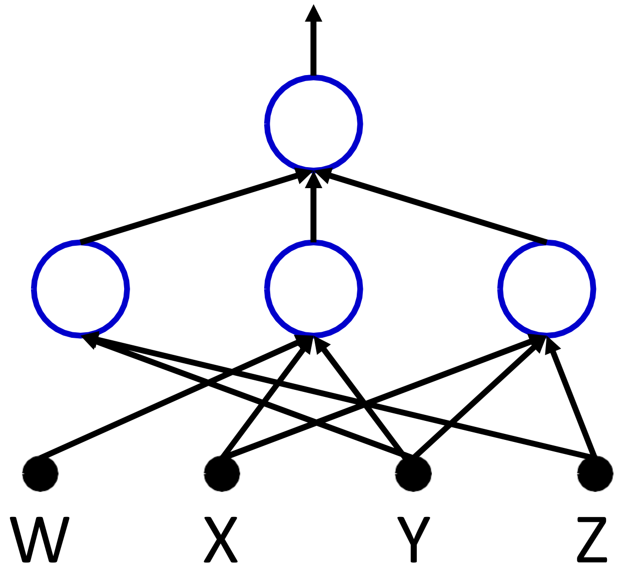
Reducing a Boolean Function



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

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

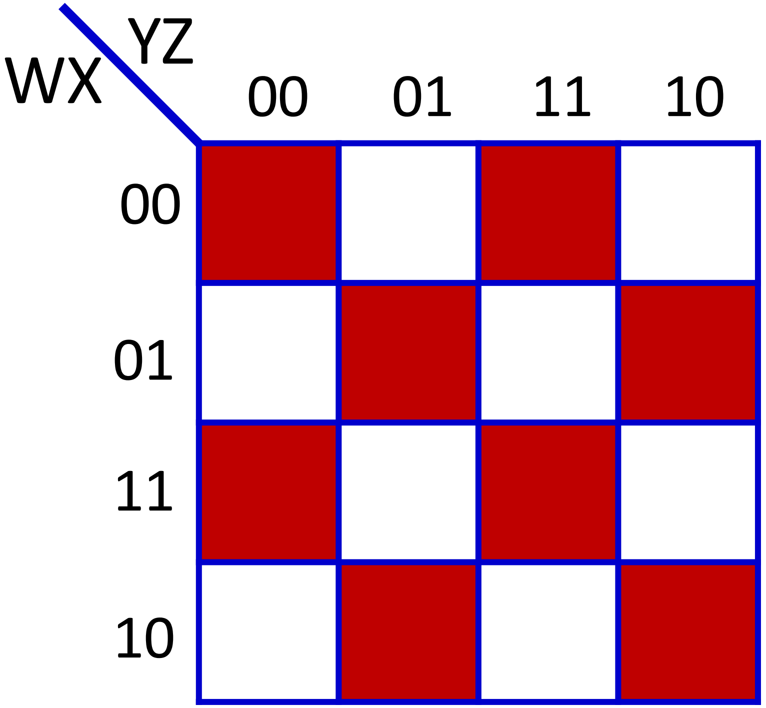
$$O = \bar{Y}\bar{Z} + \bar{W}X\bar{Y} + \bar{X}Y\bar{Z}$$



- 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?



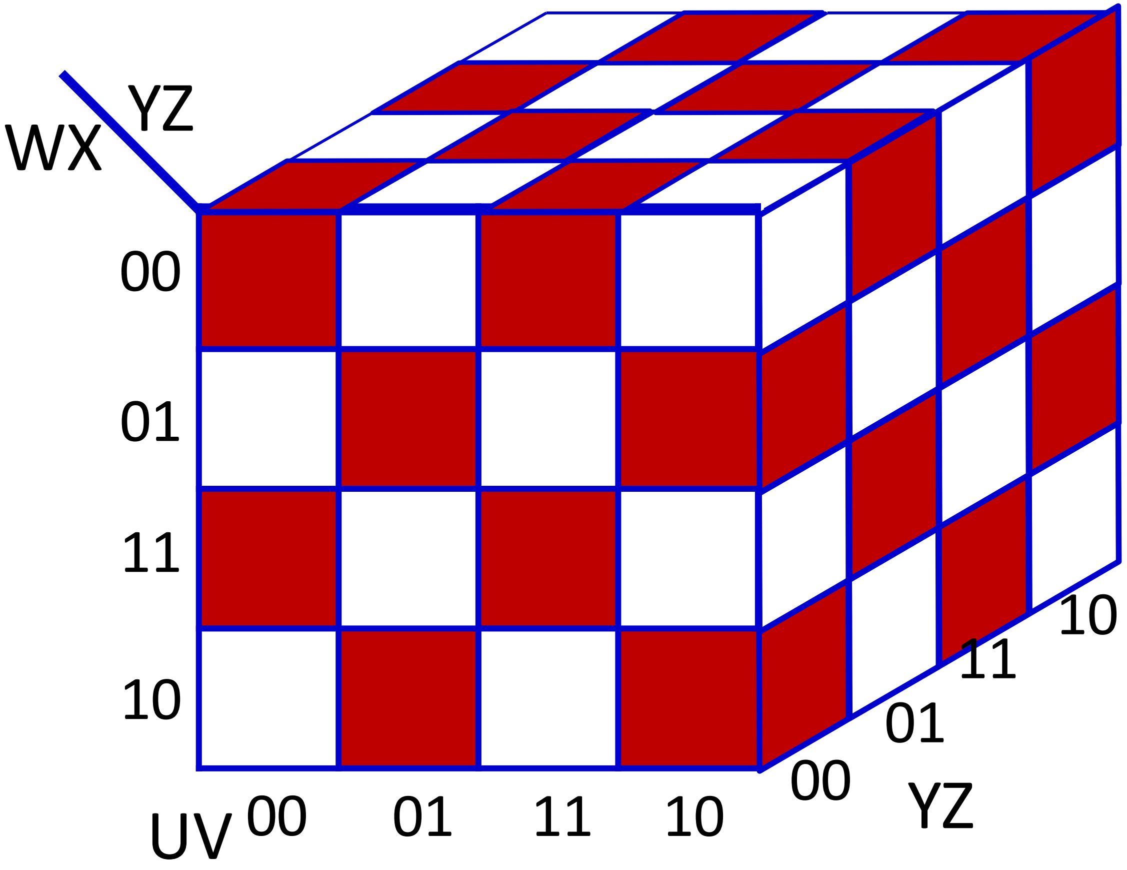
Worst Case

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

Answer： 8

Largest irreducible DNF?

for Boolean function of 6 variables?



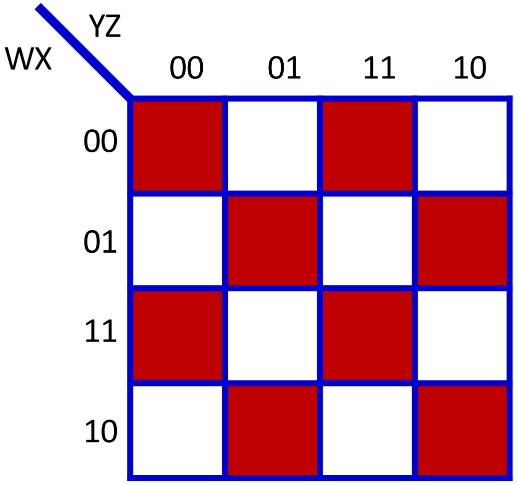
Answer: 32

Width of a one-hidden-layer Boolean MLP

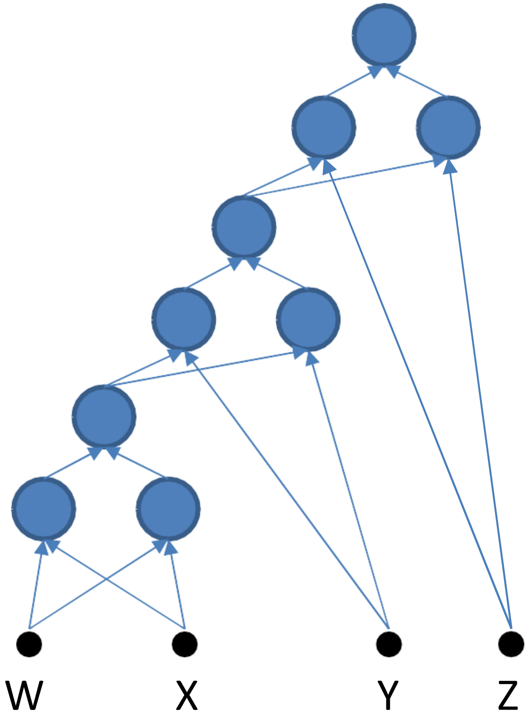
- Can be generalized.
  For N variables, Will require 2^{N-1} perceptrons in hidden layer
  Exponential in N – not good

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

Size of a deep MLP

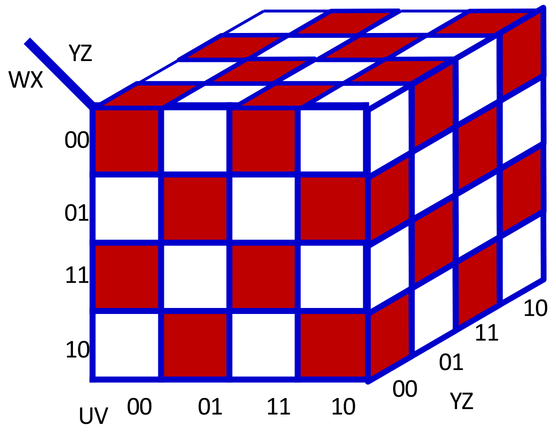


$$O = W \oplus X \oplus Y \oplus Z$$

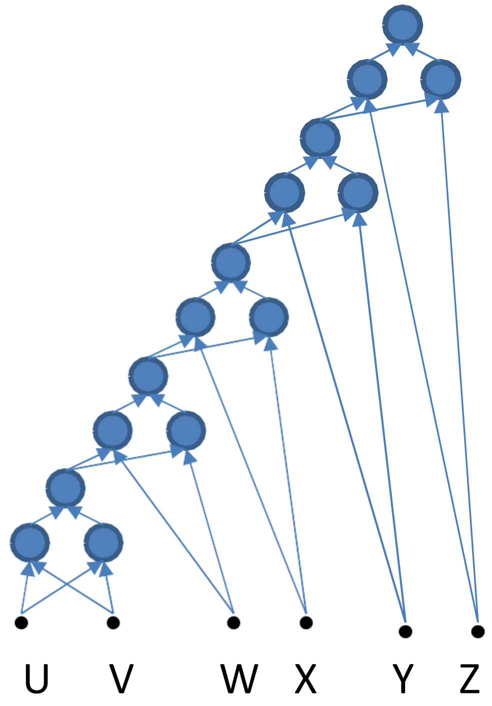


- An XOR needs 3 perceptrons
- This network will require 3x3 = **9 perceptrons**

Size of a deep MLP



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



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

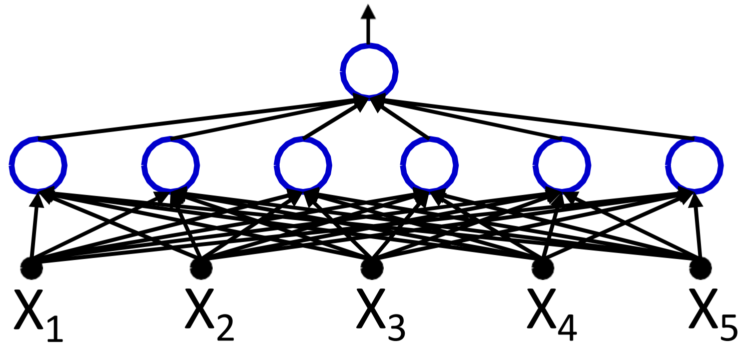
More generally, the XOR of N variables will require **3(N-1) perceptrons**

Width of a one-hidden-layer Boolean MLP

- Can be generalized.
  For N variables, Will require 2^{N-1} perceptrons in hidden layer
  Exponential in N **– not good**

- How many units if we use *multiple hidden layers?*
  Will require 3(N-1) perceptrons in a deep network
  Linear in N

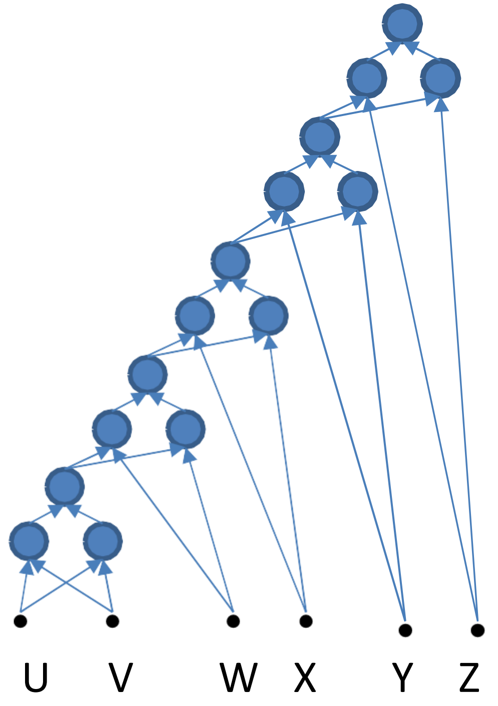
The actual number of parameters in a network



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

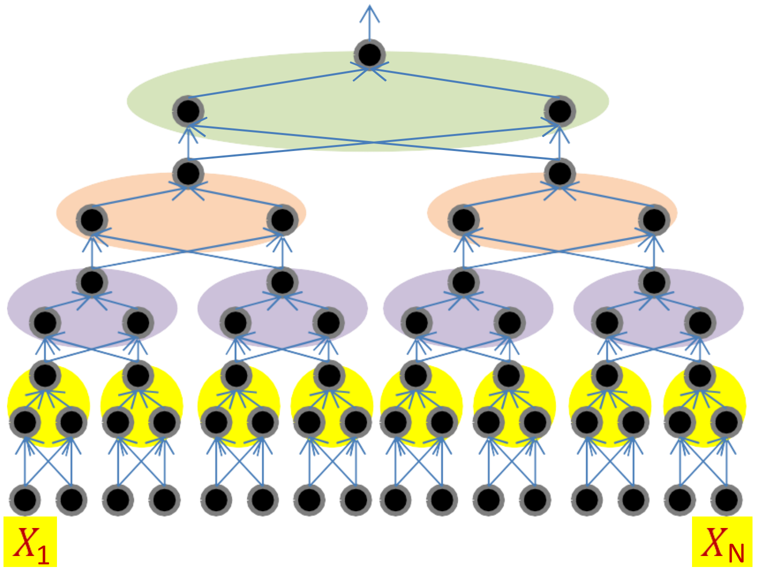
- This is the number that really matters in software or hardware implementations
- Networks that require an exponential number of neurons will require an exponential number of weights..

A More Efficient Representation – reducing depth



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

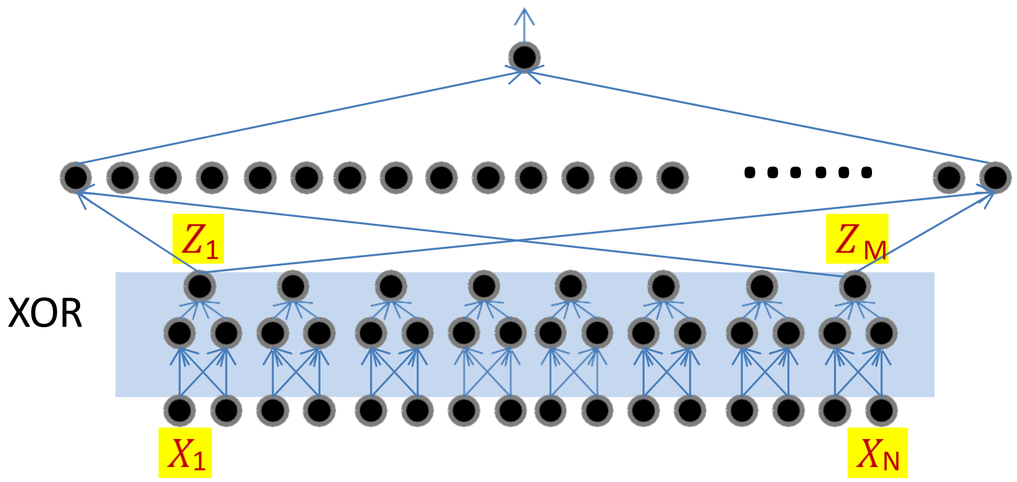
- Require 3(N-1) perceptrons
- Depth = N-1
- Brute-Force Linear XOR Network



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

- Only $2\log_2 N$ layers
  - By pairing terms through Binary Tree
  - Total layers needed to reduce N variables to 1 is $\boldsymbol{log_2}$**N**
  - Each XOR operation still requires **2 layers**

Tradeoff between Depth and Width



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

$$= Z_1 \oplus Z_2 \oplus \cdots \oplus Z_M$$

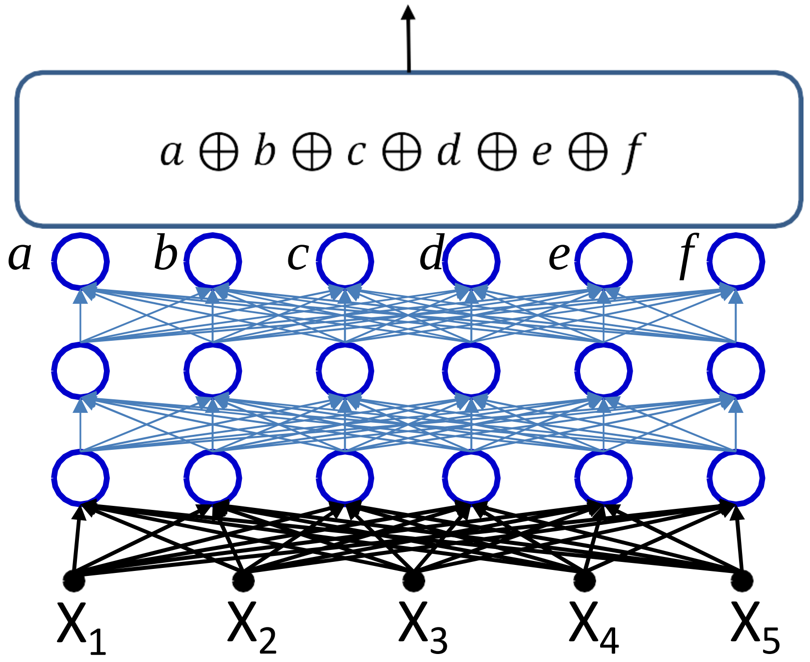
After K-1 layers, the size of intermediate layer M becomes:

$$M = N \cdot 2^{-\frac{K-1}{2}} = CN$$

- Using only K hidden layers, NN will require O(2^{CN}) neurons in the Kth layer, where $C = 2^{-\frac{K-1}{2}}$
  - Because the output is the XOR of all the values output by the K-1th hidden layer, i.e., $M = N \cdot 2^{-\frac{K-1}{2}}$
- Reducing the number of layers below the minimum will result in an exponentially sized network to express the function fully
- A network with fewer than the minimum required number of neurons *cannot* model the function

The need for depth

The XORs could occur anywhere!

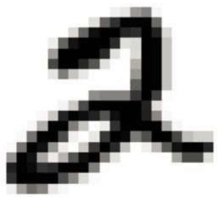


- The XOR structure could occur in any layer
- Having a few extra layers can greatly reduce network size

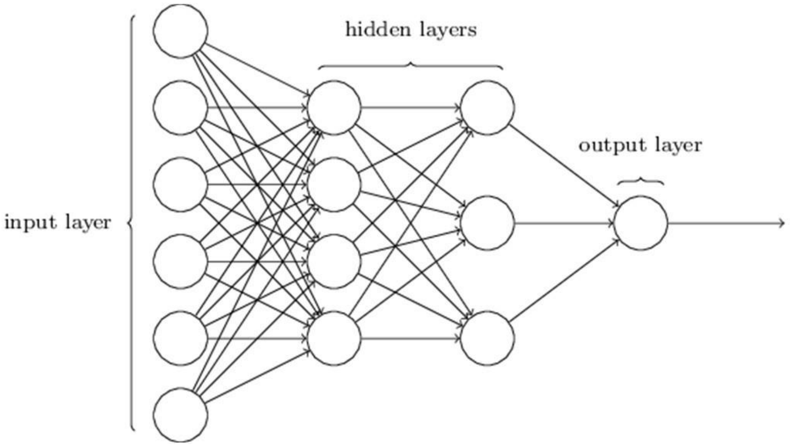
Network size: summary

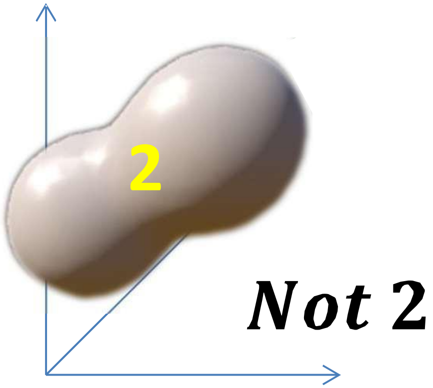
- An MLP is a universal Boolean function
- But can represent a given function only if
  - It is sufficiently wide
  - It is sufficiently deep
  - Depth can be traded off for (sometimes) exponential growth of the width of the network
- Optimal width and depth depend on the number of variables and the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

The MLP as a classifier



784 dimensions
(MNIST)

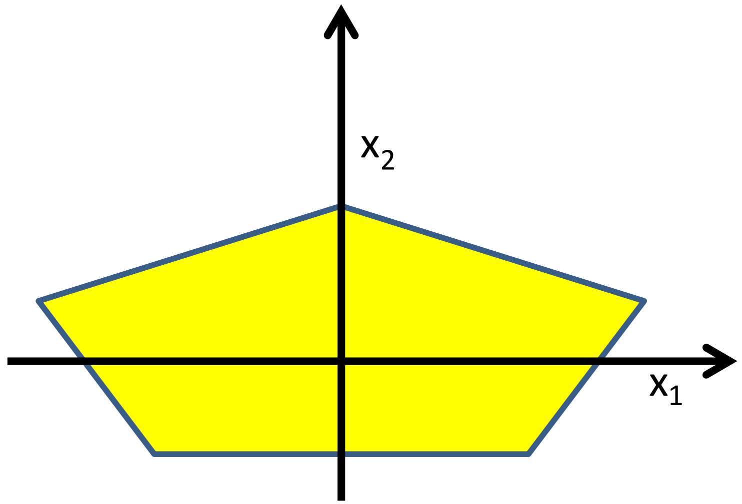




784 dimensions

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

Composing complicated "decision" boundaries

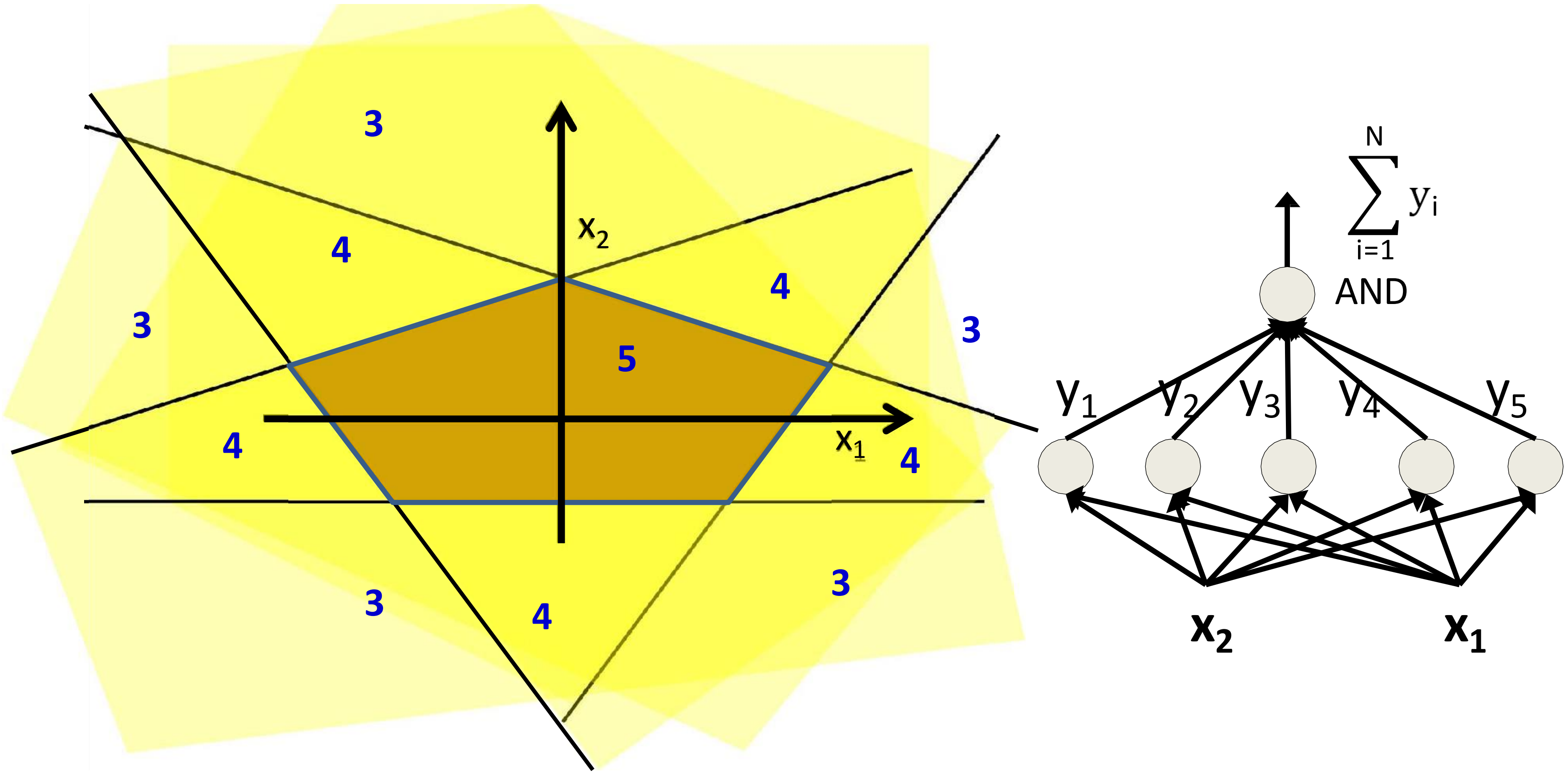


Can now be composed into "networks" to compute arbitrary classification "boundaries"

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

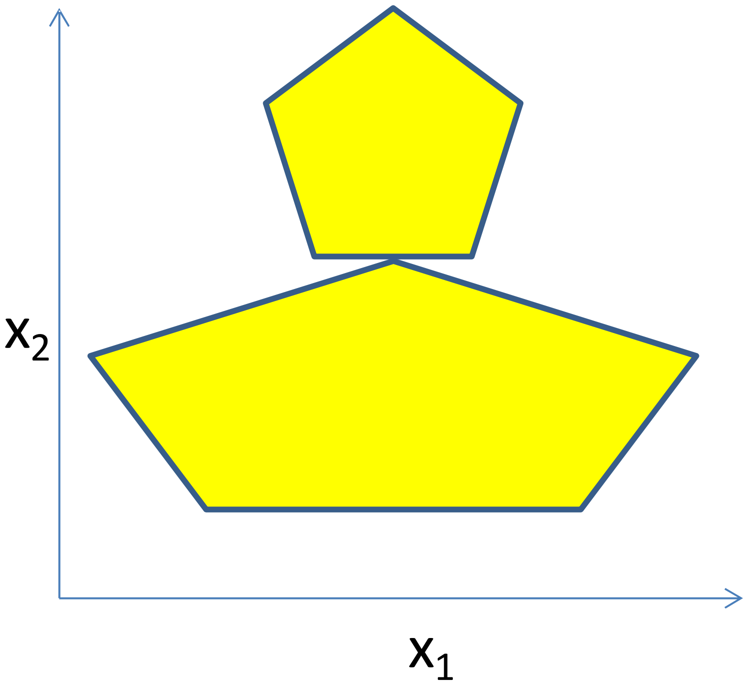
Today's Star

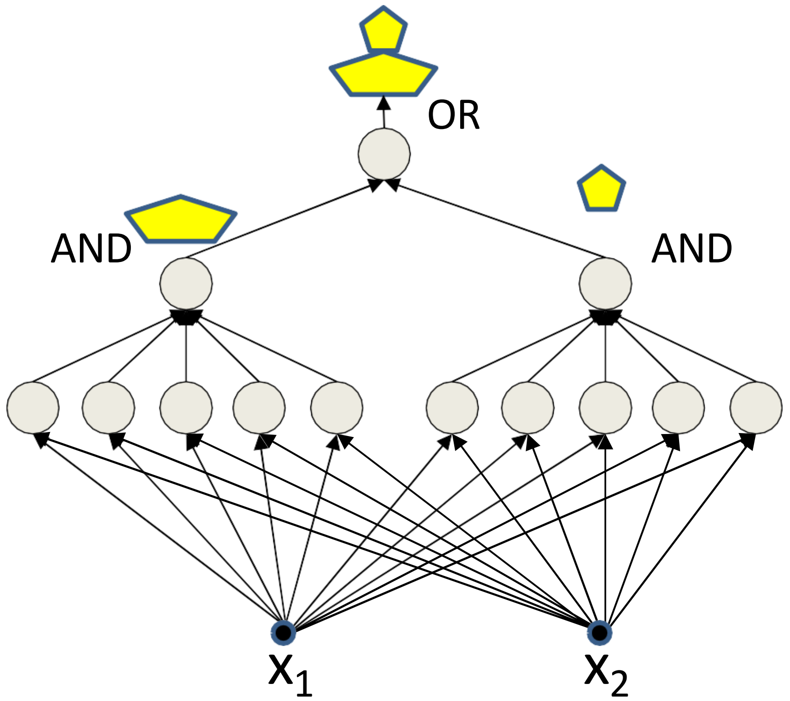
Booleans over the reals



- The network must fire if the input is in the coloured area
  – The AND compares the sum of the hidden outputs to 5

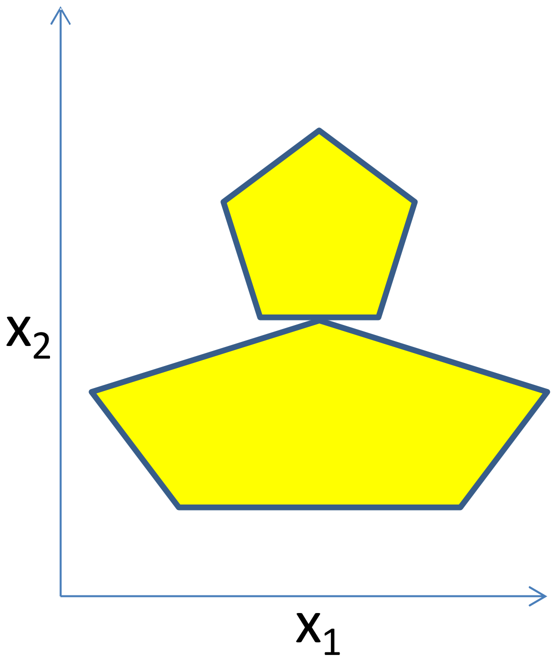
More complex decision boundaries

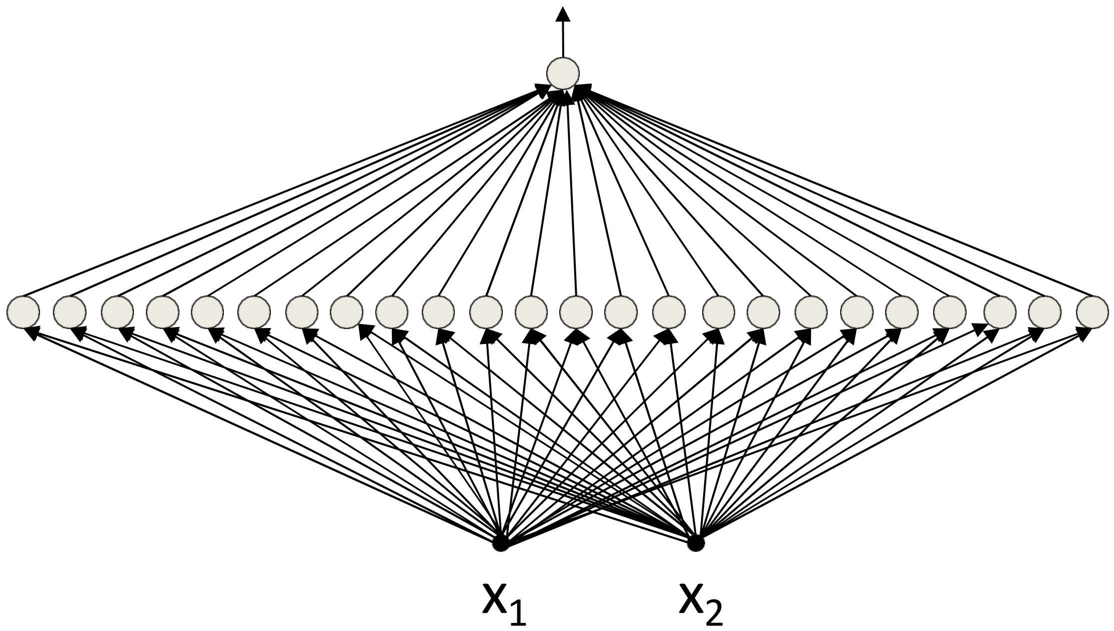




- Network to fire if the input is in the yellow area
  - "OR" two polygons
  - A third layer is required

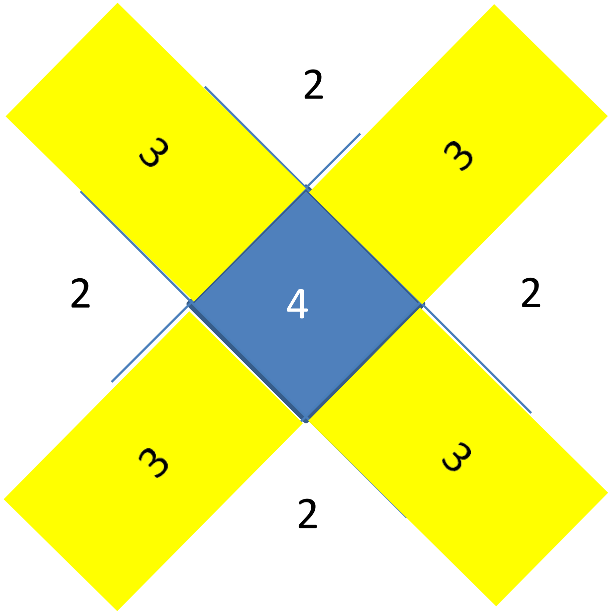
Question: compose this with one hidden layer

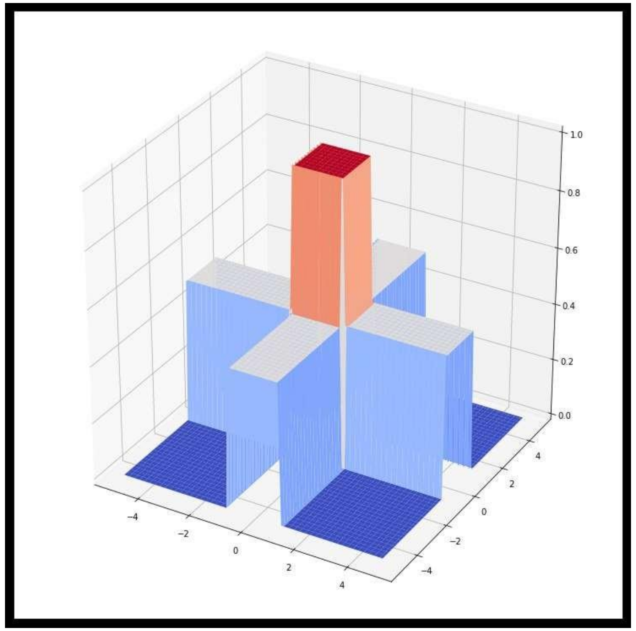


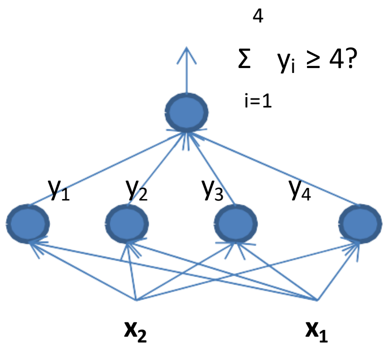


- How would you compose the decision boundary to the left with only *one* hidden layer?

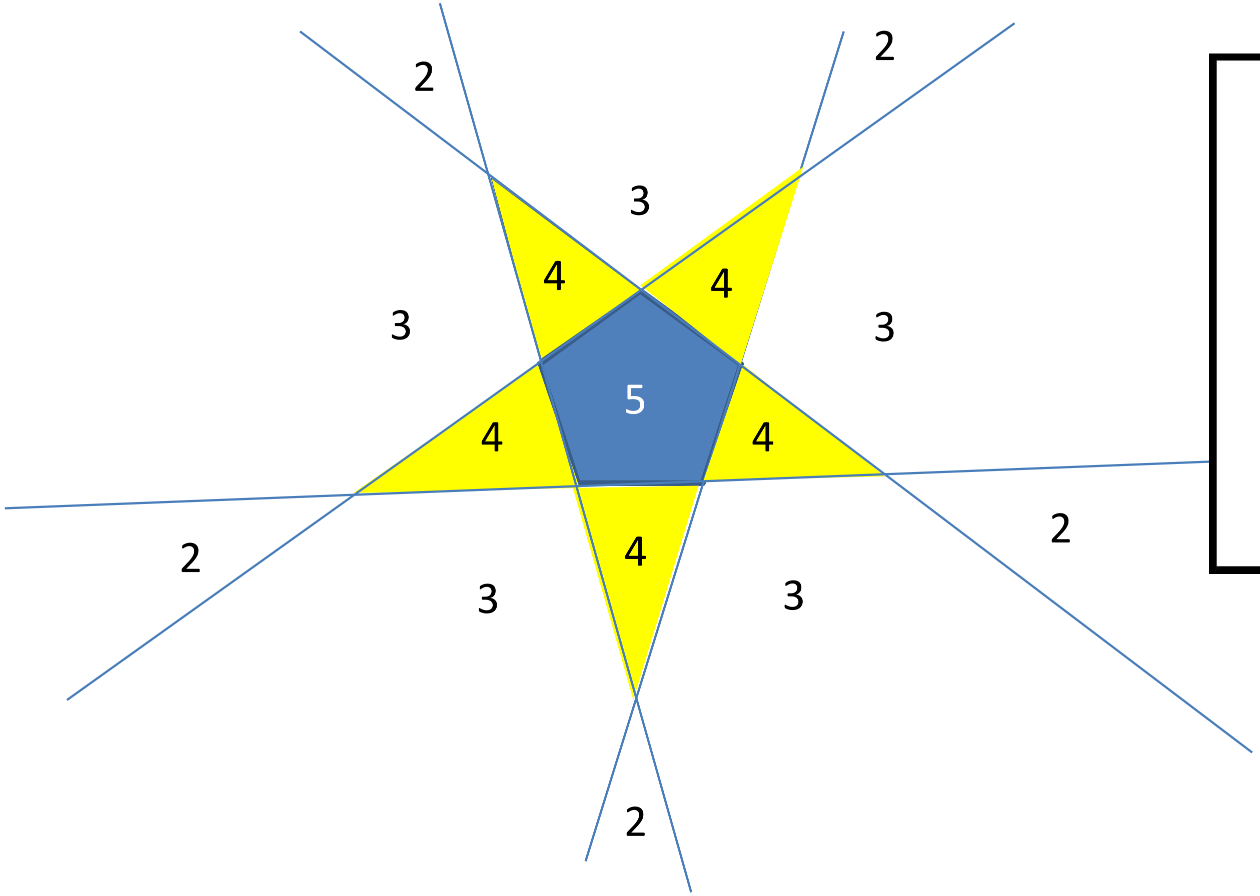
Composing a Square decision boundary

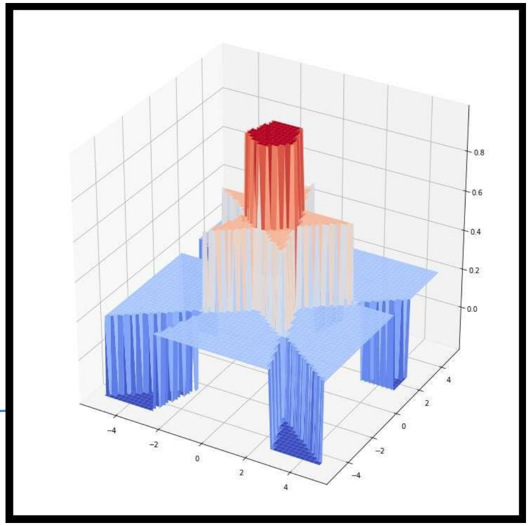


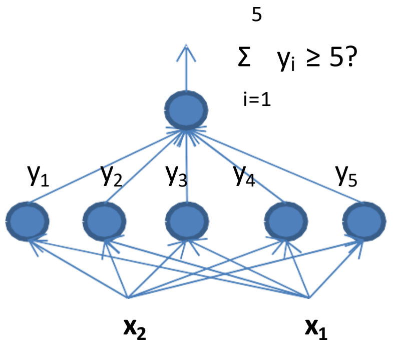




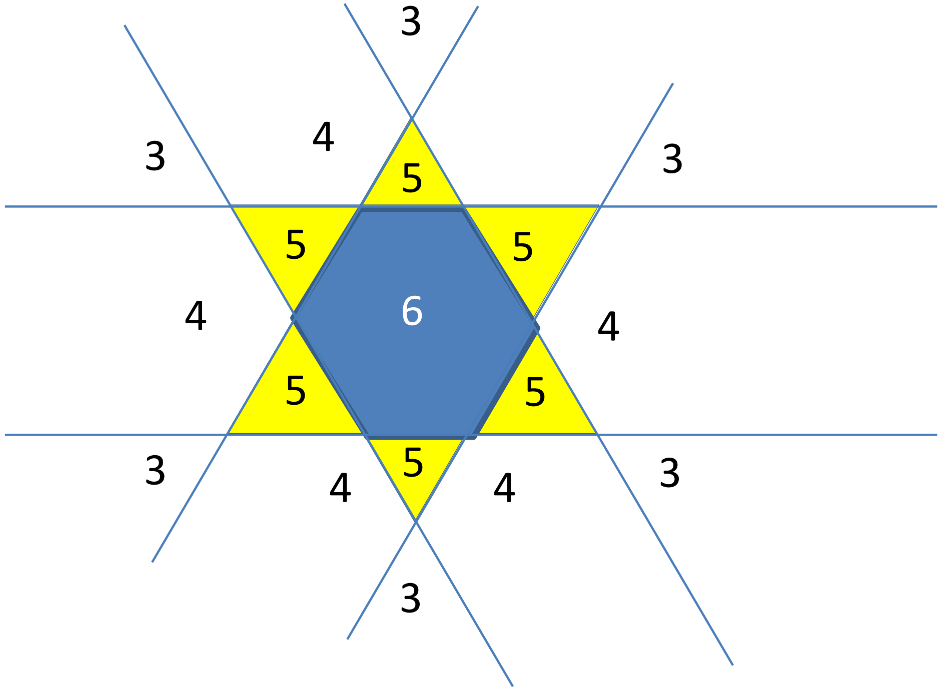
Composing a pentagon

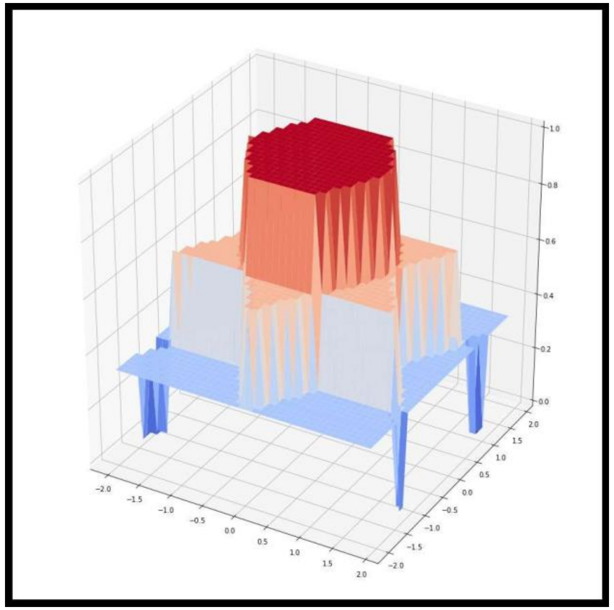




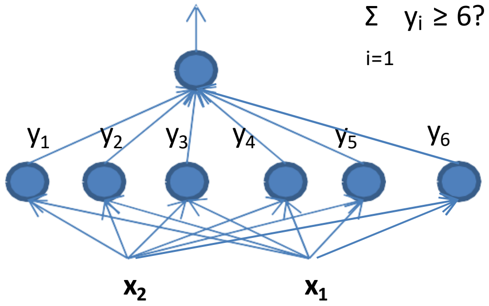


Composing a hexagon

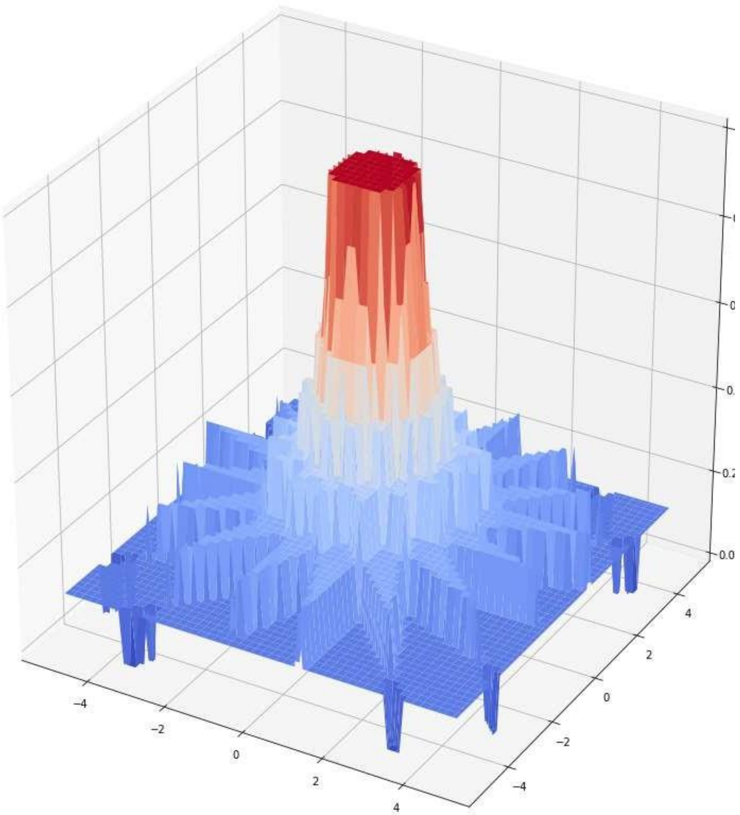




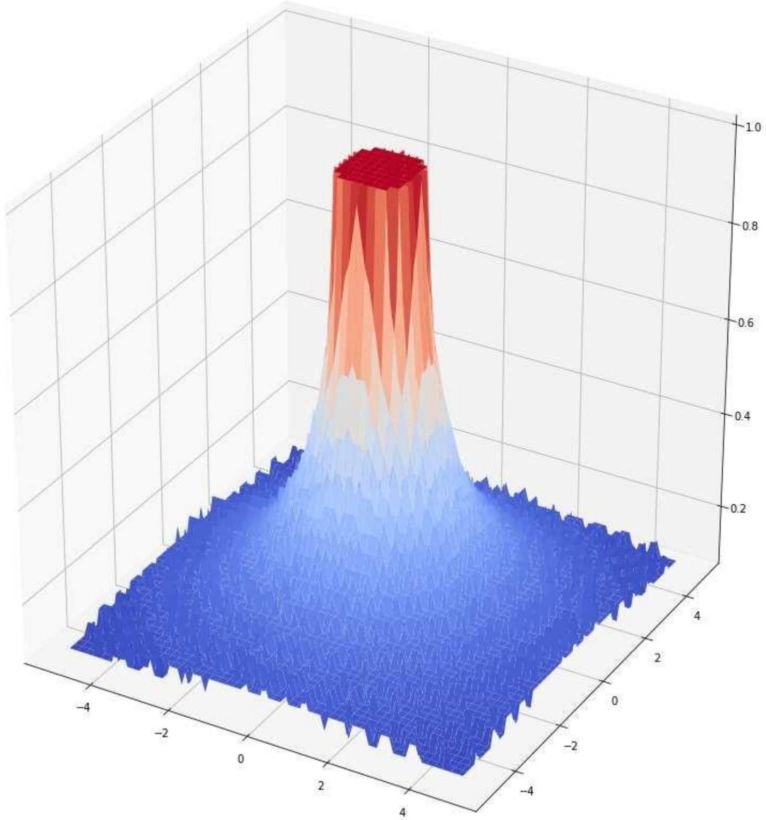
$$\sum_{i=1}^{N} y_i \geq 6?$$



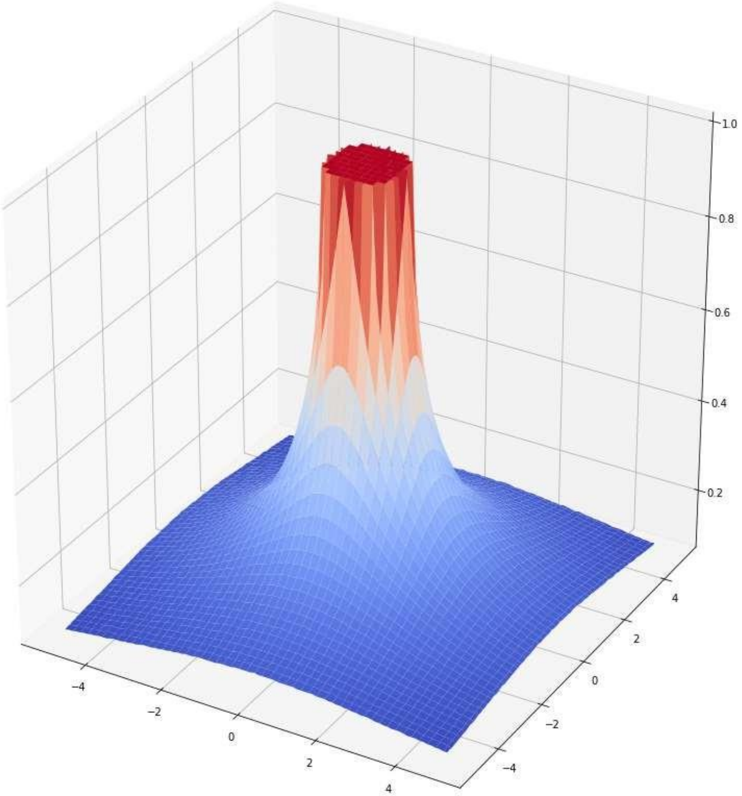
16 sides



64 sides



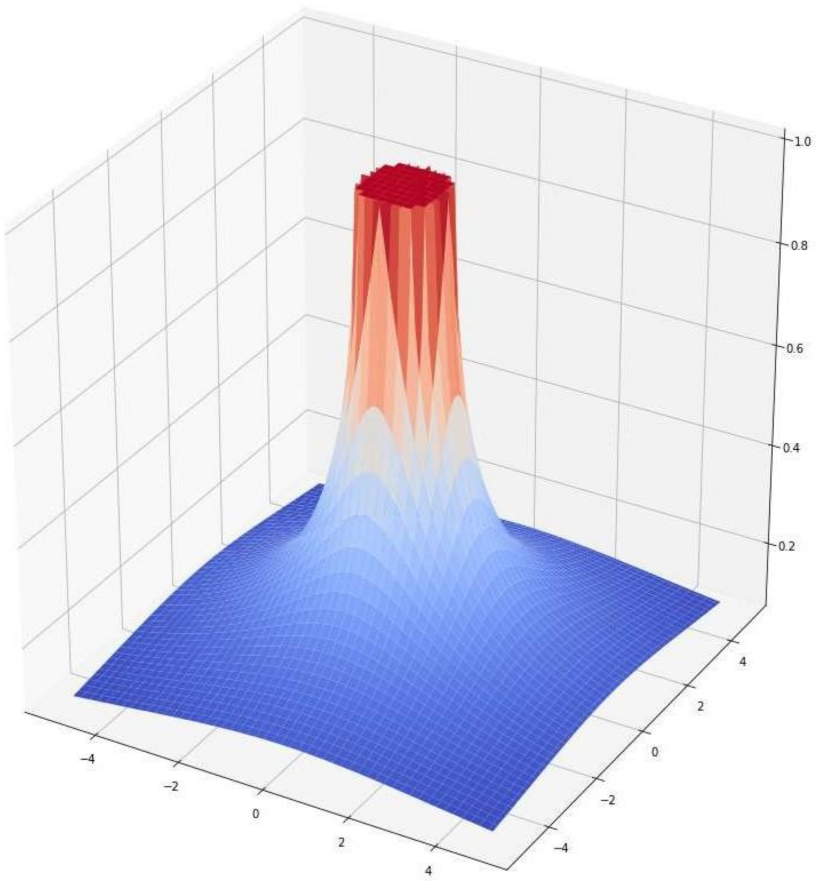
1000 sides

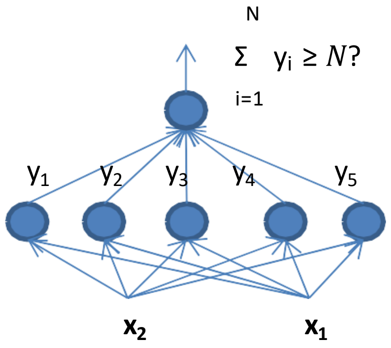


- What are the sums in the different regions?
  - A pattern emerges as we consider N > 6..

  N is the number of sides of the polygon

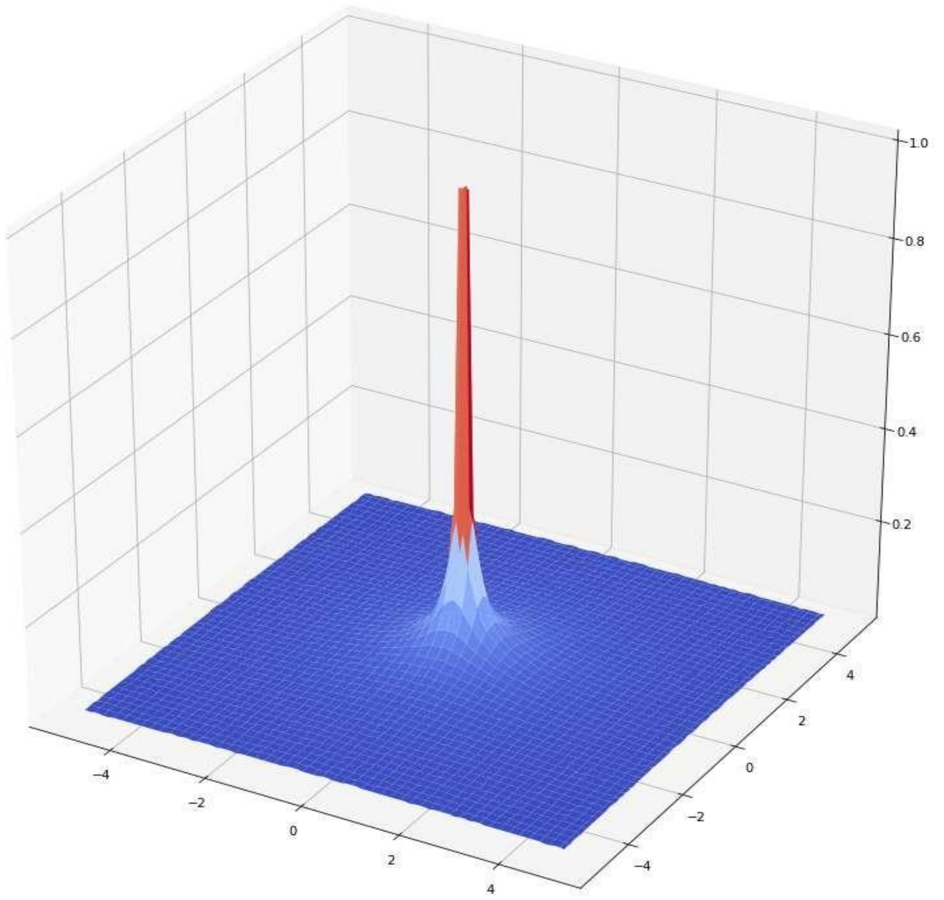
Polygon net





- Increasing the number of sides reduces the area outside the polygon

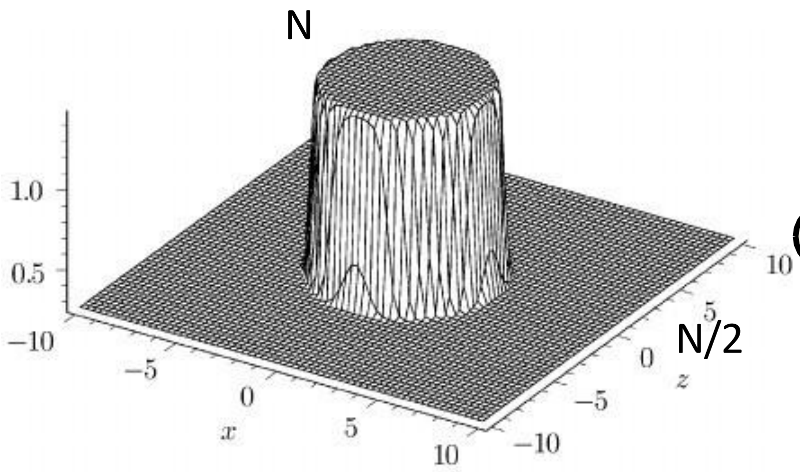
In the limit $N \to \infty$

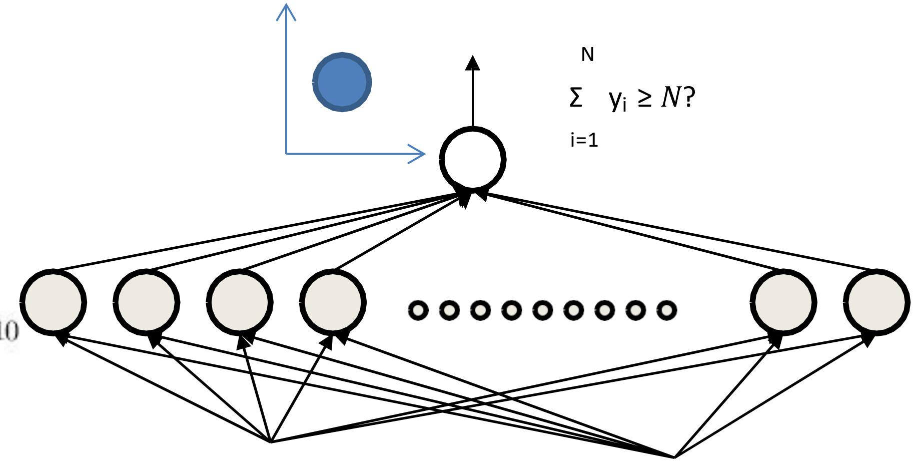


It is a near perfect cylinder

– N in the cylinder, N/2 outside

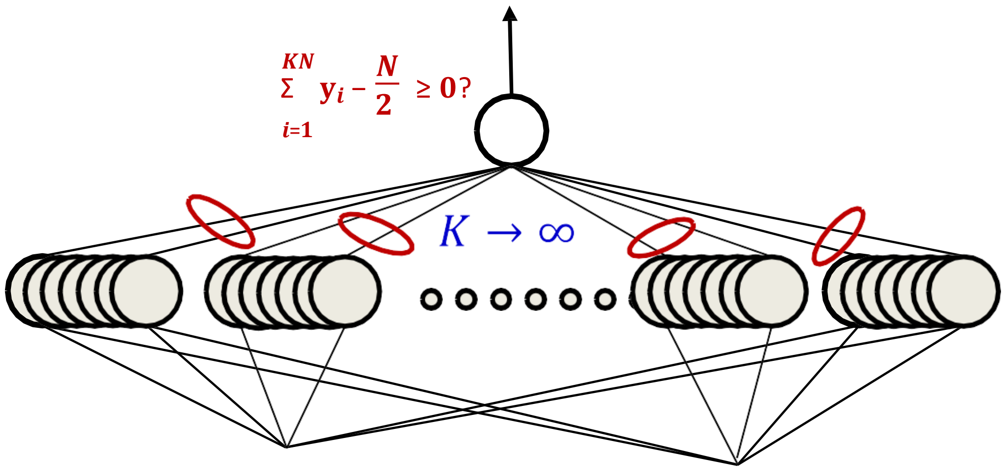
In the limit: Composing a circle

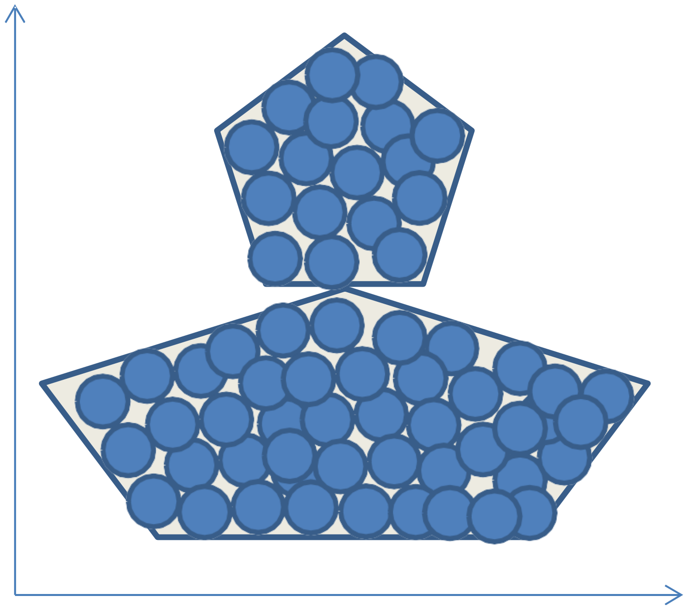




- The circle net
  - Very large number of neurons ($N \rightarrow \infty$)
  - Sum is N inside the circle, N/2 outside almost everywhere
- Circle can be at any location

Composing an arbitrary figure

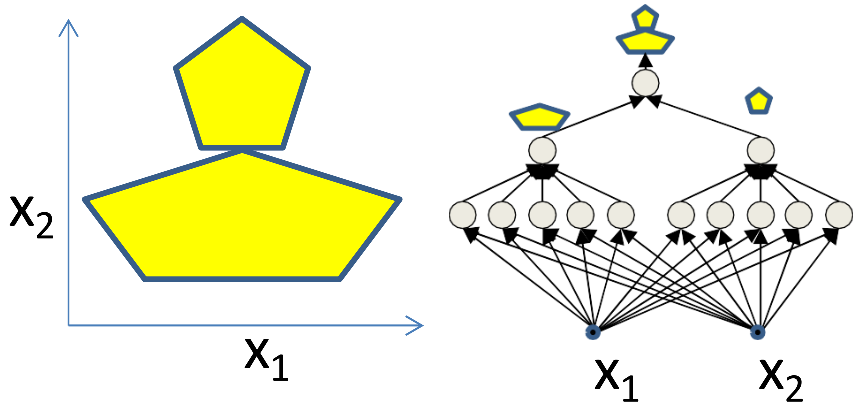


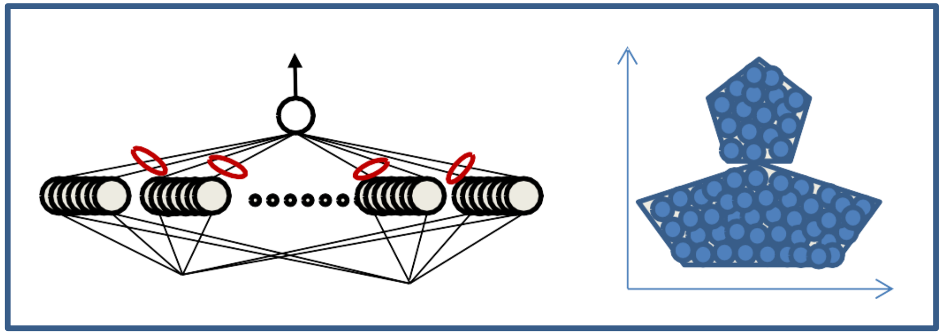


- Just fit in an arbitrary number of circles
  - More accurate approximation with greater number of smaller circles
  - Can achieve arbitrary precision
- MLPs can capture *any* classification boundary
- A *one-hidden-layer MLP* can model any classification boundary

MLPs are universal classifiers

Depth for The Universal Classifier



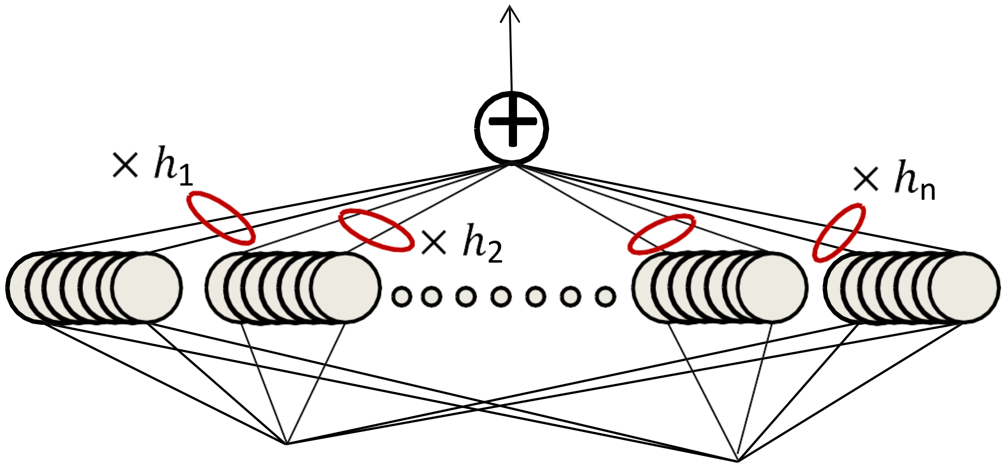


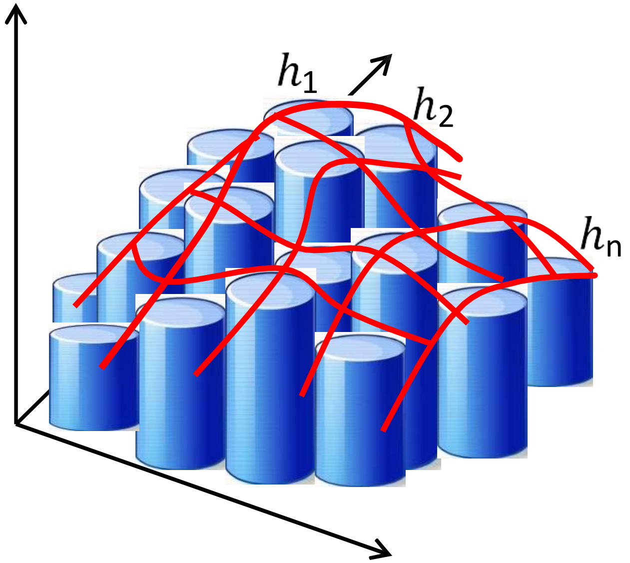
- Deeper networks can require far fewer neurons
  - 12 vs. ~infinite hidden neurons in this example

Optimal depth..

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

MLP as a continuous-valued function





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

The issue of depth

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

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

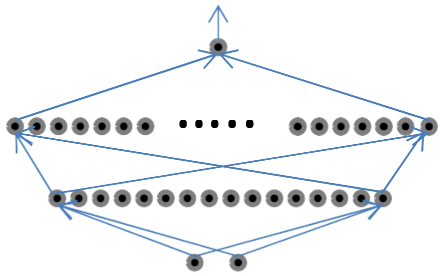
- But there are limitations…

Sufficiency of architecture

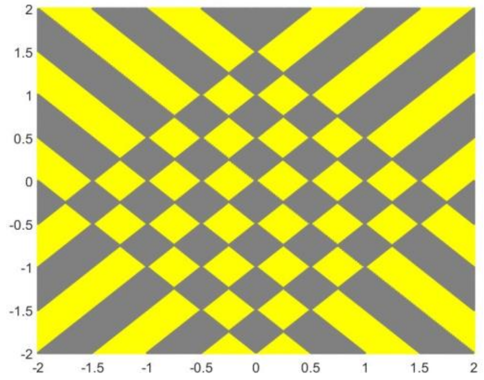
- A neural network *can* represent any function provided it has sufficient *capacity*
  - i.e. sufficiently broad and deep to represent the function

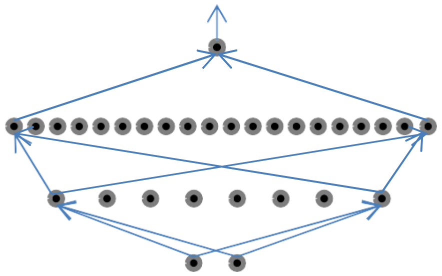
- Not all architectures can represent any function

Sufficiency of architecture



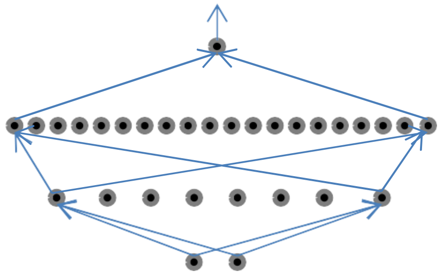
A network with 16 or more neurons in the first layer *is capable of* representing the figure to the right perfectly

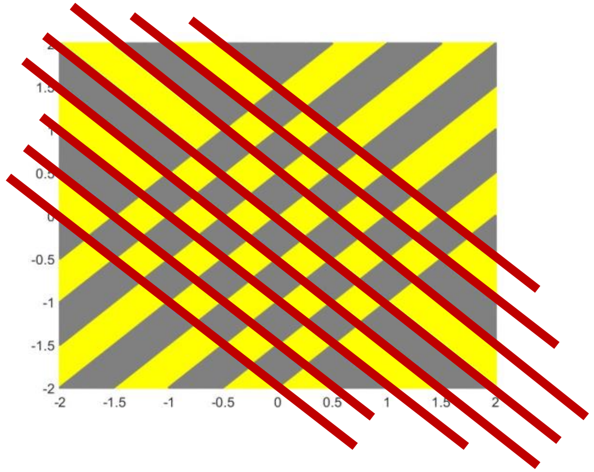


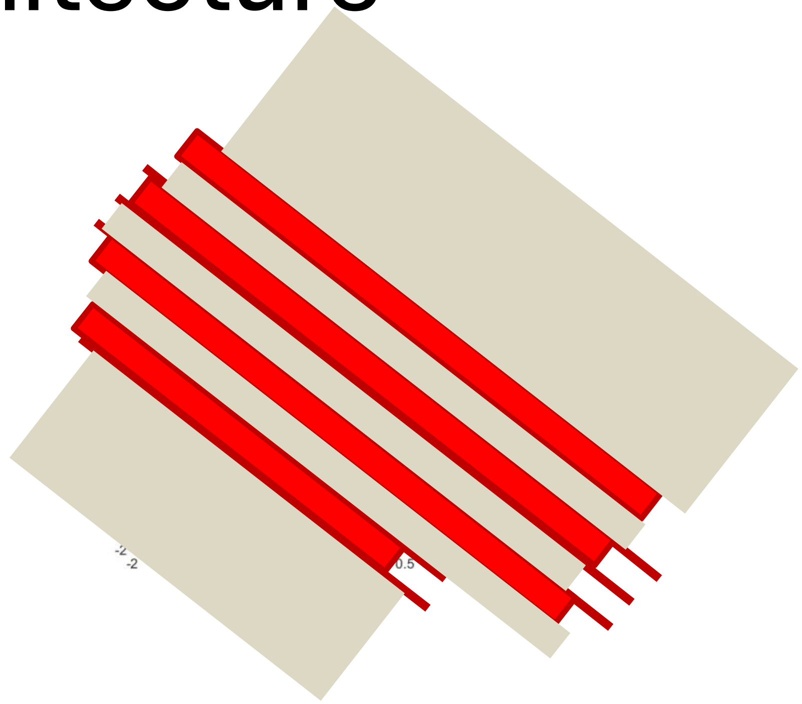


A network with less than 16 threshold-activation neurons in the first layer cannot represent this pattern exactly

Sufficiency of architecture

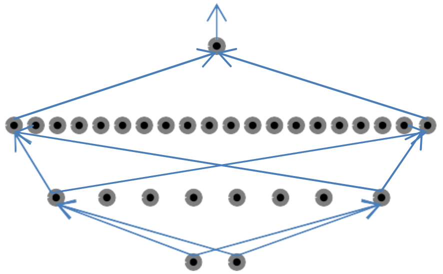


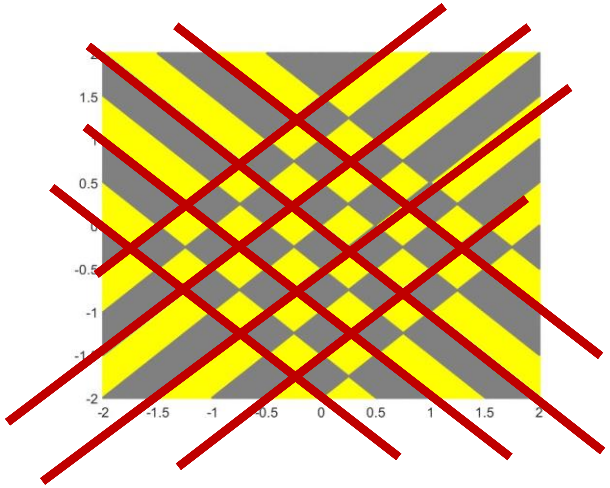


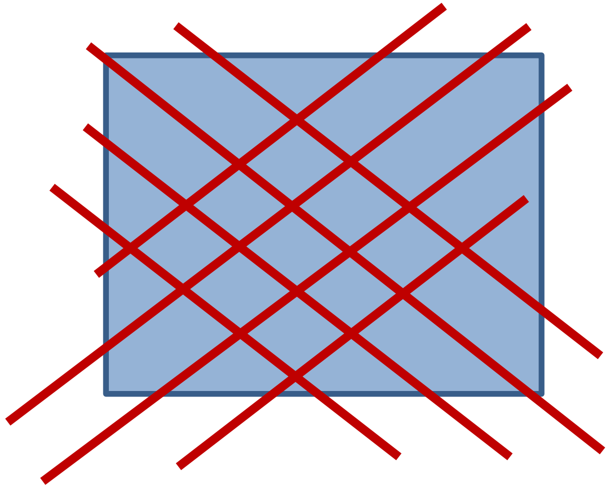


- A network with only 8 threshold neurons in the first layer may capture these 8 boundaries
- That can only give you information about which of these strips the input is in, but not *where* in the strip

Sufficiency of architecture



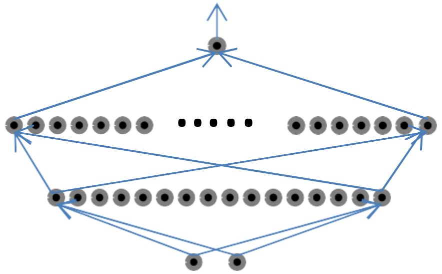




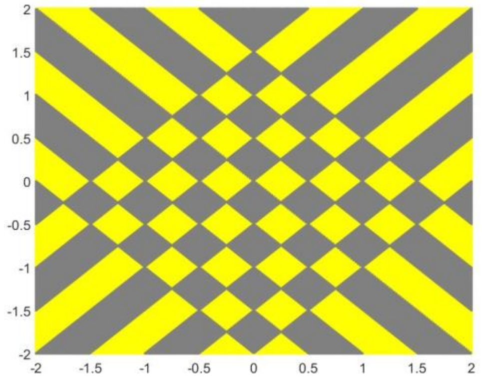
- Even if the 8 first-layer neurons capture *these* boundaries…

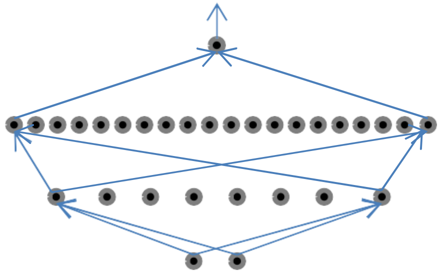
- … they can only place you in one of these 25 cells, but cannot inform you of *where* in the cell

Sufficiency of architecture

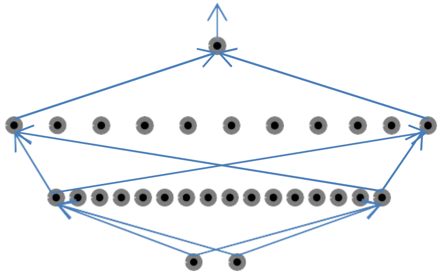


A network with 16 or more neurons in the first layer *is capable of* representing the figure to the right perfectly





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

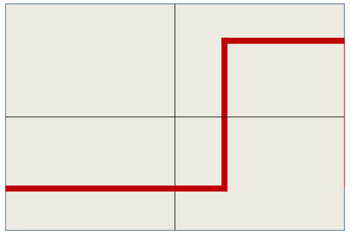


A 2-layer network with 16 neurons in the first layer cannot represent the pattern **with less than 40 neurons in the second layer**

- Similar restrictions apply to higher layers
- Regardless of depth, every layer must be sufficiently wide in order to capture the function

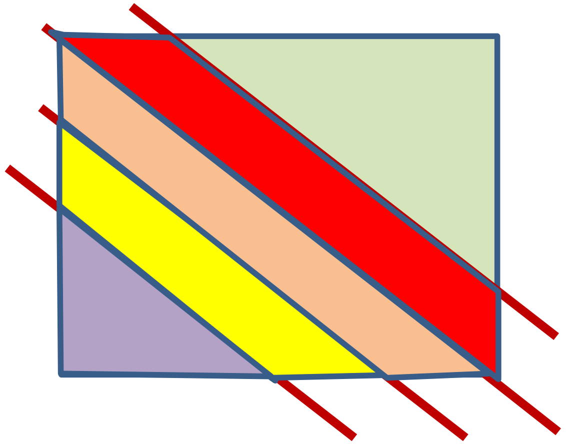
- Not all architectures can represent any function

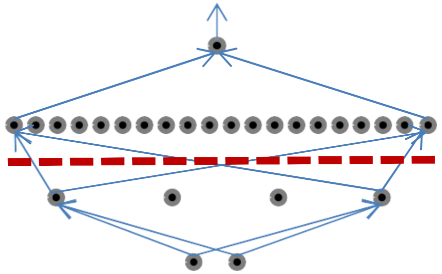
Sufficiency of architecture



This effect is because we use the threshold activation

It *gates* information in the input from later layers

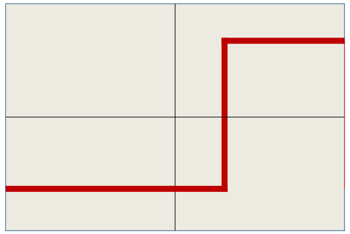


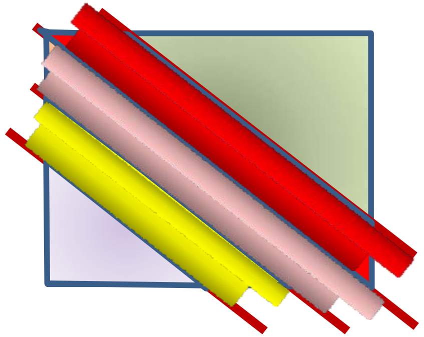


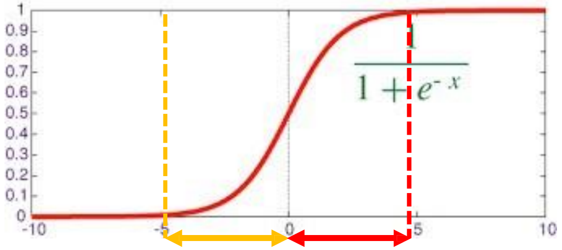
The pattern of outputs within any colored region is identical

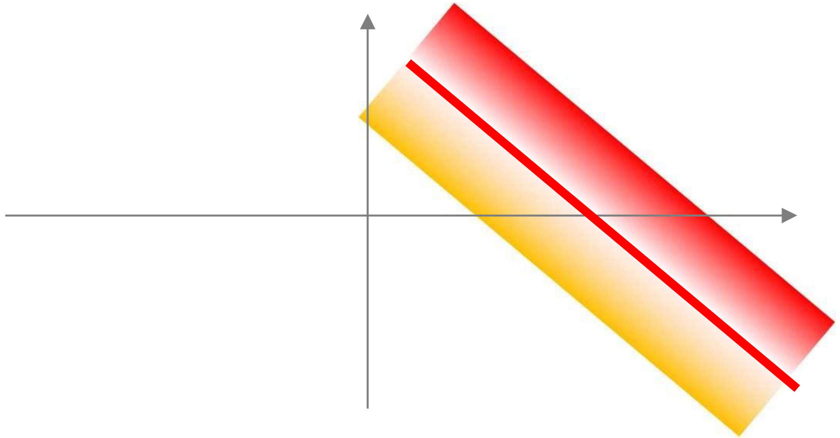
Subsequent layers do not obtain enough information to partition them

Sufficiency of architecture









Continuous activation functions result in graded output at the layer

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

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

Width vs. Activations vs. Depth

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

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

- In summary, Deeper MLPs can achieve the same precision with far fewer neurons, but must still have sufficient *capacity*
  - The activations must pass information through
  - Each layer must still be sufficiently wide to convey all relevant information to subsequent layers

Lessons today

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

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