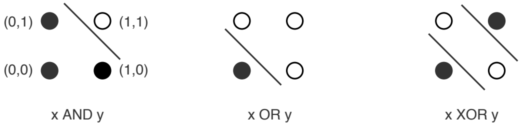
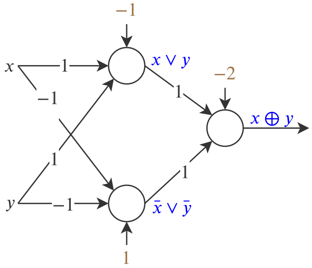
MLP and the XOR Problem

- We have seen that a single perceptron cannot solve the XOR problem because *XOR is not a linear classification problem*.



- No single line can separate the 0s (black) from the 1s (white).
- But 3 perceptrons arranged in 2 layers can solve it.

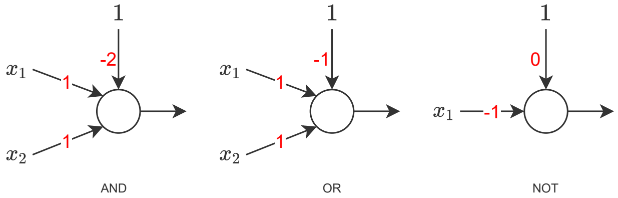


Perceptrons can do everything!

- In this lecture, we will see that multilayer perceptrons (MLPs) can model
  1. *any* Boolean function,
  2. *any* classification boundary, and
  3. *any* continuous function.

MLPs and Boolean Functions

- A single perceptron can model the basis set {AND, OR, NOT} of logic gates.



- All Boolean functions can be written using combinations of these basic gates.
- Therefore, combinations of perceptrons (MLPs) can model all Boolean functions.
- However, there is the issue of *width*.

MLPs and Boolean Functions
Width

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

- A Boolean function of N variables has 2^N different input combinations.
- Disjunctive normal form (DNF) models the truth values (1s only).

$$f = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$

- DNF corresponds to OR of AND gates.

Reducible DNF

x \ yz	00	01	11	10
0	0	1	0	0
1	1	1	1	1

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$$f = \bar{x}\bar{y}z + x\bar{y}\bar{z} + x\bar{y}z + xyz + xy\bar{z}$$
$$= x + \bar{y}z$$

Irreducible DNF

x \ yz	00	01	11	10
0	0	1	0	1
1	1	0	1	0

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$f = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$

Maximum possible ANDs in DNF is 2^{N-1}.

Reducible DNF

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

x \ yz	00	01	11	10
0	0	1	0	0
1	1	1	1	1

Karnaugh-map
K-map

$$f = \bar{x}\bar{y}z + x\bar{y}\bar{z} + x\bar{y}z + xyz + xy\bar{z}$$
$$= x + \bar{y}z$$

Irreducible DNF

XOR

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

x \ yz	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$\frac{2^N}{2} = 2^{N-1}$$

$$f = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$

Maximum possible ANDs in DNF is 2^{N-1}.

MLPs and Boolean Functions
Width

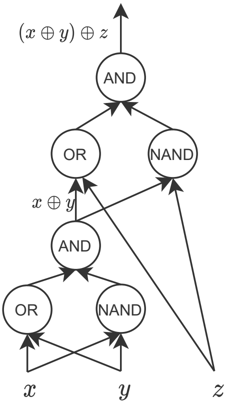
- Maximum possible ANDs in DNF is 2^{N-1}.
- Each AND corresponds to one perceptron in the hidden layer.
- *So size of hidden layers will be exponential in N*.
- OR corresponds to one perceptron in output layer.

Any Boolean function in N variables can be modelled by an MLP using
- 1 hidden layer of 2^{N-1} AND perceptrons
- followed by 1 OR perceptron.

Exponentially large width can be reduced by adding more layers.

MLPs and Boolean Functions
Depth

- Function f on last slide was actually XOR(x, y, z). It required $2^{N-1}+1$ perceptrons using 2-layers only.
- $x \oplus y \oplus z$ can be modelled using pairwise XORs as $(x \oplus y) \oplus z$.
- Corresponds to a *deep* MLP.
  - Deep: more than 2 layers.
- Requires $3(N-1)$ perceptrons.



Number of perceptrons required in single hidden layer MLP is exponential in N.
Number of perceptrons required in deep MLP is linear in N.

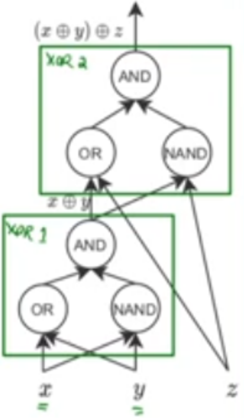
MLPs and Boolean Functions

Depth

$((x_1 \oplus x_2) \oplus x_3) \cdots x_N$
1
2

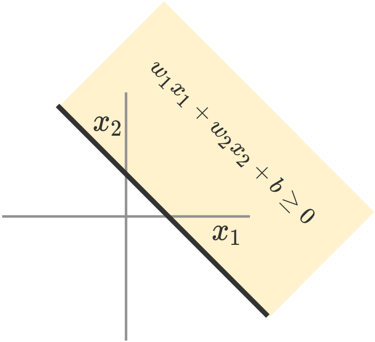
N-1 XOR,
3(N-1) perceptrons

- Function f on last slide was actually $\text{XOR}(x, y, z)$. It required $\underbrace{2^{N-1}}_{\text{AND}} + \underbrace{1}_{\text{OR}}$ perceptrons using 2-layers only.
  hidden output
- $x \oplus y \oplus z$ can be modelled using pairwise XORs as $(x \oplus y) \oplus z$.
- Corresponds to a *deep* MLP.
  - Deep: more than 2 layers.
- Requires $3(N-1)$ perceptrons. which is linear in N.



> Number of perceptrons required in single hidden layer MLP is exponential in N.
> Number of perceptrons required in deep MLP is linear in N.

MLPs and Classification Boundaries



A perceptron divides input space into 2 regions. Dividing boundary is a line.

MLPs and Classification Boundaries

$w_1x_1 + w_2x_2 + b \geq 0 \leftarrow 1$

$ax_1 + bx_2 + c < 0 \leftarrow 0$

$y = mx + c \leftarrow \boxed{\text{1-var. } x}$

$ax + by + c = 0 \leftarrow \boxed{\text{2 var. } x, y}$

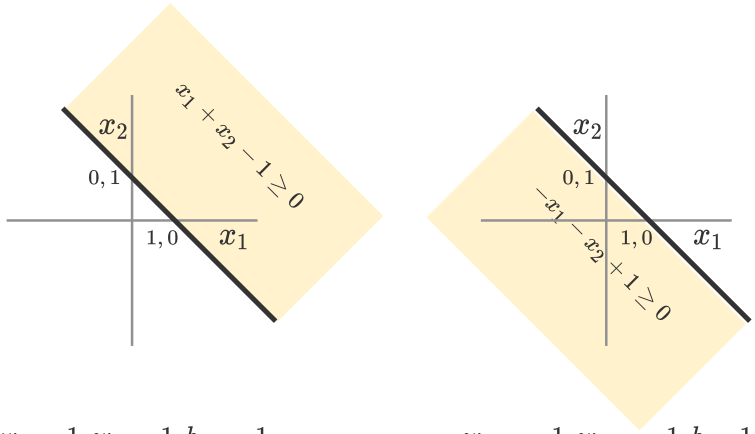
$\underset{a}{-m}x + \underset{b}{1}y \underset{c}{-c} = 0$

$\underbrace{\underset{\text{scalar}}{w_1x_1} + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5}_{\text{scalar}} \geq 0$

$\times 5$

A perceptron divides input space into 2 regions. Dividing boundary is a line.

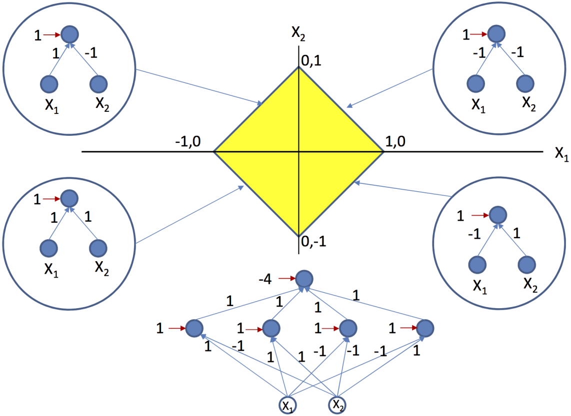
MLPs and Classification Boundaries



$w_1 = 1, w_2 = 1, b = -1$ $\qquad$ $w_1 = -1, w_2 = -1, b = 1$

Weights determine the linear boundary and classification into region 1 and region 2.

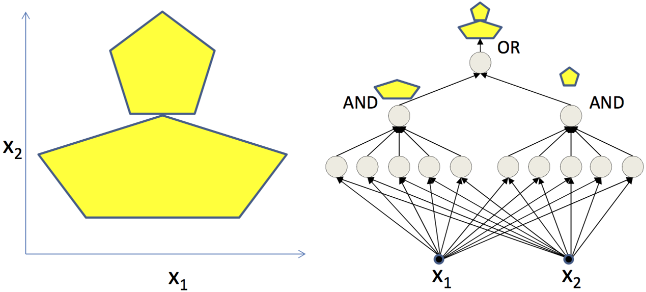
MLPs and Classification Boundaries



Yellow region modelled by ANDing 4 linear classifiers (perceptrons). First layer contains 4 perceptrons for modelling 4 lines and second layer contains a perceptron for modelling an AND gate. Source: Bhiksha Raj

MLPs and Classification Boundaries

Non-contiguous

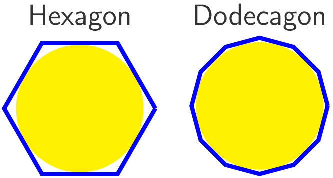


Yellow region equals OR(polygon 1, polygon 2). Each polygon equals AND of some lines. Each line equals 1 perceptron. Source: Bhiksha Raj

> Since inputs and outputs are visible, all layers in-between are known as *hidden layers*.

MLPs and Classification Boundaries
Benefit of Depth

- Can the region in the last slide be modelled using a single hidden layer?
- Detour – can you model a circular boundary? Yes, via *many* lines.

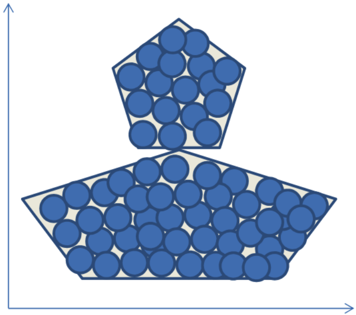


- Circle $= \lim_{k \to \infty} k$-gon.
- As number of sides approaches ∞, regular polygons approximate circles.

MLPs and Classification Boundaries

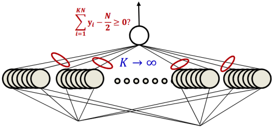
Benefit of Depth

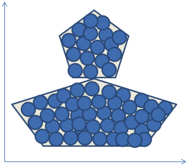
- Any shape can be modelled by filling it with *many circles*, where each circle is modelled via *many lines*.
- Precision increases as number of circles approaches ∞ and as number of lines per circle approaches ∞.



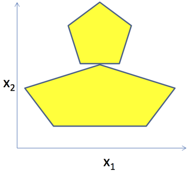
MLPs and Classification Boundaries
Benefit of Depth

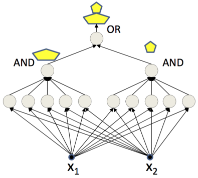
- In other words, shape equals OR(many circles) where each circle equals AND(many lines).
- Can be done with 1 *really really wide* hidden layer.





- Adding more layers *exponentially reduces* the number of required neurons.





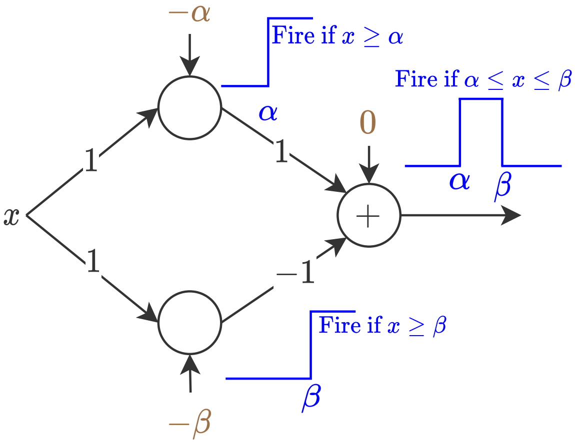
MLPs and Continuous Functions

- MLPs are *universal approximators*.

> A two-layer network with linear outputs can uniformly approximate any continuous function on a compact input domain to arbitrary accuracy, *provided* that the network has a sufficiently large number of hidden units.

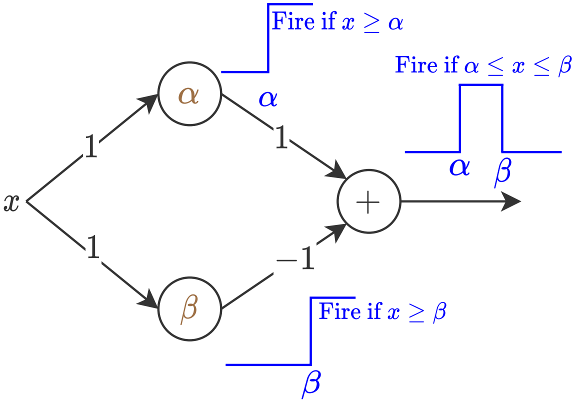
- The next few slides present a proof of this statement.

Generating a pulse using an MLP



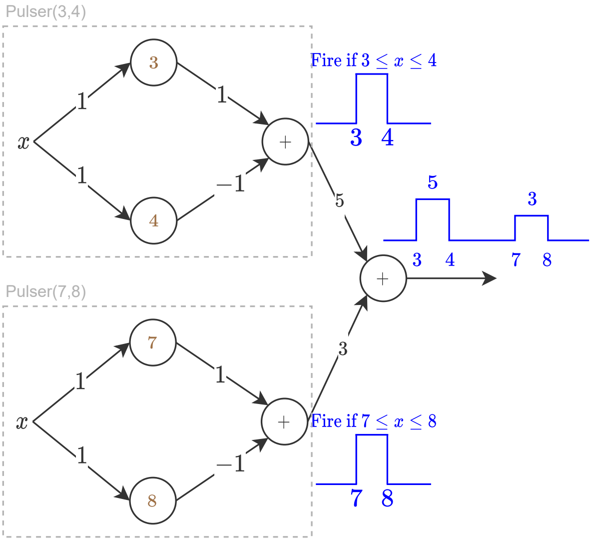
For $\alpha, \beta \in \mathbb{R}$, the pulse can be made infinitely wide when $(\beta - \alpha) \to \infty$ and infinitesimally thin when $(\beta - \alpha) \to 0$.

Generating a pulse using an MLP

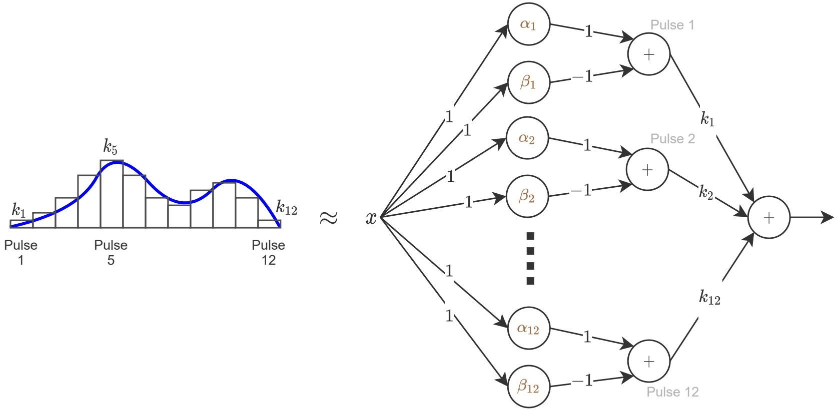


Since $\sum w_i x_i + b \geq 0 \implies \sum w_i x_i \geq -b$, we have removed each neuron's bias b by setting $-b$ as the firing threshold instead of 0.

Combining MLP Pulsers

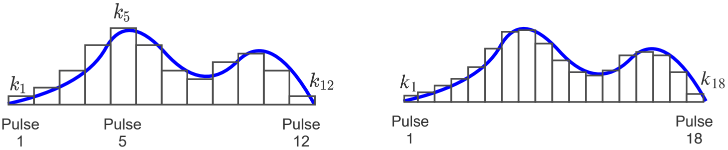


Functions as pulse combinations



Approximation using 12 pulsers. This is similar to approximation of area under a function using integration as width of strip/pulse $\delta \to 0$.

Functions as pulse combinations



- More pulsers will yield better approximation of the function.

> **Universal Approximation Theorem**
> A linear combination of 2-layer perceptrons (pulsers) can approximate any function to arbitrary precision as long as we use *enough* pulsers.

- At the cost of 3 perceptrons per pulse.

Summary

- MLP with a single hidden layer is a *universal approximator* of
    1. Boolean functions,
    2. Classification boundaries, and
    3. Continuous functions.
- Size of hidden layer needs to be exponential in number of inputs.
- Adding more layers *exponentially reduces* the number of neurons.
- Next lecture: learning of weights in a perceptron.