Machine learning

Introduction to neural nets

Mohammad-Reza A. Dehaqani

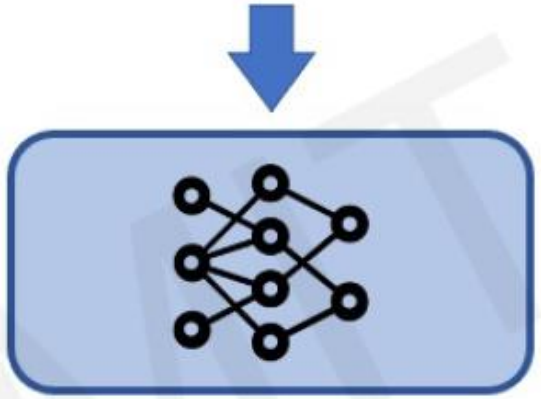
dehaqani@ut.ac.ir

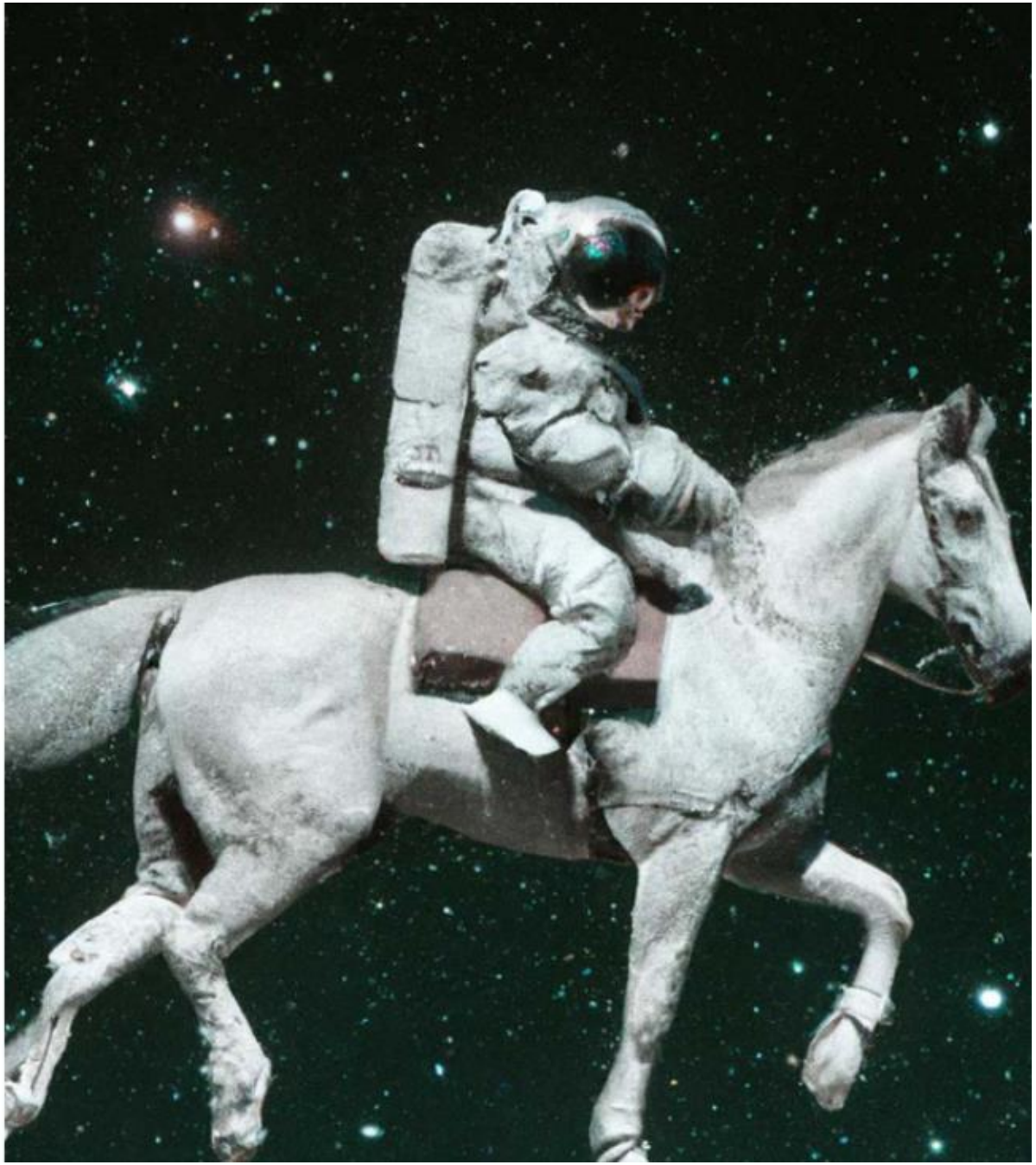
Based on MIT 6.S191

The Rise of Deep Learning

Generating Images from Natural Language

"A photo of an astronaut riding a horse."

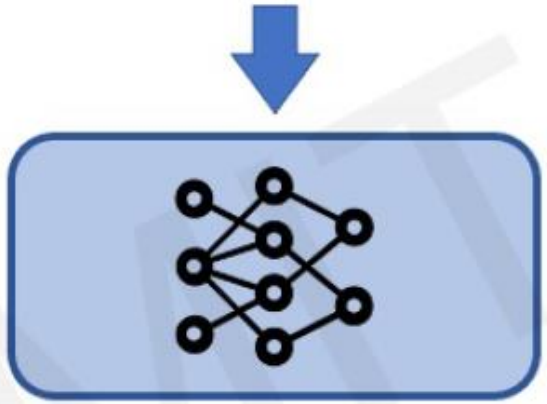




Ramesh+ *arXiv* 2022

Generating Language from Natural Language

"Write code in TensorFlow to train a neural network."



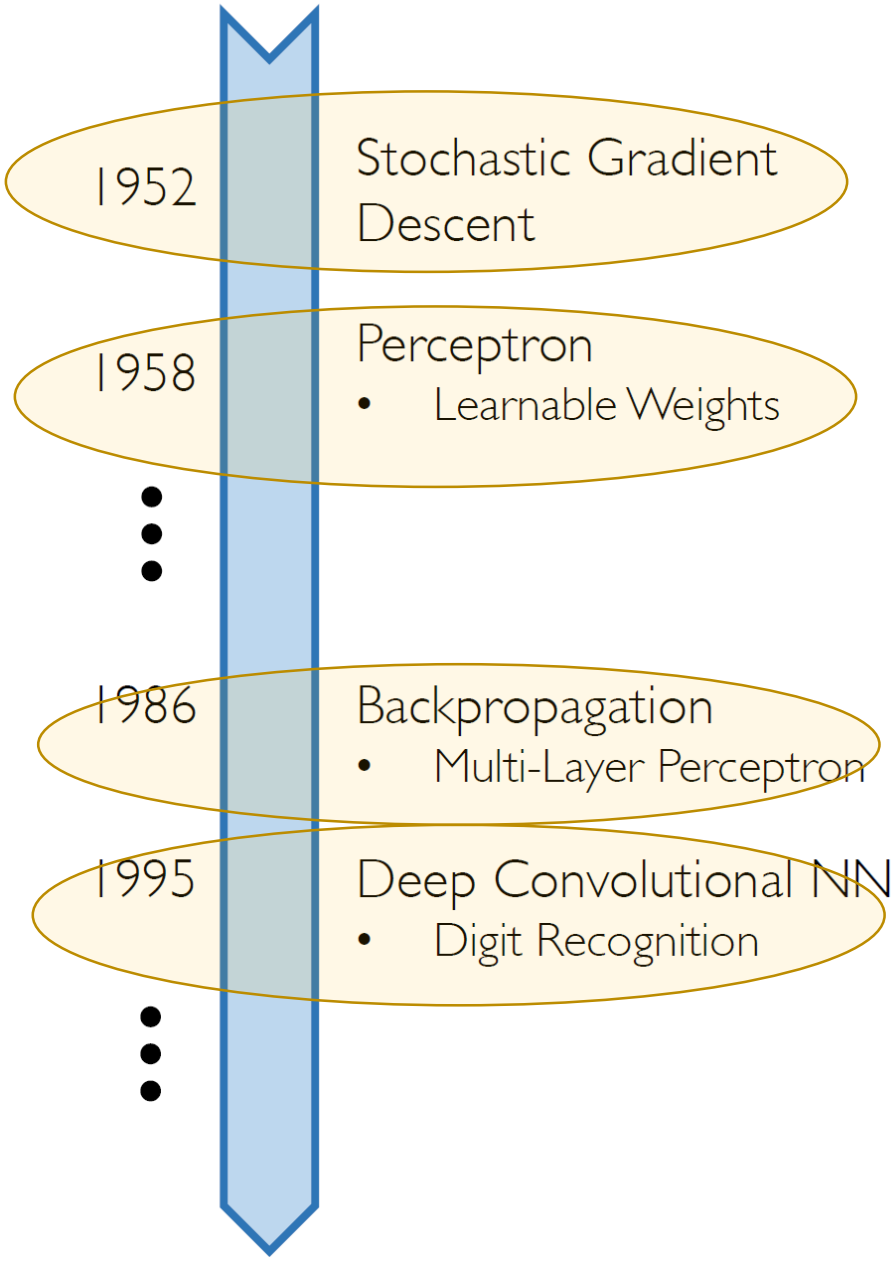
OpenAI.

Certainly! Here is an example of how you can use TensorFlow to train a neural network for binary classification:

Copy code

```
import tensorflow as tf

# Load the data
(x_train, y_train), (x_test, y_test) = load_data()

# Define the model
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=x_train.shape[1:]),
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=
['accuracy'])

# Train the model
model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=
(x_test, y_test))
```

This code assumes that you have a function `load_data()` that returns the training and test datasets as tuples of NumPy arrays. The input shape of the model is determined by the shape of the training data, and the model has two dense layers with 16 and 1 units, respectively, with ReLU and sigmoid activations. The model is then compiled with an Adam optimizer and a binary cross-entropy loss function, and is trained using the `fit()` method.



Why Now?

Neural Networks date back decades, so why the resurgence?

1. Big Data

- Larger Datasets
- Easier Collection & Storage

IMAGENET

WIKIPEDIA
The Free Encyclopedia

2. Hardware

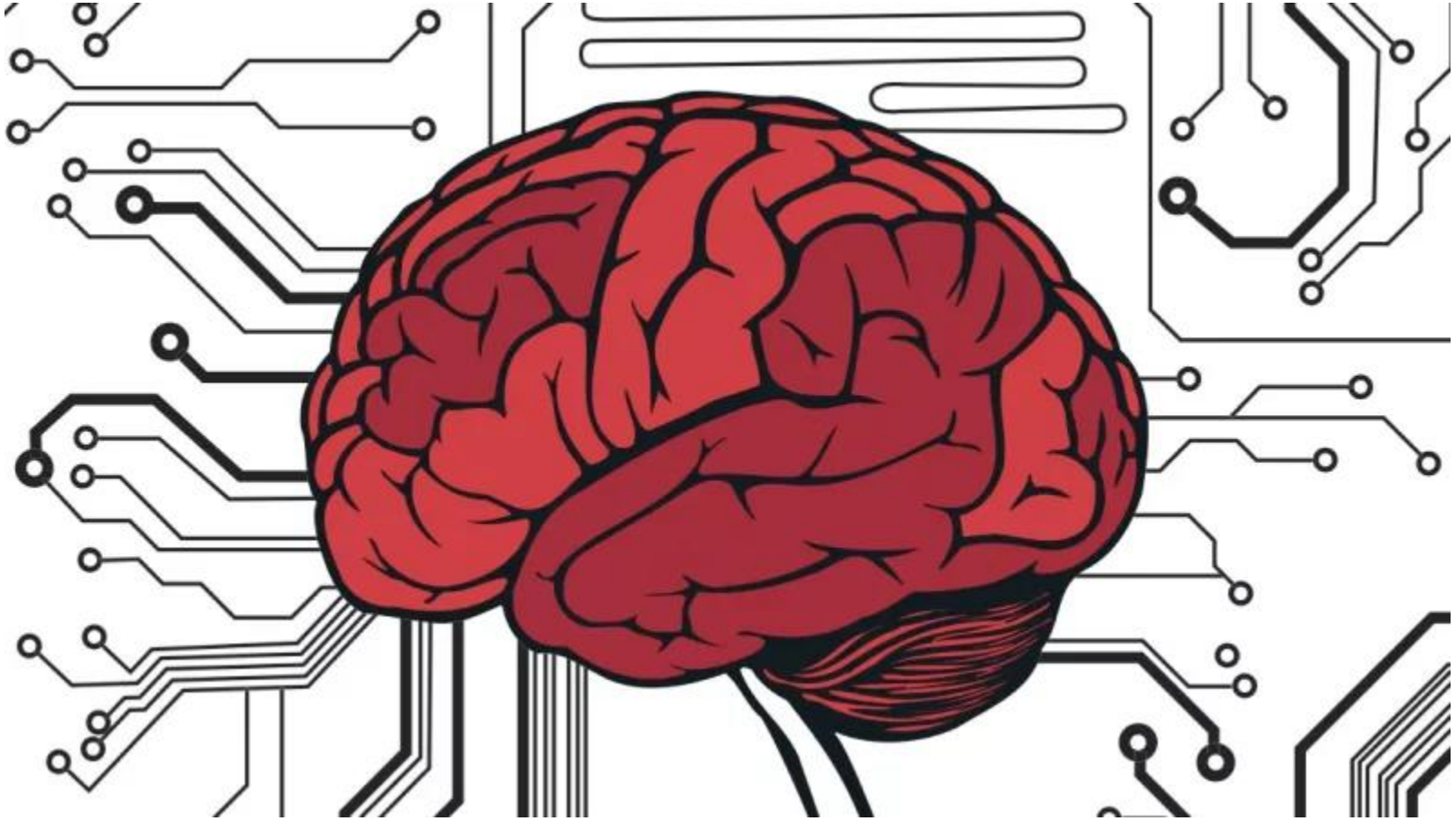
- Graphics Processing Units (GPUs)
- Massively Parallelizable

3. Software

- Improved Techniques
- New Models
- Toolboxes

TensorFlow

Brain as biological computing machines



complex computational system, consisting of some 100 billion neurons, connected by an estimated 100 trillion synapses

Herculano-Houzel, Suzana. "The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost." *Proceedings of the National Academy of Sciences*109.Supplement 1 (2012): 10661-10668.

Brain vs. Silicon

1 mm^3 of cortex:

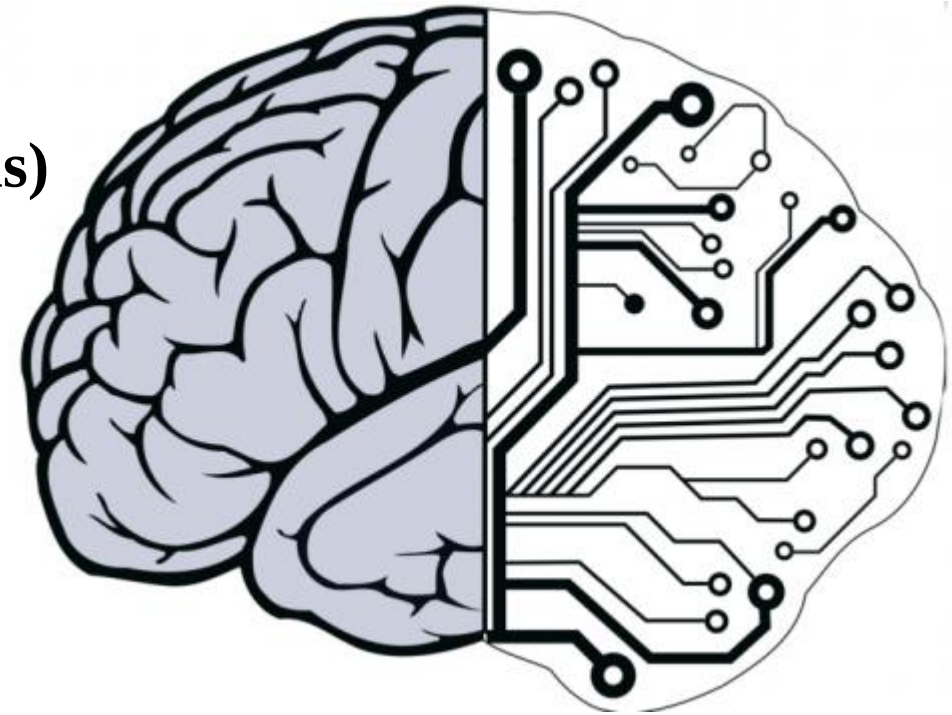
50,000 neurons
1000 connections/neuron
(=> 50 million connections)
4 km of axons

whole brain (2 kg):

10^{11} neurons
10^{14} connections
8 million km of axons
20 watts

Parallelism
Connectivity

Speed
Scalability



1 mm^2 of a CPU:

1 million transistors
2 connections/transistor
(=> 2 million connections)
.002 km of wire

whole CPU:

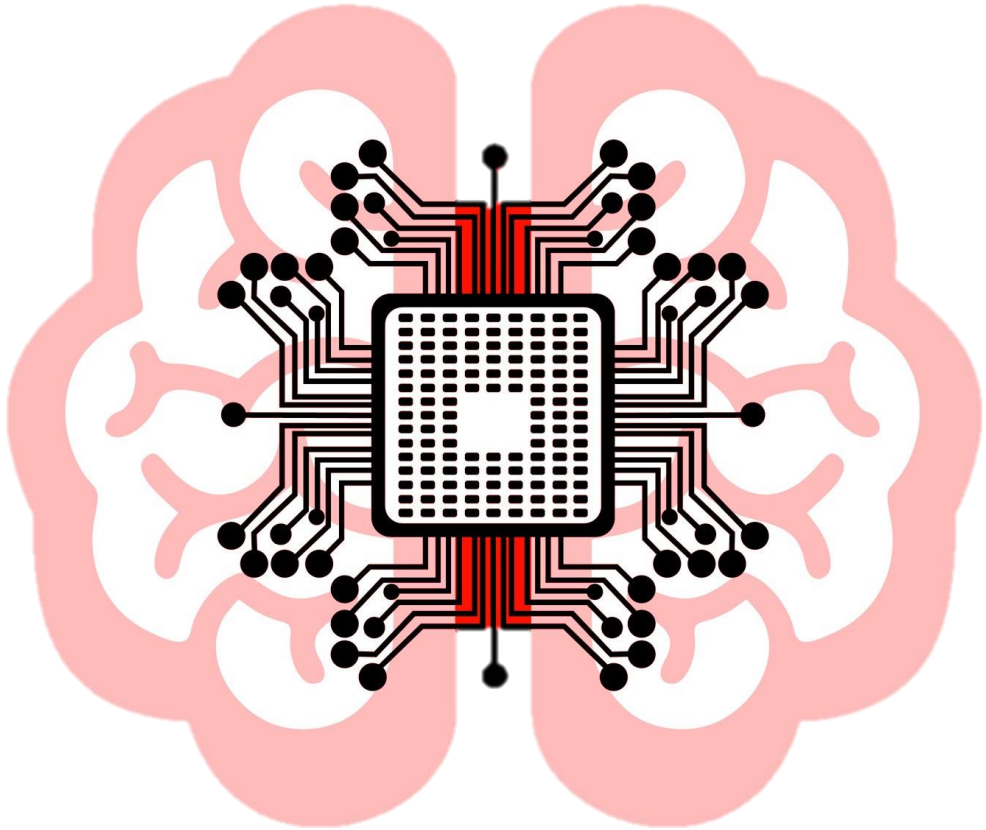
10^9 transistors
2×10^9 connections
2 km of wire
scaled to brain: MW

Bassett, Danielle Smith, and E. D. Bullmore. "Small-world brain networks." *The neuroscientist*12.6 (2006): 512-523.

Mink, Jonathan W., Robert J. Blumenschine, and David B. Adams. "Ratio of central nervous system to body metabolism in vertebrates: its constancy and functional basis." *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology* 241.3 (1981): R203-R212.

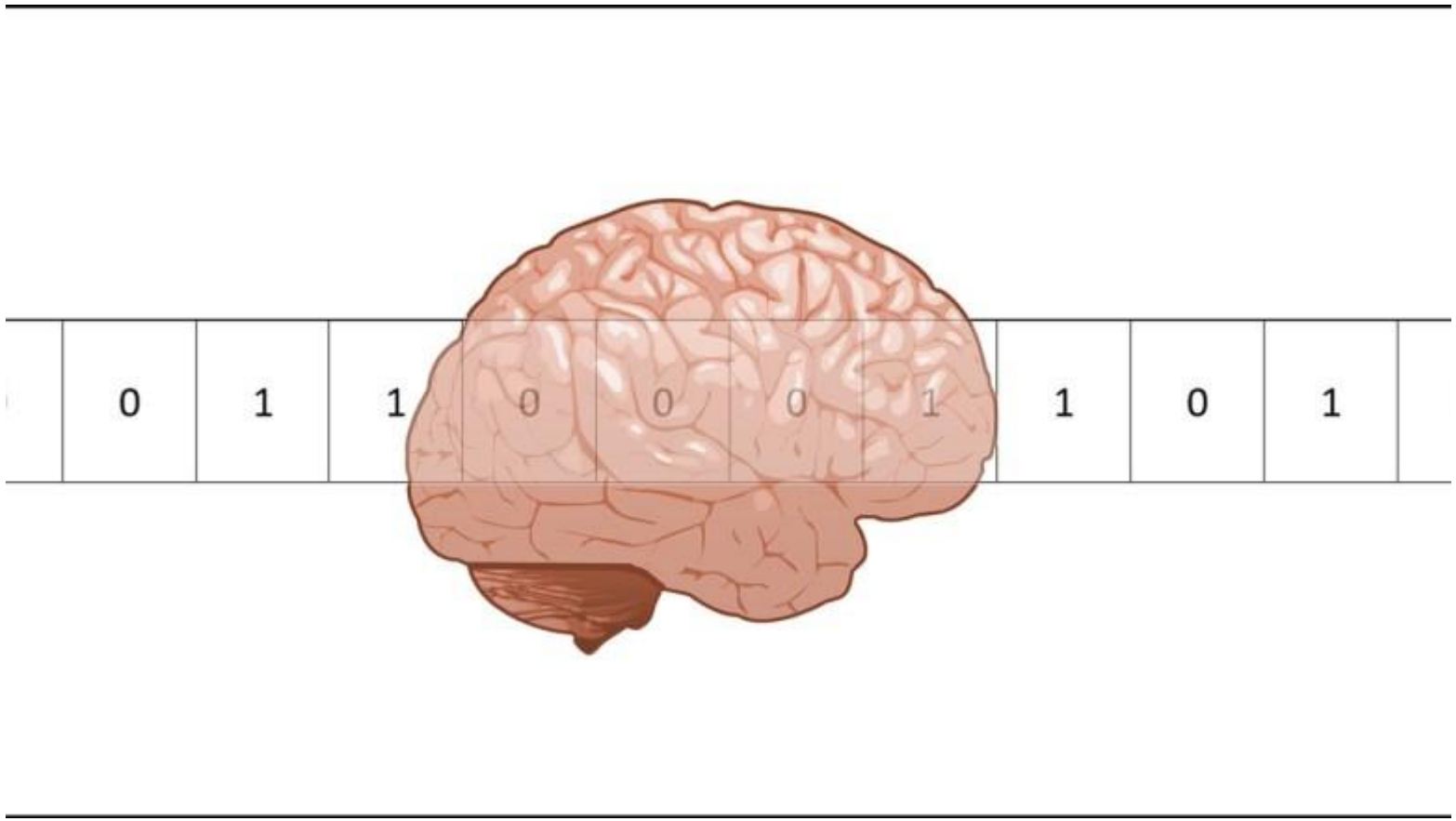
Today

Neuroscientists increasingly speak of neuronal '**computations**' and the '**circuits**' responsible for behaviors; distant brain regions **communicate** to form '**networks**' of activity.



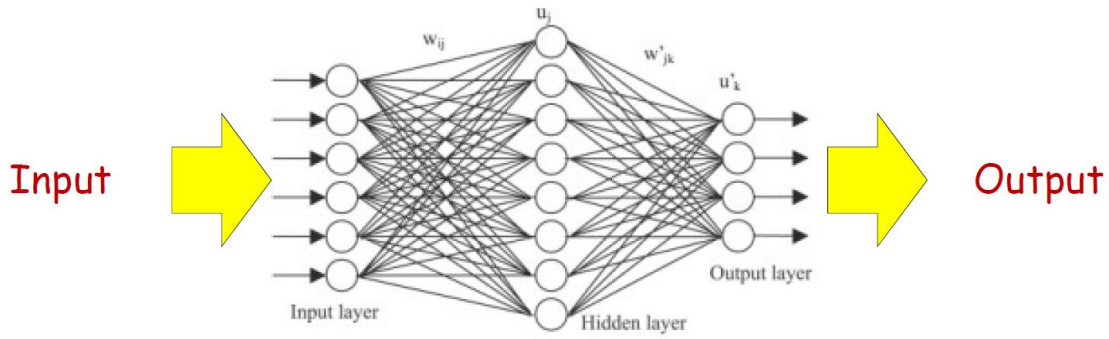
'**Computational**' perspective on neuroscience goes beyond the metaphor of 'the brain is a computer'

computational science provides a rigorous formal framework and tools for reasoning about information-processing systems



NNets in AI

- Model memory
  - Loopy networks can "remember" patterns
- Represent probability distributions
- Over integer, real and complex-valued domains
- MLPs can model both a **posteriori** and a **priori** distributions of data
- The network as a function



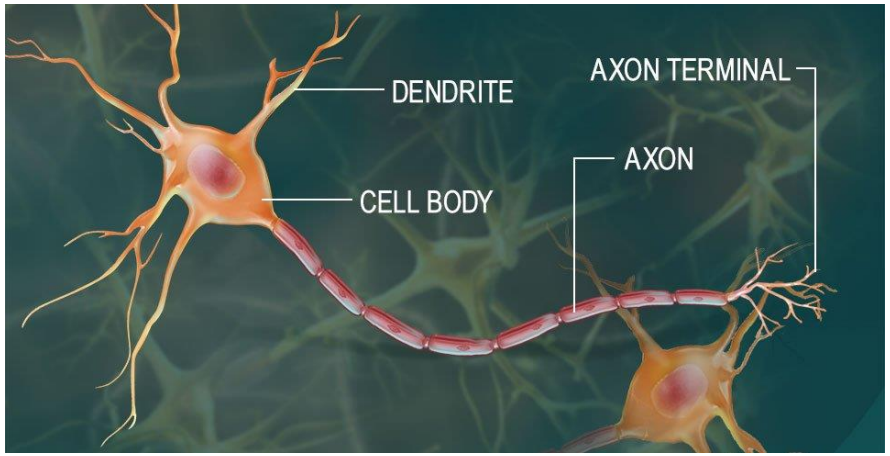
- Inputs are numeric vectors: Numeric representation of input, e.g. audio, image, game state, etc.
- Outputs are numeric scalars or vectors: Numeric "encoding" of output from which actual output can be derived
  - E.g. a score, which can be compared to a threshold to decide if the input is a face or not
- Output may be **multi-dimensional**, if task requires it

Early Models of Human Cognition

- Associationism: Humans learn **through association**
  - 400BC-1900AD: Plato, David Hume, Ivan Pavlov..
- "**Pairs** of thoughts become associated based on the organism's **past experience**"
- Learning is a mental process that forms associations between temporally related phenomena
- Aristotle's four laws of association:
  - The law of contiguity. Things or events that occur **close together** in space or time get linked together
  - The law of frequency. The more often two things or events **are linked**, the more powerful that association.
  - The law of similarity. If two things are similar, the thought of one will trigger the thought of the other
  - The law of contrast. Seeing or recalling something may trigger the recollection of something opposite

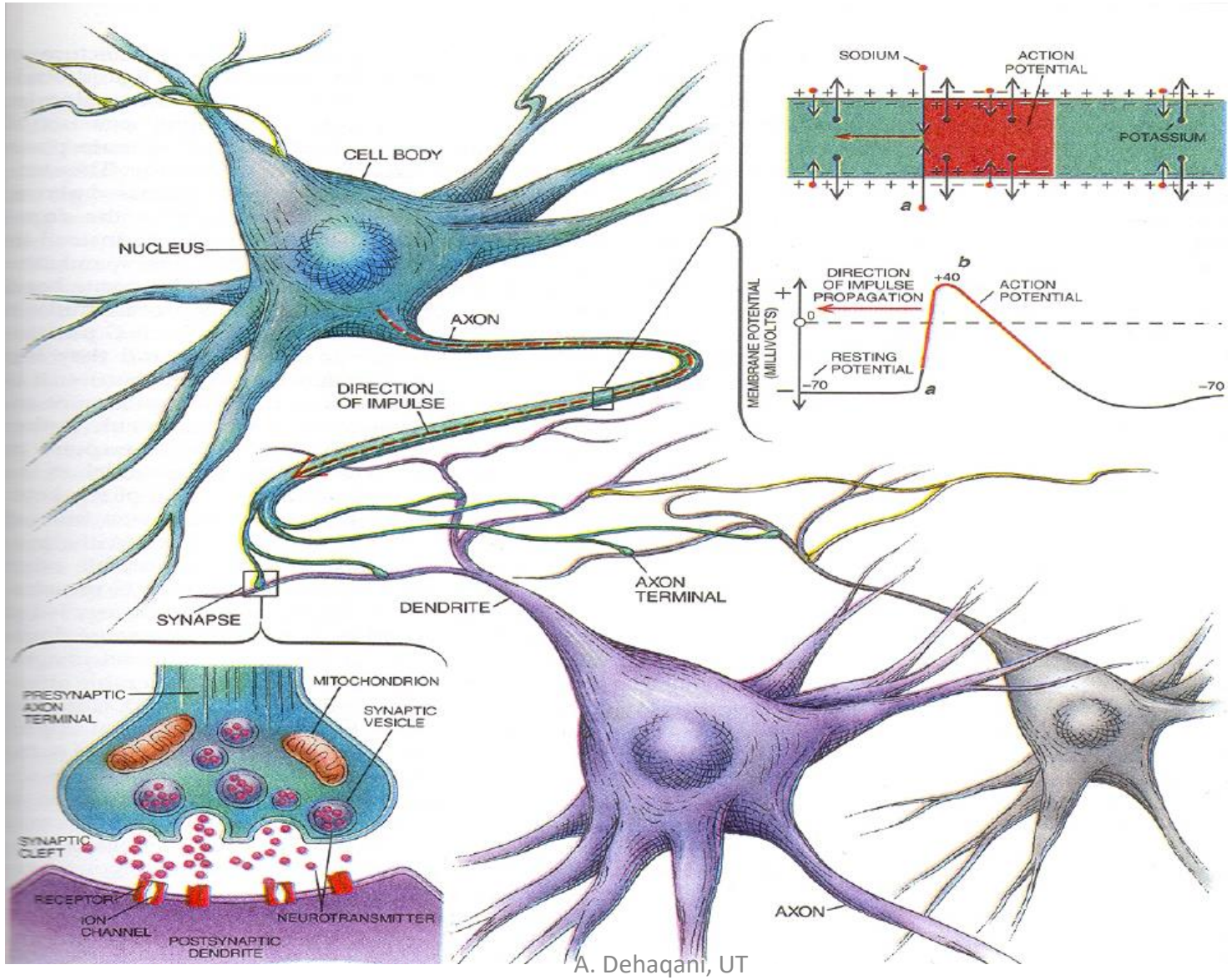
Modelling the brain

- What are the units?



- Signals come in through the dendrites into the Soma
- A signal goes out via the axon to other neurons

How neurons communicate



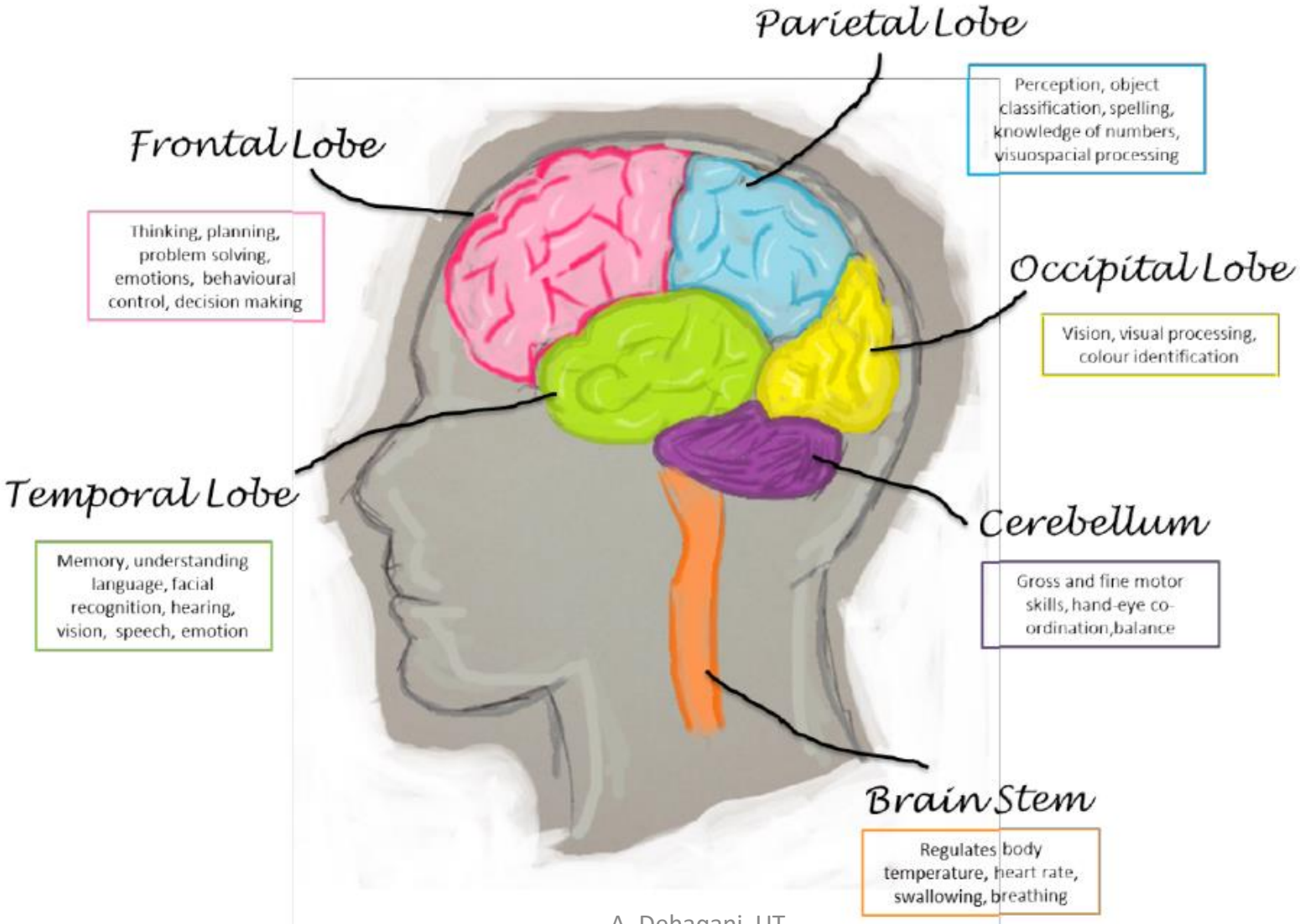
Basic neuron's concepts

- Neuron: A cell in brain/nervous system for thinking/communication
- Action potential or spike: An electrochemical impulse fired by a neuron to communicate w/other neurons
- Axon: The limb(s) along which the action potential propagates; "output"
- Dendrite: Smaller limbs by which neuron receives info; "input"
- Synapse: Connection from one neuron's axon to another's dendrite
- Neurotransmitter: Chemical released by axon terminal to stimulate dendrite

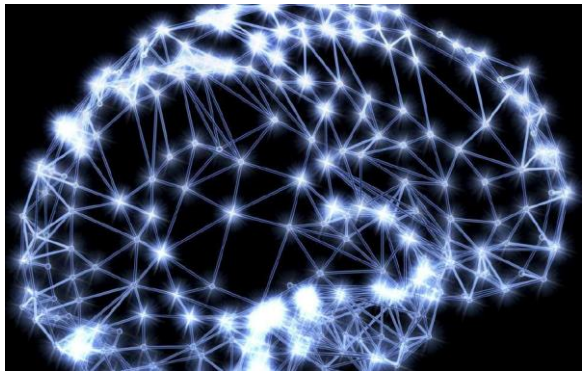
Analogies

- Output of unit ~ firing rate of neuron
- Weight of connection ~ synapse strength
- Positive weight ~ excitatory neurotransmitter (e.g. glutamine)
- Negative weight ~ inhibitory neurotransmitter (e.g. GABA, glycine)–
- Linear combo of inputs ~ summation
- Logistic/sigmoid fn ~ firing rate saturation
- Weight change/learning ~ synaptic plasticity
- Hebb's rule (1949): "Cells that fire together, wire together."
  - There are simple computer learning algorithms based on Hebb's rule. It can work, but it's generally not nearly as fast or effective as backpropagation
- Brains probably do not do backpropagation though some people do speculate otherwise.

Our brain

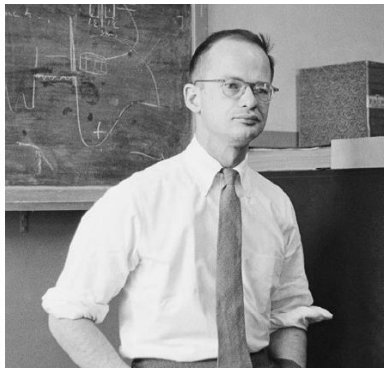

Parietal Lobe
Perception, object classification, spelling, knowledge of numbers, visuospacial processing
Frontal Lobe
Thinking, planning, problem solving, emotions, behavioural control, decision making
Occipital Lobe
Vision, visual processing, colour identification
Temporal Lobe
Memory, understanding language, facial recognition, hearing, vision, speech, emotion
Cerebellum
Gross and fine motor skills, hand-eye co-ordination,balance
Brain Stem
Regulates body temperature, heart rate, swallowing, breathing

But how do we store them? Dawn of Connectionism



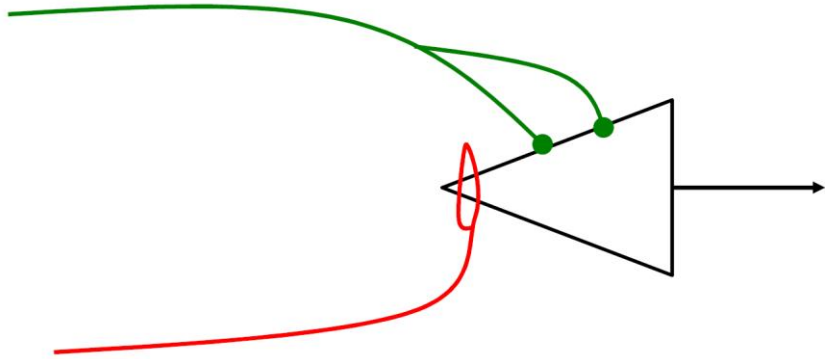
- Observation: The Brain: is a mass of interconnected neurons (Mid 1800s)
- Many neurons connect in to each neuron
- Each neuron connects out to many neurons
- Alexander Bain:
- philosopher, mathematician,

  logician, linguist, professor
- 1873: The information is in the connections (The mind and body (1873))



McCullough and Pitts

- The Doctor and the Hobo.
  - Warren McCulloch: Homeless wannabe logician
  - Walter Pitts: Neurophysician
- The McCulloch and Pitts model
  - Pitts was only 20 years old
  - Neuron is an "all-or-none" process
    - Excitatory synapse: Transmits weighted input to the neuron
    - Inhibitory synapse: Any signal from an inhibitory synapse forces output to zero
    - The activity of any inhibitory synapse absolutely prevents excitation of the neuron at that time. Regardless of other inputs at this time

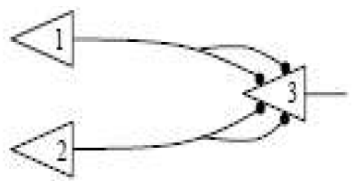


McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

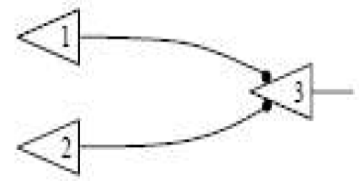
McCulloch and Pitts Model

- Could compute arbitrary Boolean propositions
  - Since any Boolean function can be emulated, any Boolean function can be composed
- Models for memory
  - Networks with loops can "remember"
  - Lawrence Kubie (1930): Closed loops in the central nervous system explain memory
- Didn't provide a learning mechanism

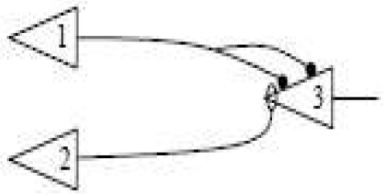
$N_3(t) = N_1(t\text{-}1)$ or $N_2(t\text{-}1)$

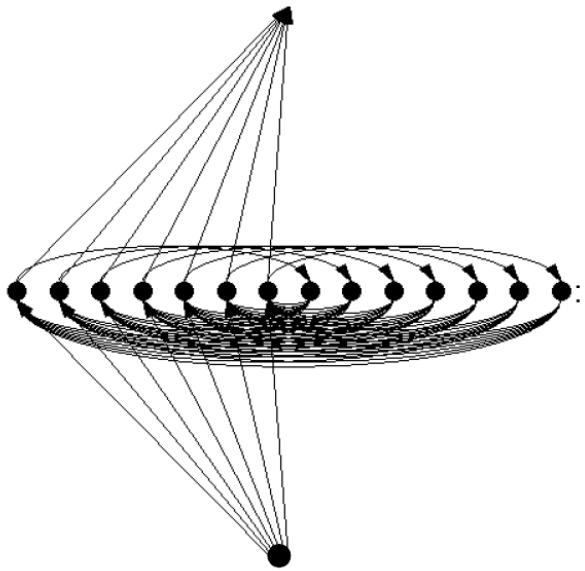


$N_3(t) = N_1(t\text{-}1)$ and $N_2(t\text{-}1)$



$N_3(t) = N_1(t\text{-}1)$ and $\sim N_2(t\text{-}1)$



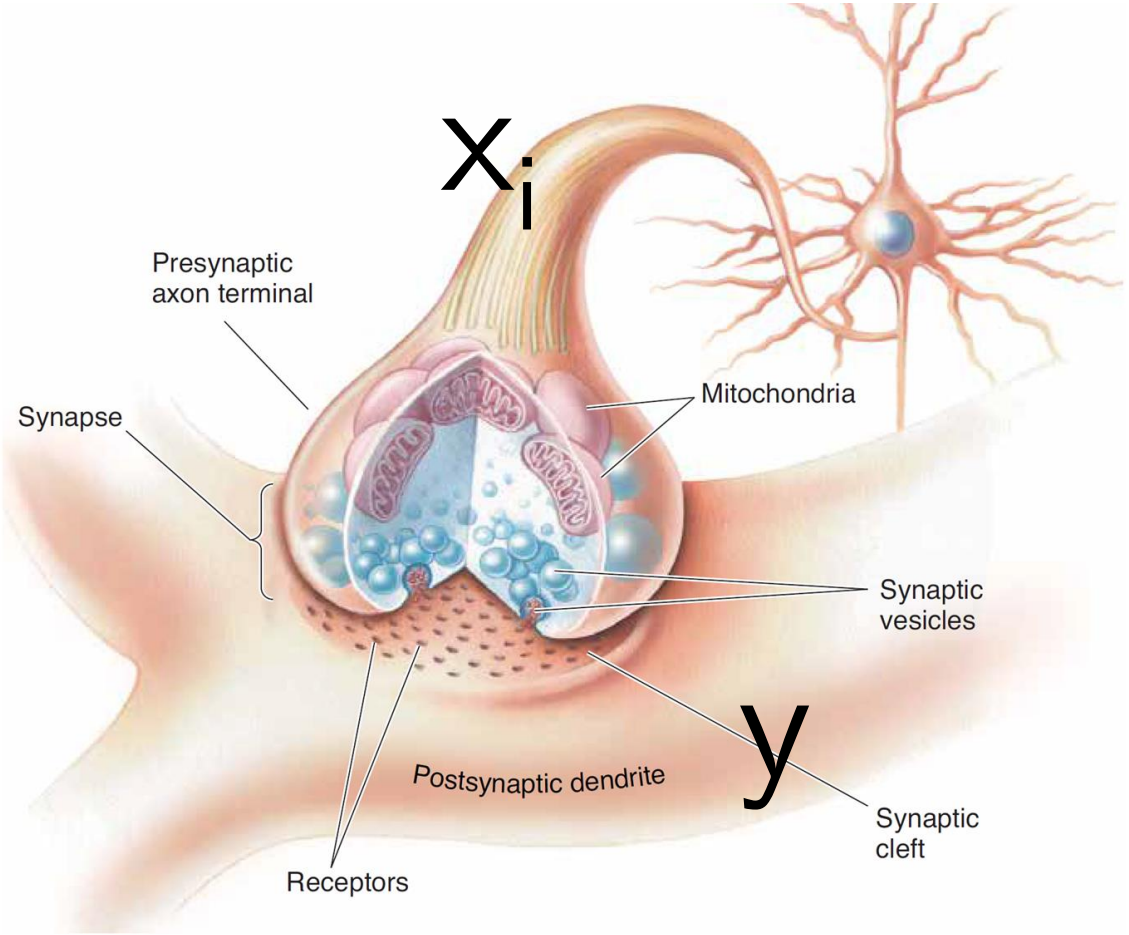


Hebbian Learning

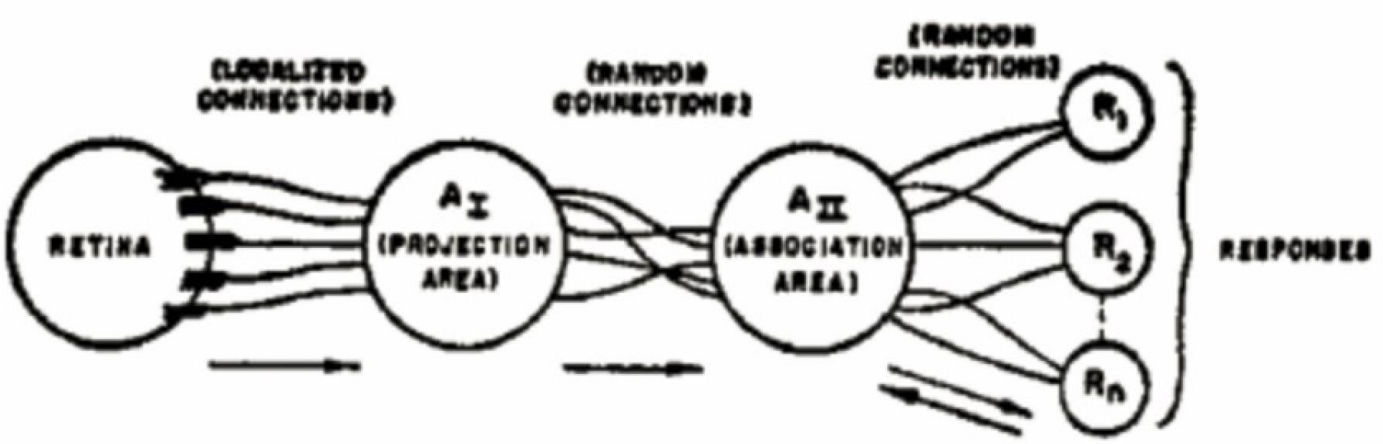
- If neuron x_i repeatedly triggers neuron y, the synaptic knob connecting x_i to y gets larger
- In a mathematical model:

$$w_i = w_i + \eta x_i y$$

- Weight w_i of the i^{th} neuron's input to output neuron y
- This simple formula is actually the basis of many learning algorithms in ML
- This models is fundamentally unstable
  - Stronger connections will enforce themselves
  - No notion of "competition"
  - No reduction in weights
  - Learning is unbounded

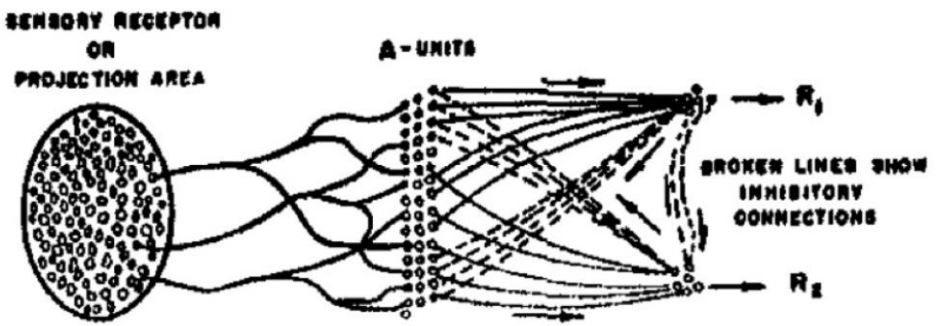


Rosenblatt's perceptron

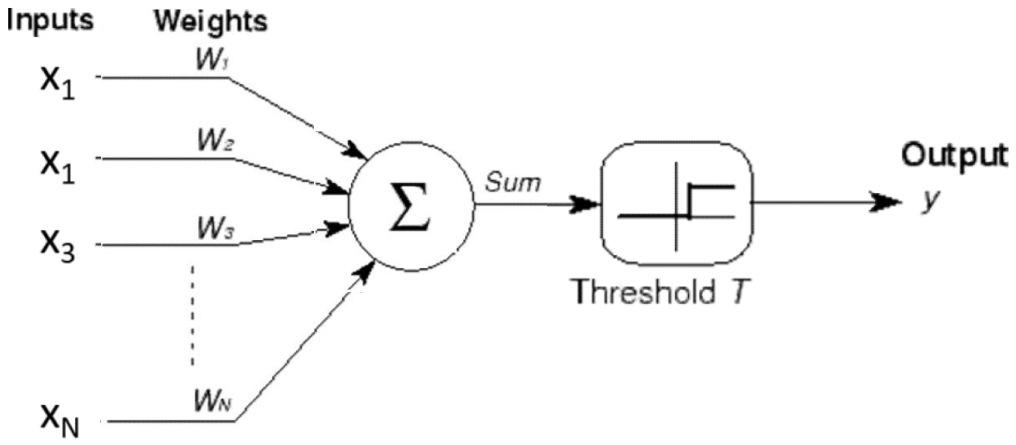


Frank Rosenblatt
– Psychologist, Logician

- Original perceptron model
  - Groups of sensors (S) on retina combine onto cells in association area A1
  - Groups of A1 cells combine into Association cells A2
  - Signals from A2 cells combine into response cells R
  - All connections may be excitatory or inhibitory
- Even included feedback between A and R cells



Simplified mathematical model



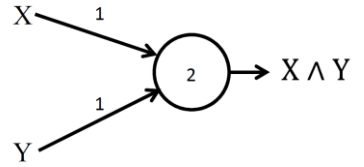
- Number of inputs combine linearly
- – Threshold logic: Fire if combined input exceeds threshold

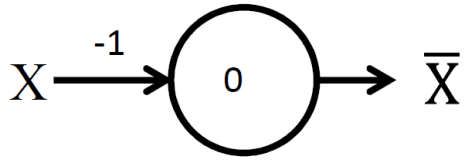
$$Y = \begin{cases} 1 \;\; if \sum_i w_i x_i - T > 0 \\ 0 \qquad\qquad else \end{cases}$$

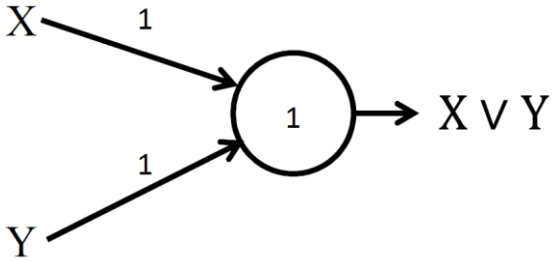
Also provided a learning algorithm

$$W = W + \eta(d(x) - y(x))x$$

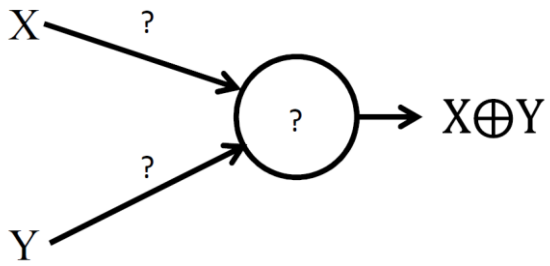
- Sequential Learning:
  - d(**x**) is the **desired output** in response to input **x**
  - Y(**x**) is the **actual** output in response to **x**
- Boolean tasks
  - Update the weights whenever the perceptron output is wrong
  - Proved convergence for **linearly separable classes**
- Easily shown to mimic any Boolean gate







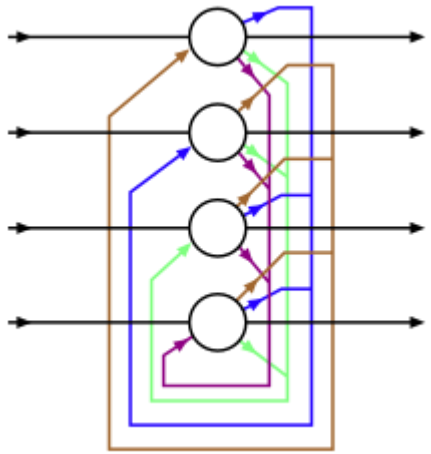
- **But**

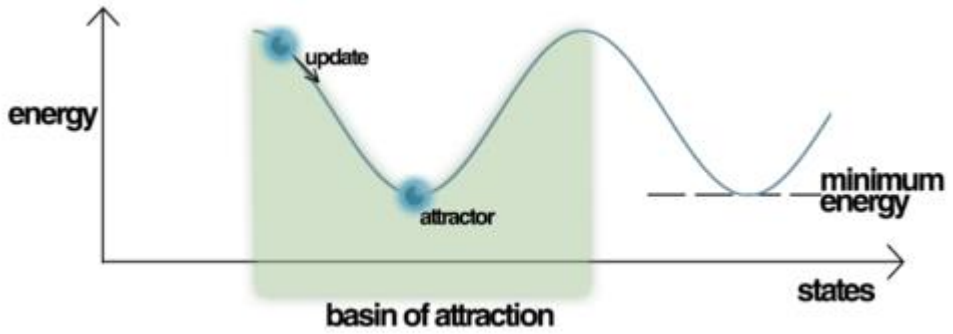


No solution for XOR!
Not universal!
- Minsky and Papert, 1968

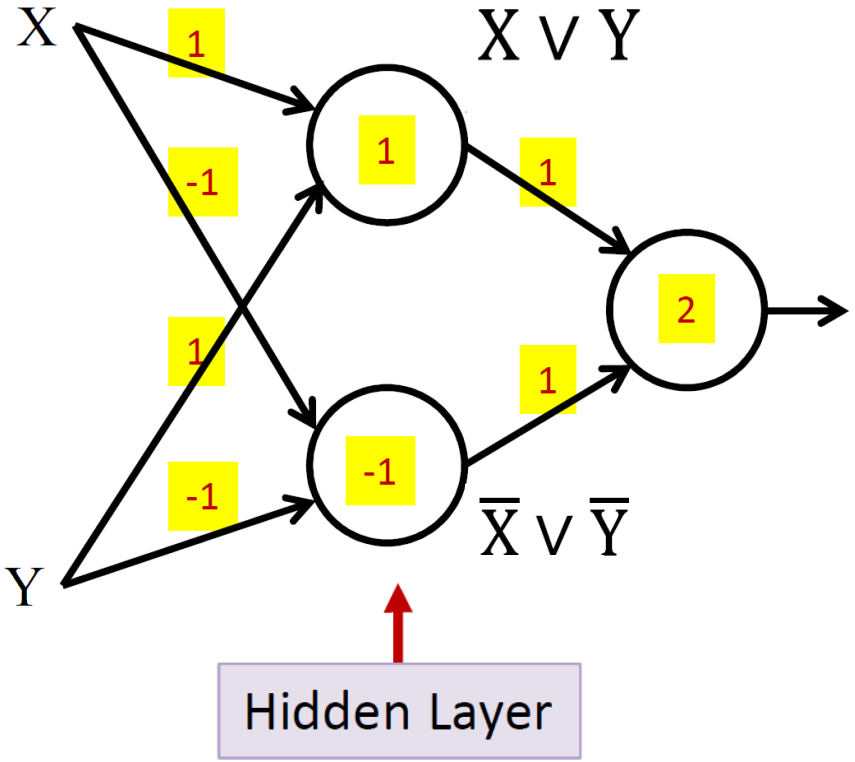
Hopfield network (1982)

- Recurrent neural networks (**connections between nodes can create a cycle**) with dynamical trajectories converging to fixed point attractor states and described by an energy function (there is dynamics)
- Content-addressable ("associative") memory systems
- A model for understanding human memory



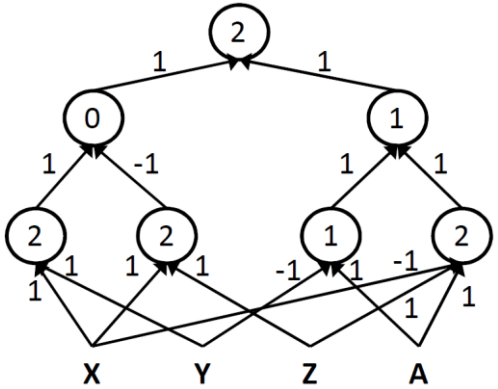


Multi-layer Perceptron!



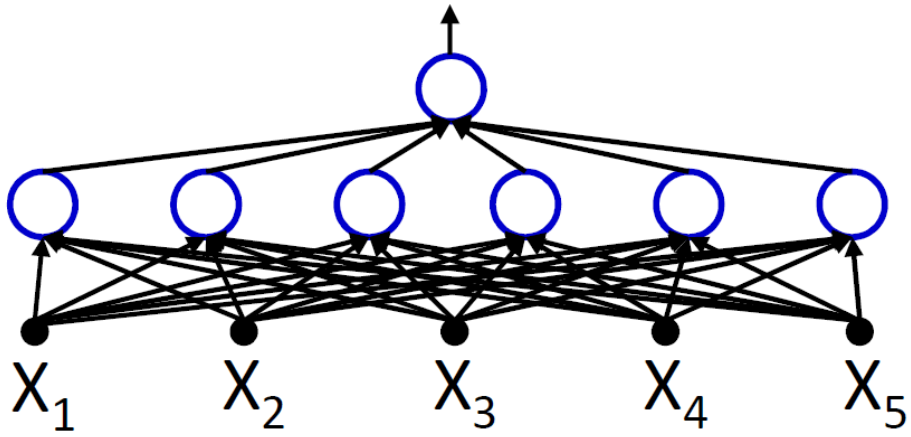
$X \oplus Y$

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



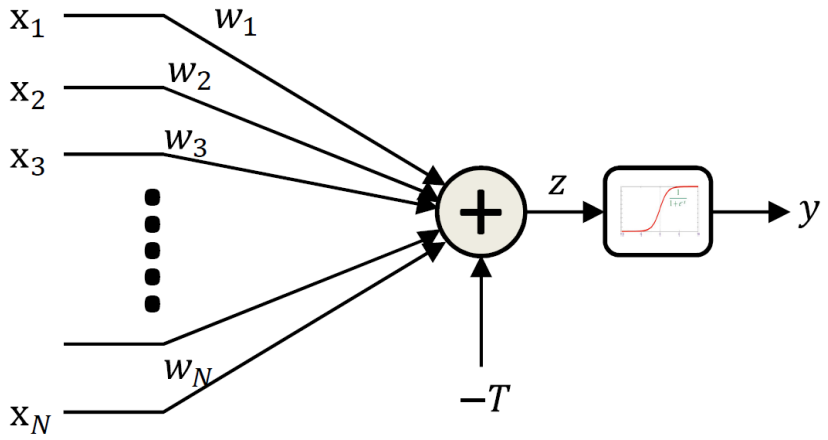
- XOR
  - The first layer is a "hidden" layer
  - Also originally suggested by Minsky and Paper 1968
- Can compose arbitrarily complicated Boolean functions

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
- Having a few extra layers can greatly reduce network size

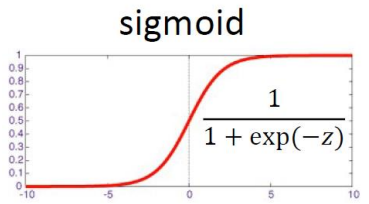
The "soft" perceptron (logistic)

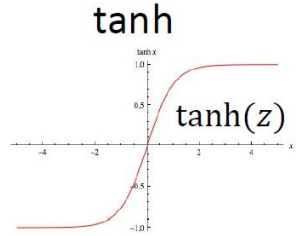


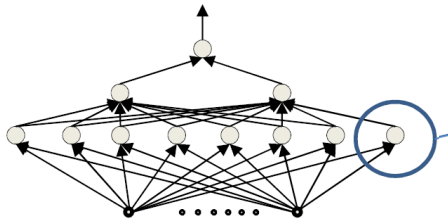
$$z = \sum_i w_i x_i - T$$

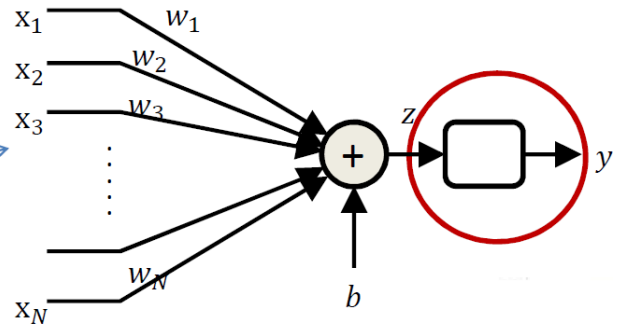
$$y = \frac{1}{1 + exp(-z)}$$

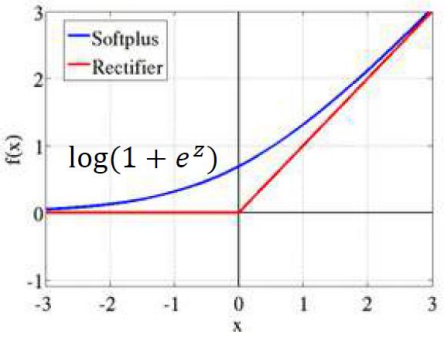
- A "squashing" function instead of a threshold at the output
  - The sigmoid "activation" replaces the threshold
- Activation: The function that acts on the weighted combination of inputs (and threshold)
- Other "activations"



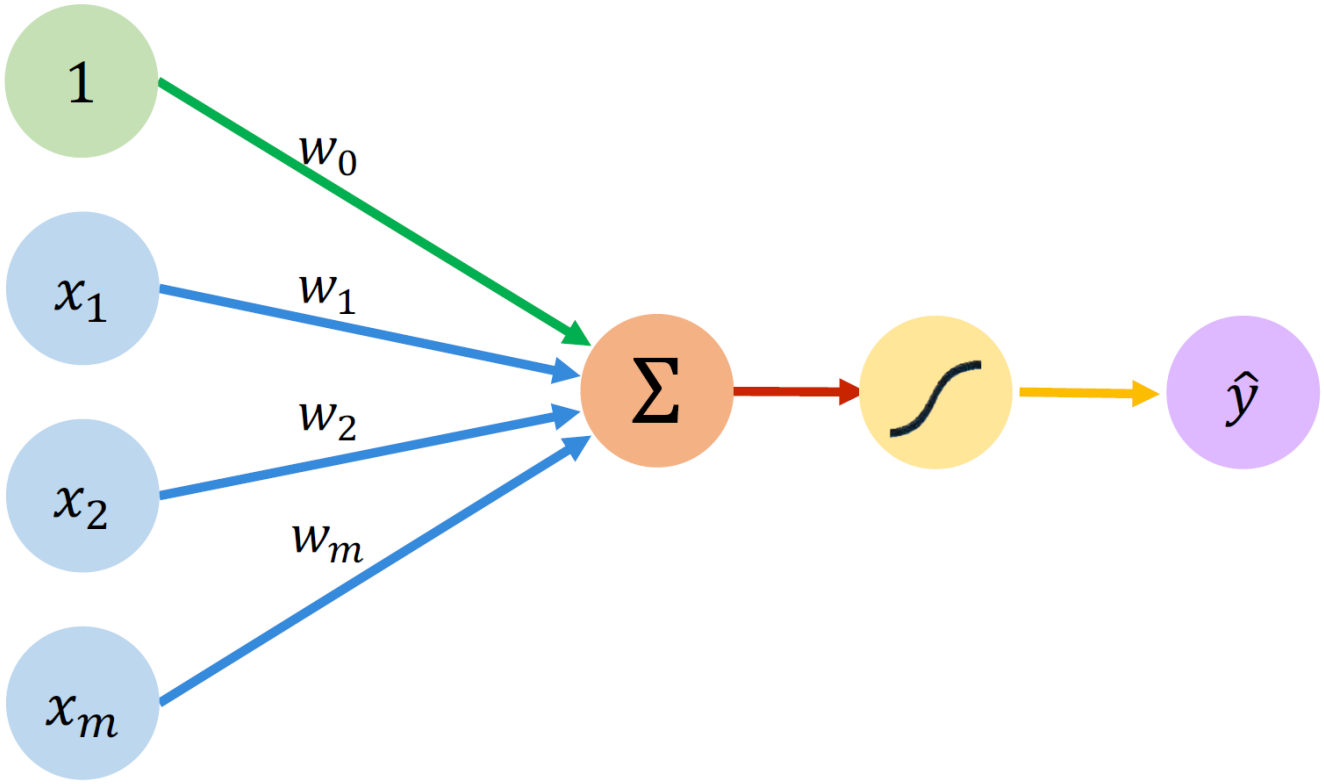








Basic Structure of Neural Nets; The Perceptron



Inputs Weights Sum Non-Linearity Output

Output

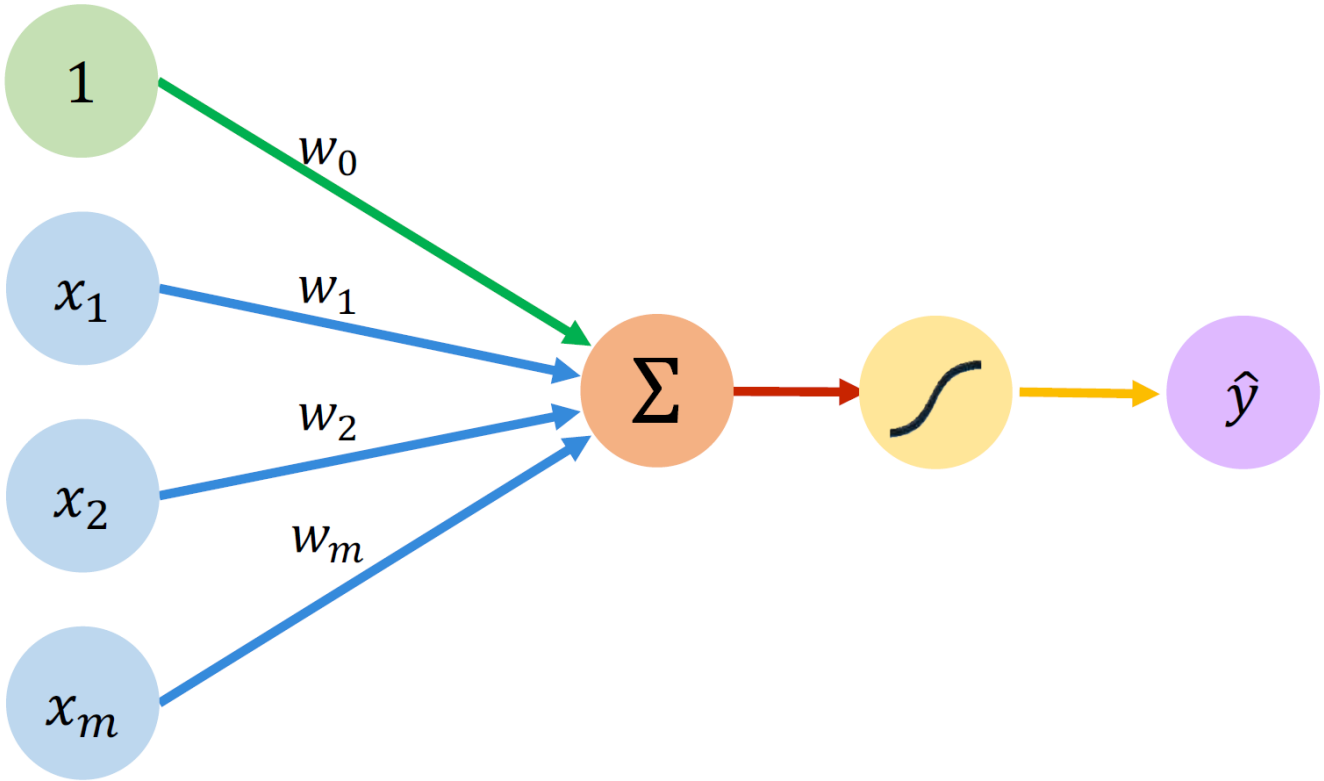
Linear combination of inputs

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i \, w_i\right)$$

Non-linear activation function

Bias

The Perceptron: Forward Propagation



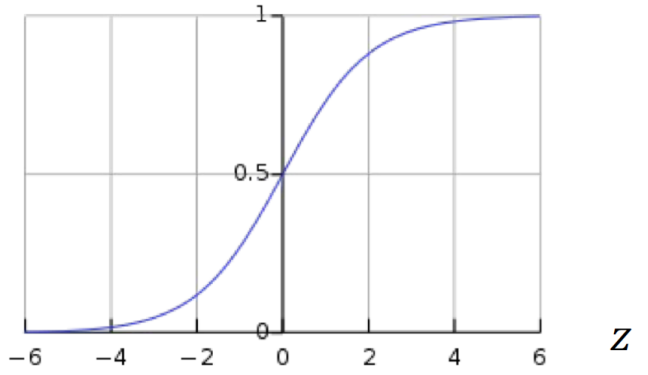
Inputs Weights Sum Non-Linearity Output

Activation Functions

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$

- Example: sigmoid function

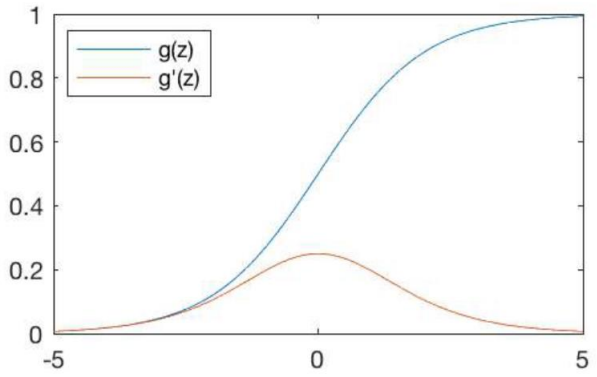
$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



where: $\boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ and $\boldsymbol{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$

Common Activation Functions

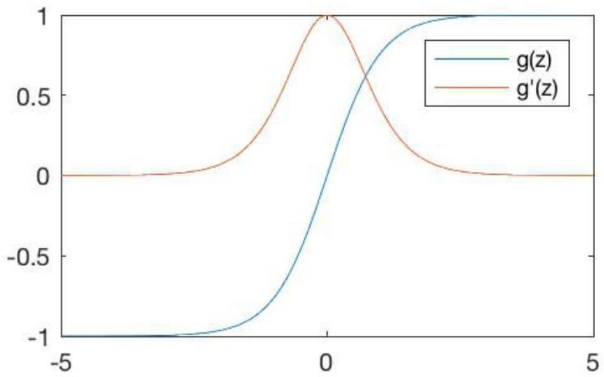
Sigmoid Function



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

$$g'(z) = g(z)(1 - g(z))$$

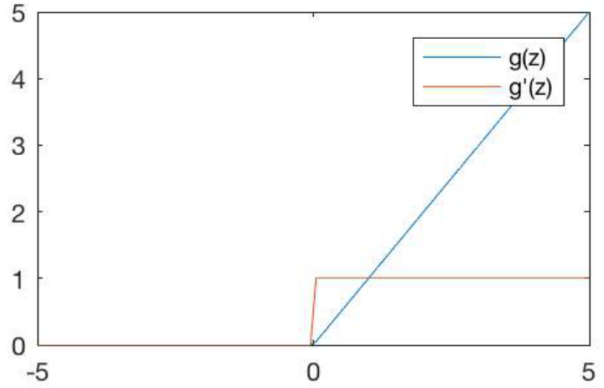
Hyperbolic Tangent



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

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)



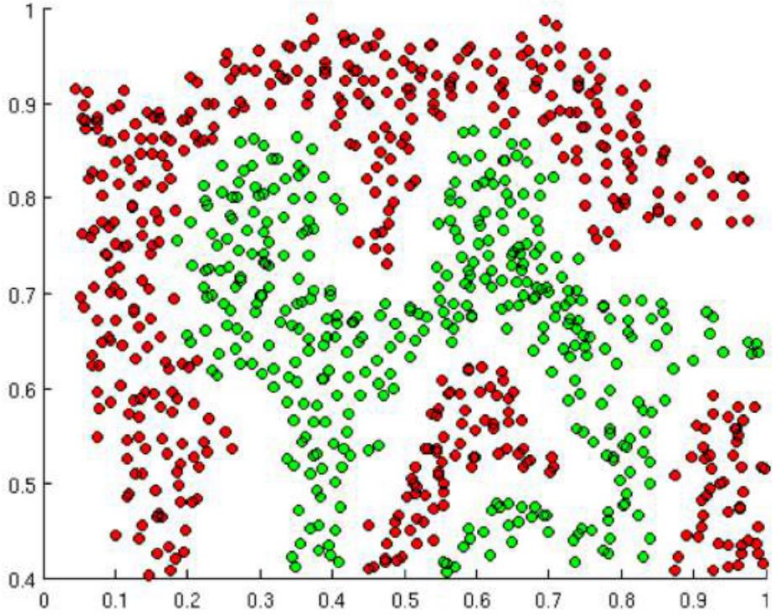
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

NOTE: All activation functions are non-linear

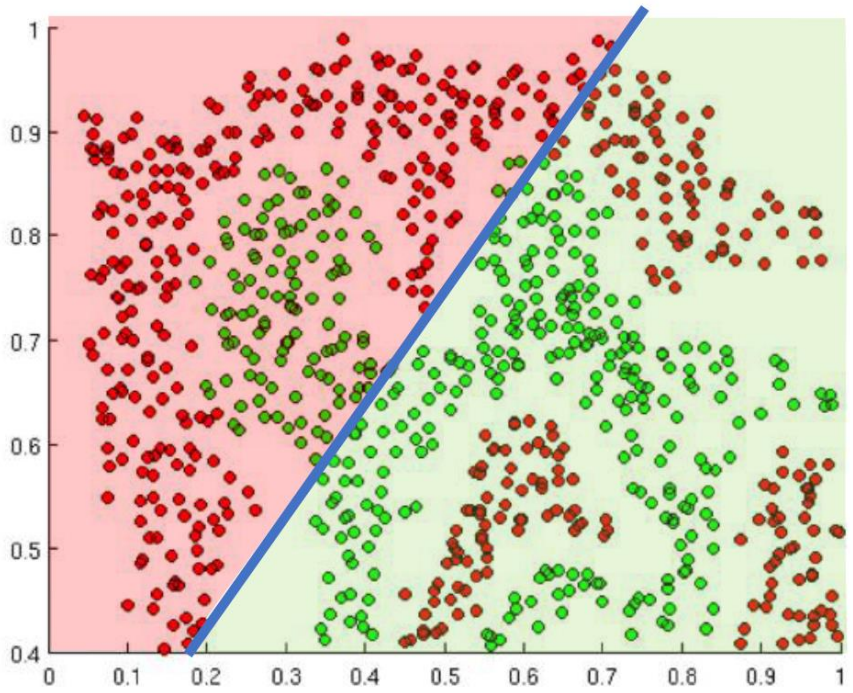
Importance of Activation Functions

The purpose of activation functions is to ***introduce non-linearities*** *into the network*

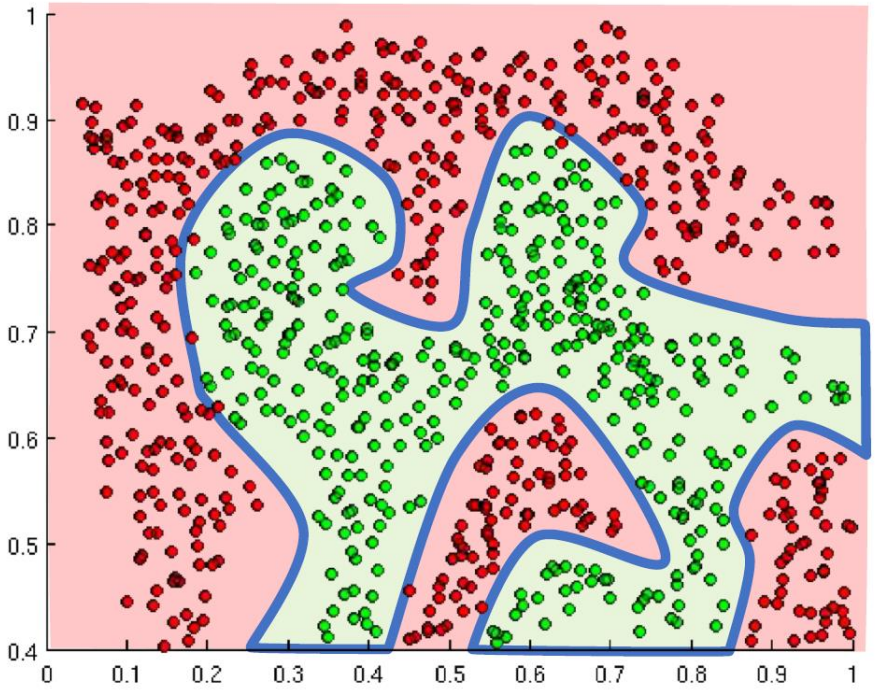


What if we wanted to build a Neural Network to distinguish green vs red points?

Importance of Activation Functions

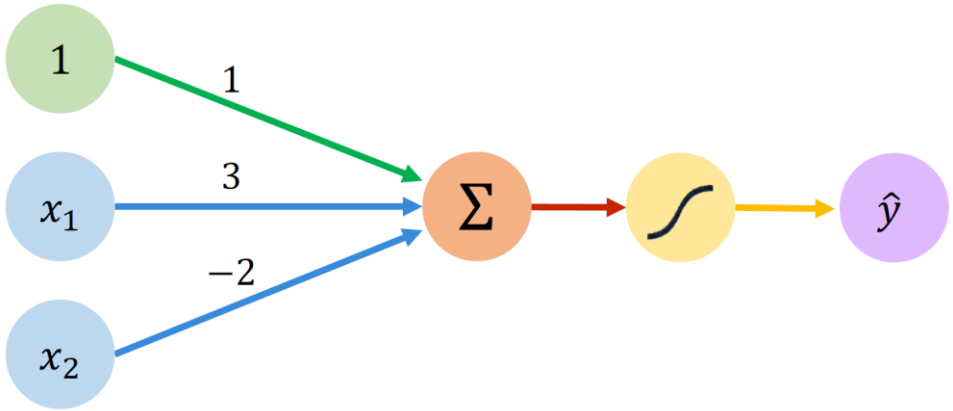


Linear Activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

The Perceptron: Example



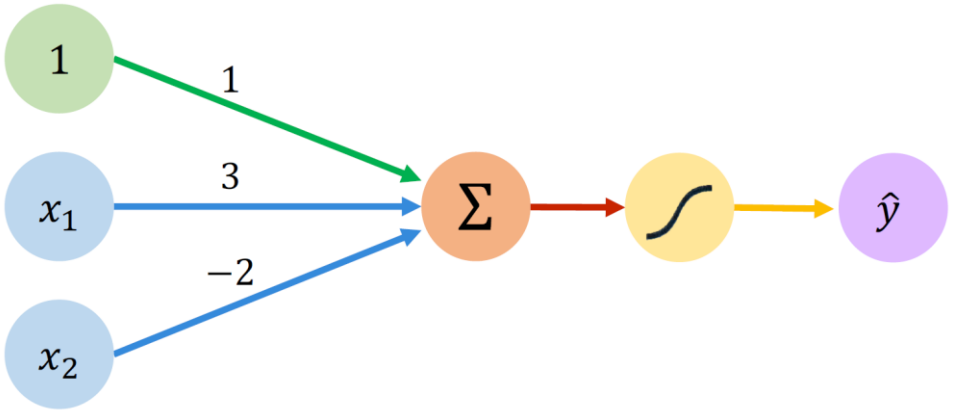
We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$

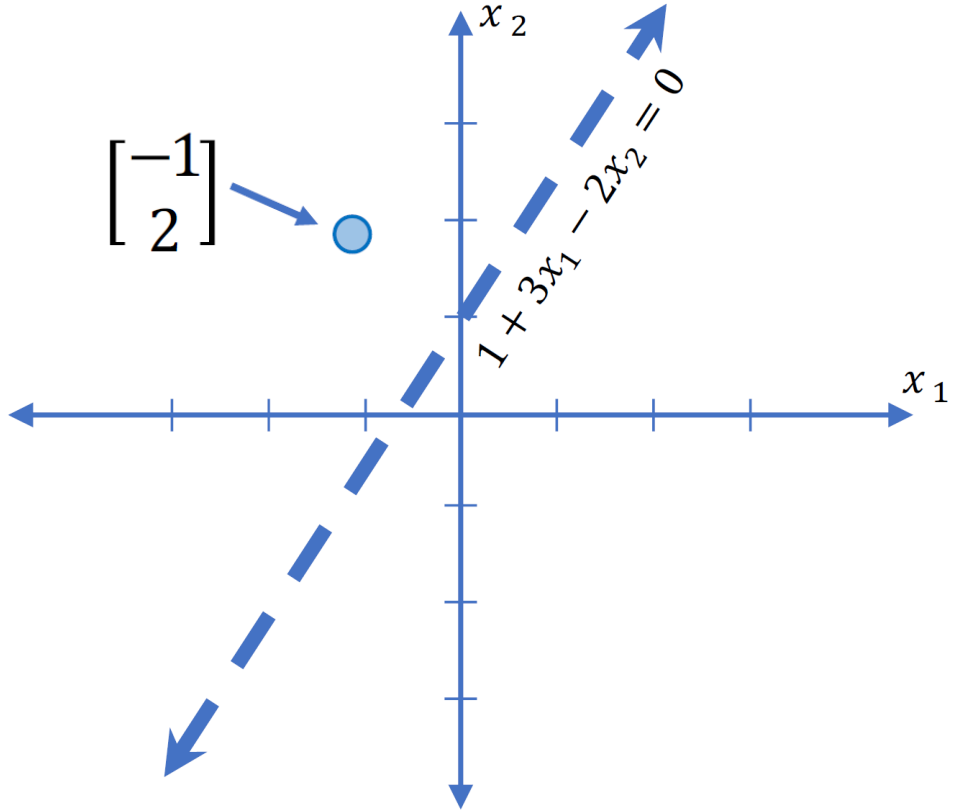
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$

$$\hat{y} = g(\underbrace{1 + 3x_1 - 2x_2}_{\text{This is just a line in 2D!}})$$

The Perceptron: Example



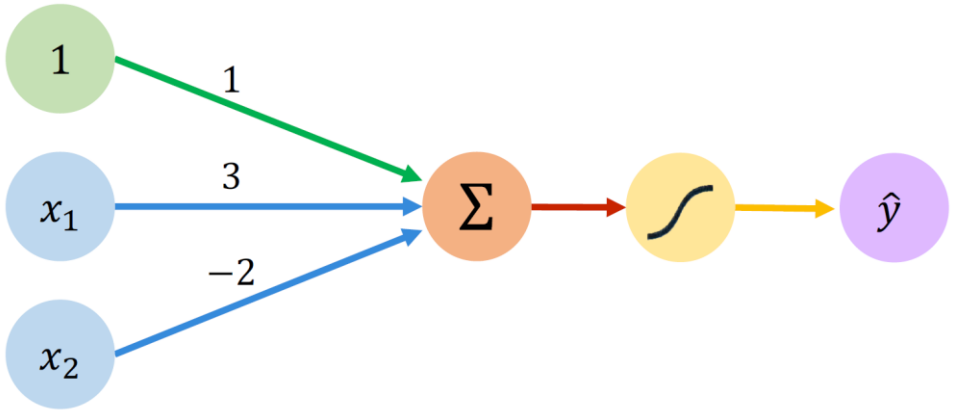
$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



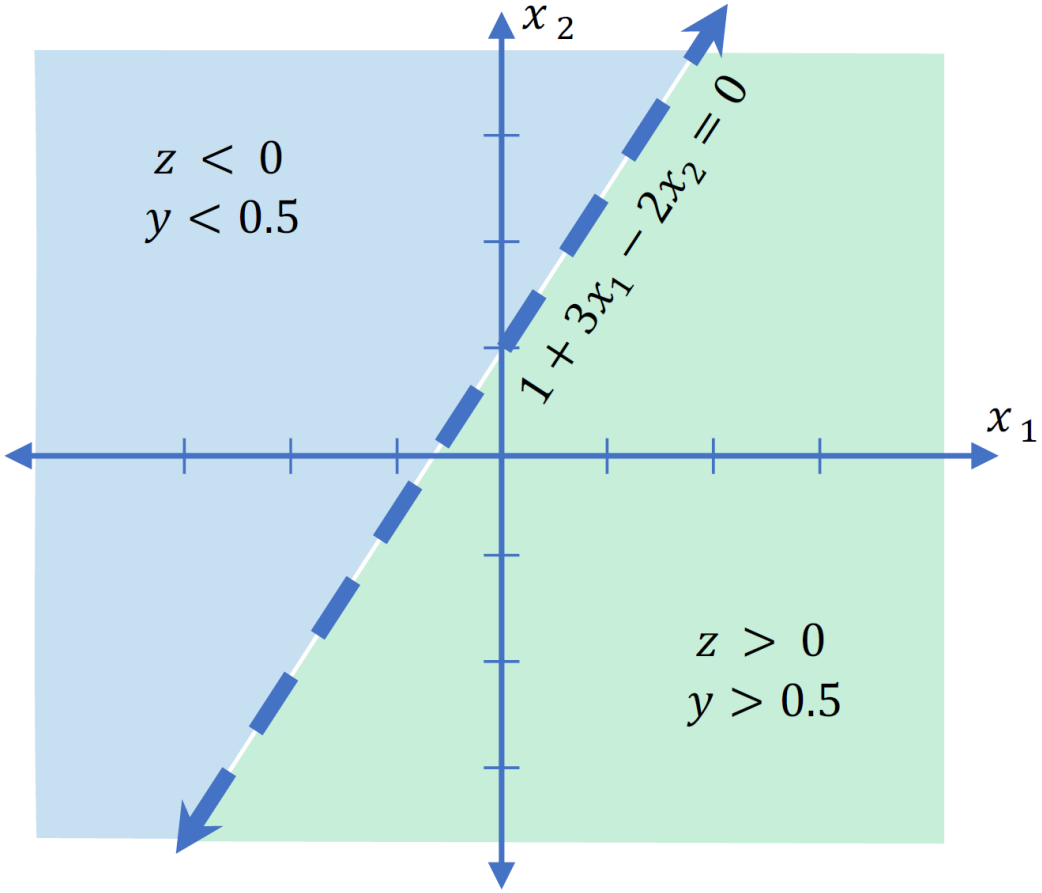
Assume we have input: $\boldsymbol{X} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

$$\hat{y} = g\left(1 + (3 * -1) - (2 * 2)\right)$$
$$= g(-6) \approx 0.002$$

The Perceptron: Example

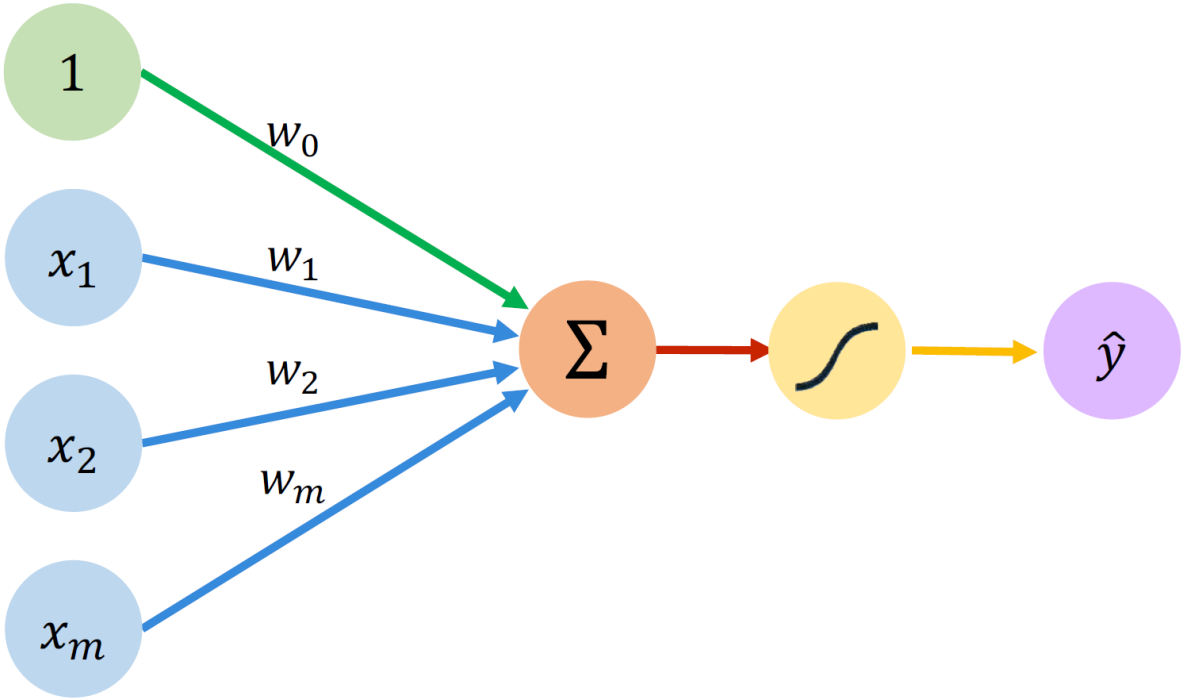


$$\hat{y} = g(1 + 3x_1 - 2x_2)$$

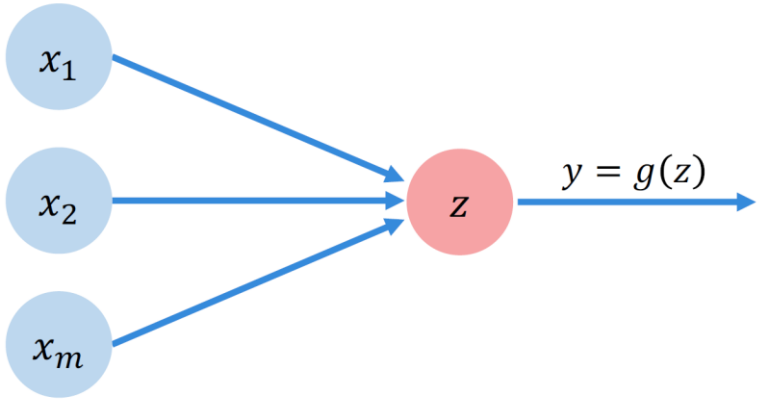


Building Neural Networks with Perceptrons

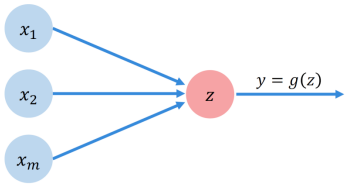
The Perceptron: Simplified



Inputs Weights Sum Non-Linearity Output

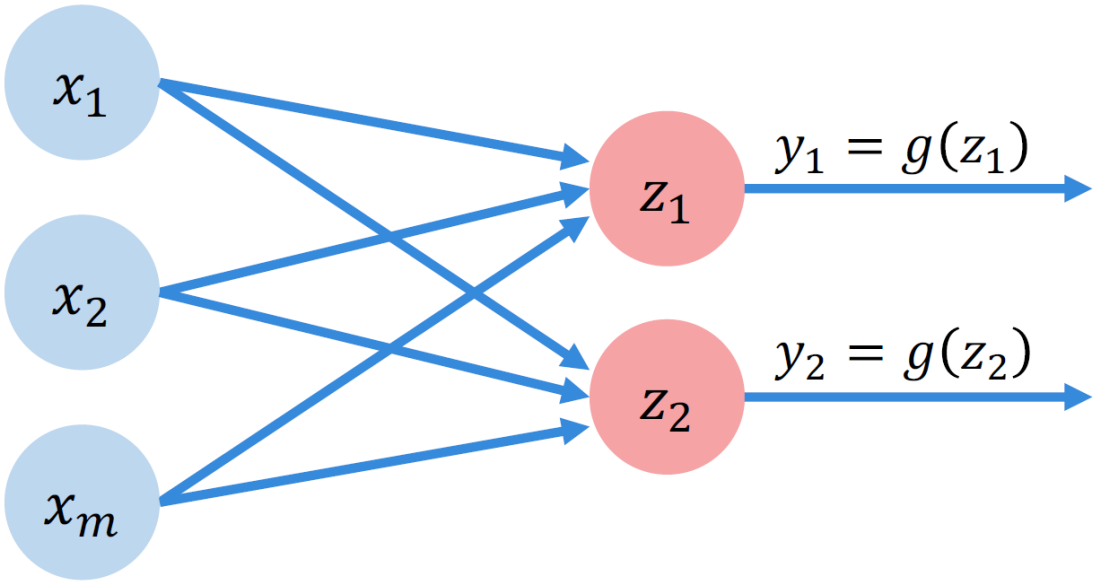


$$z = w_0 + \sum_{j=1}^{m} x_j w_j$$



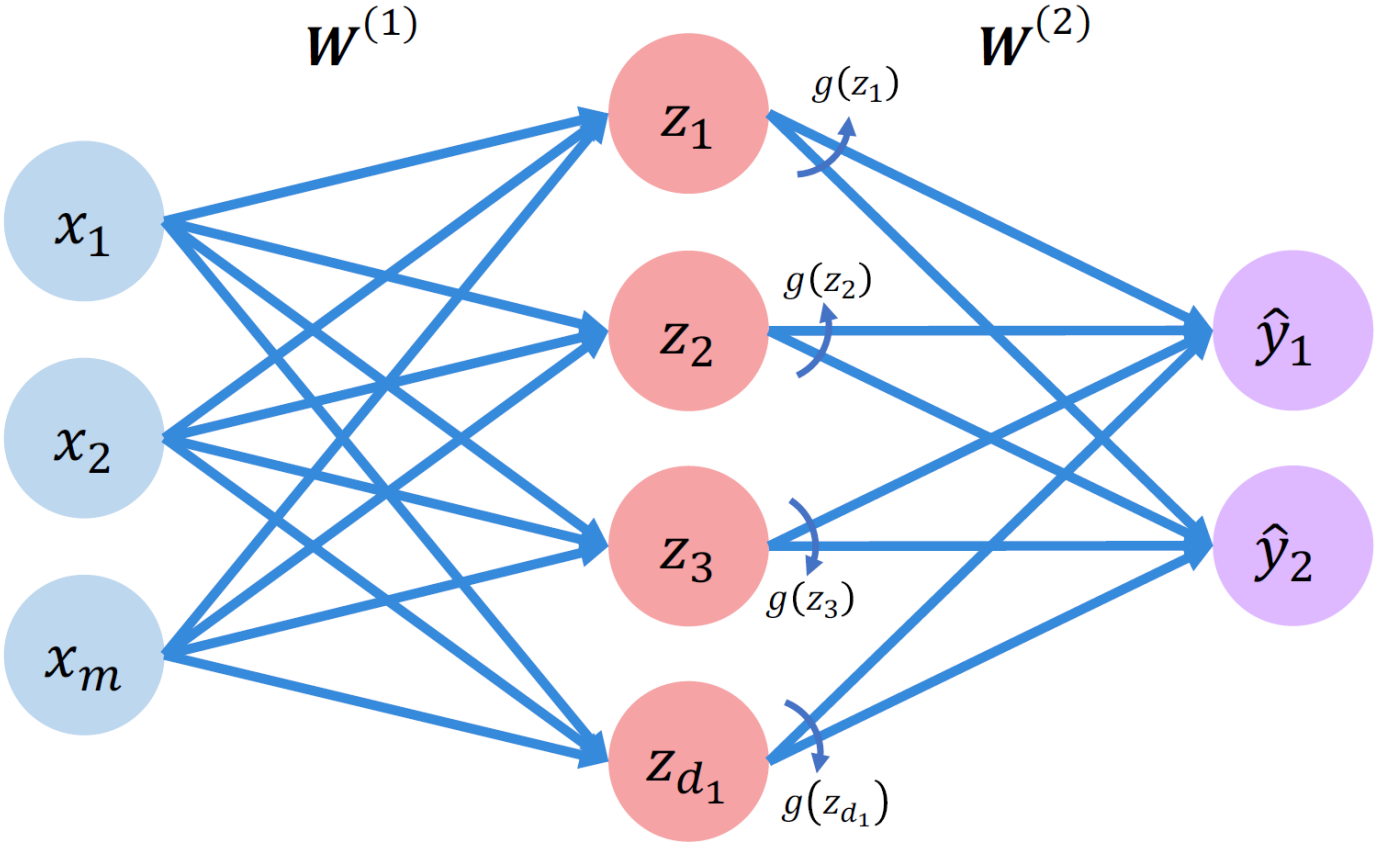
$$z = w_0 + \sum_{j=1}^{m} x_j \, w_j$$

Multi Output Perceptron



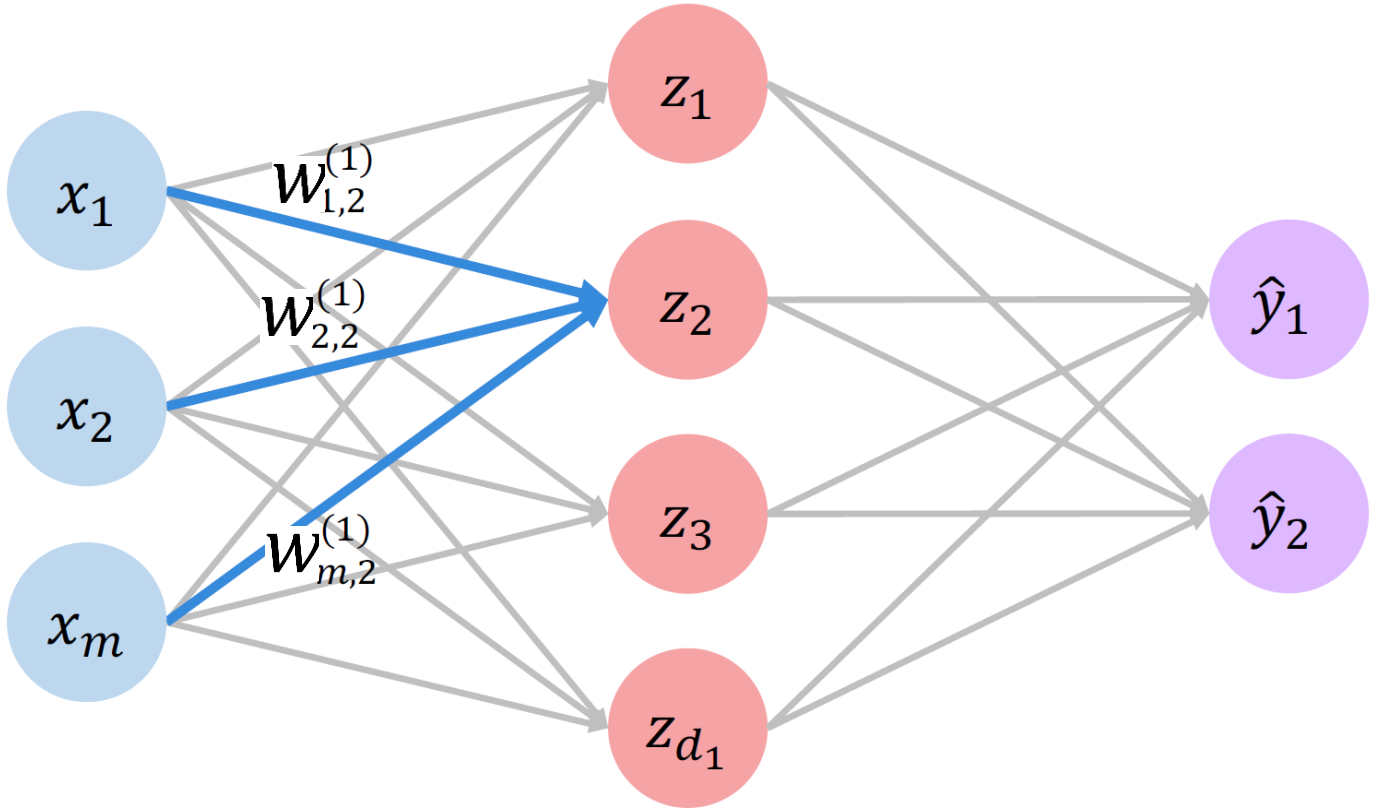
$$z_i = w_{0,i} + \sum_{j=1}^{m} x_j \, w_{j,i}$$

Basic Structure of Neural Nets; Single Layer Neural Network



$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,i}^{(1)} \qquad \hat{y}_i = g\left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} z_j \, w_{j,i}^{(2)} \right)$$

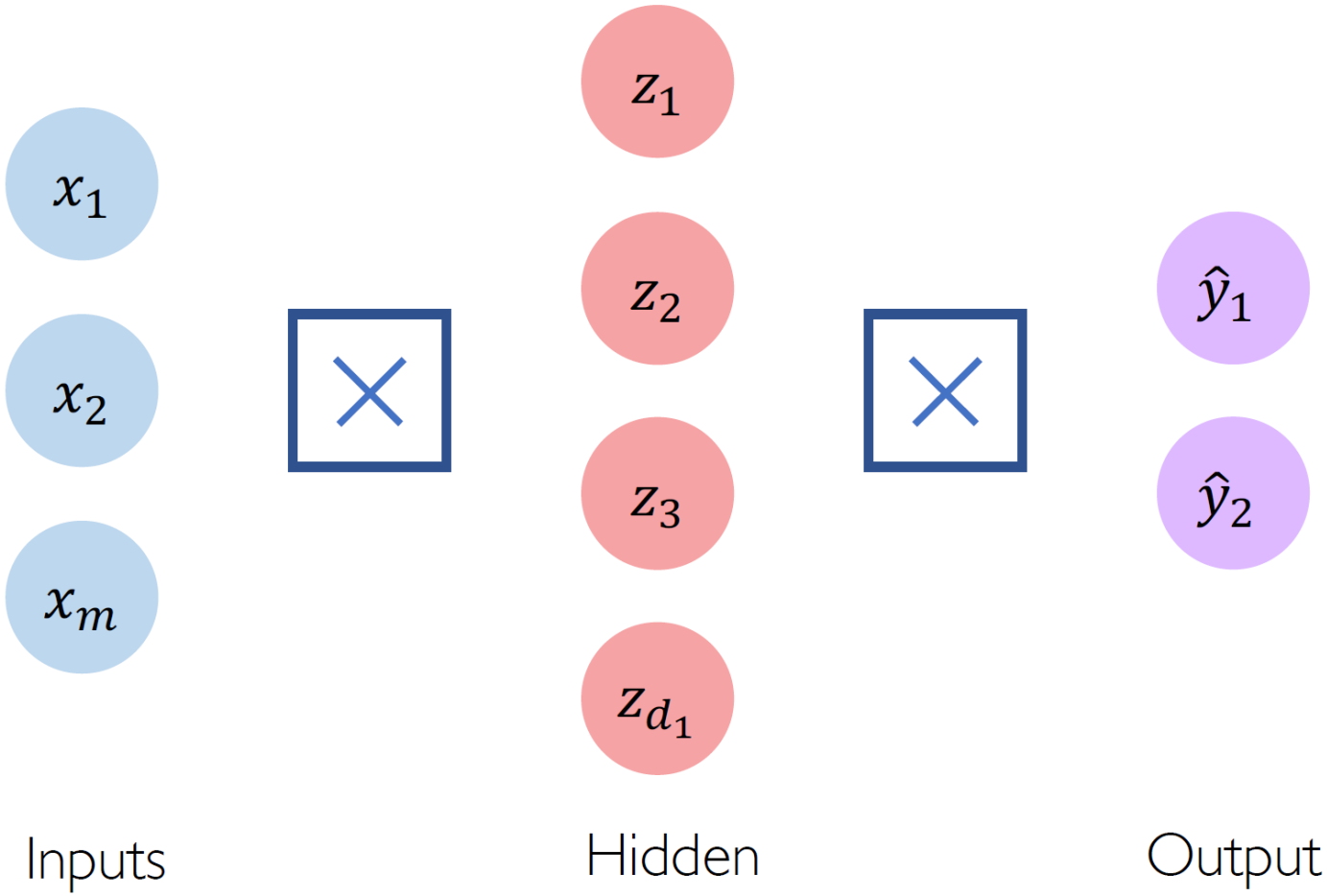
Single Layer Neural Network



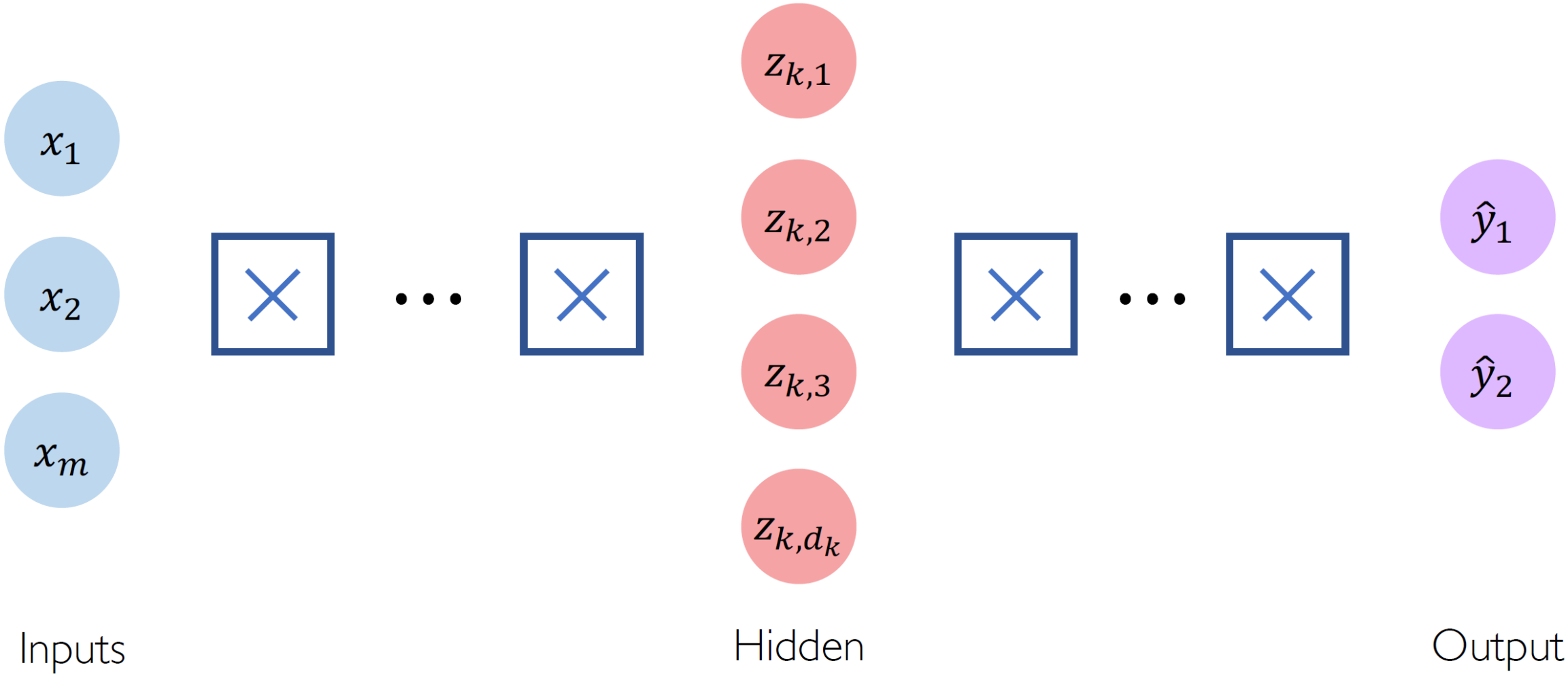
$$z_2 = w_{0,2}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,2}^{(1)}$$

$$= w_{0,2}^{(1)} + x_1 \, w_{1,2}^{(1)} + x_2 \, w_{2,2}^{(1)} + x_m \, w_{m,2}^{(1)}$$

Multi Output Perceptron

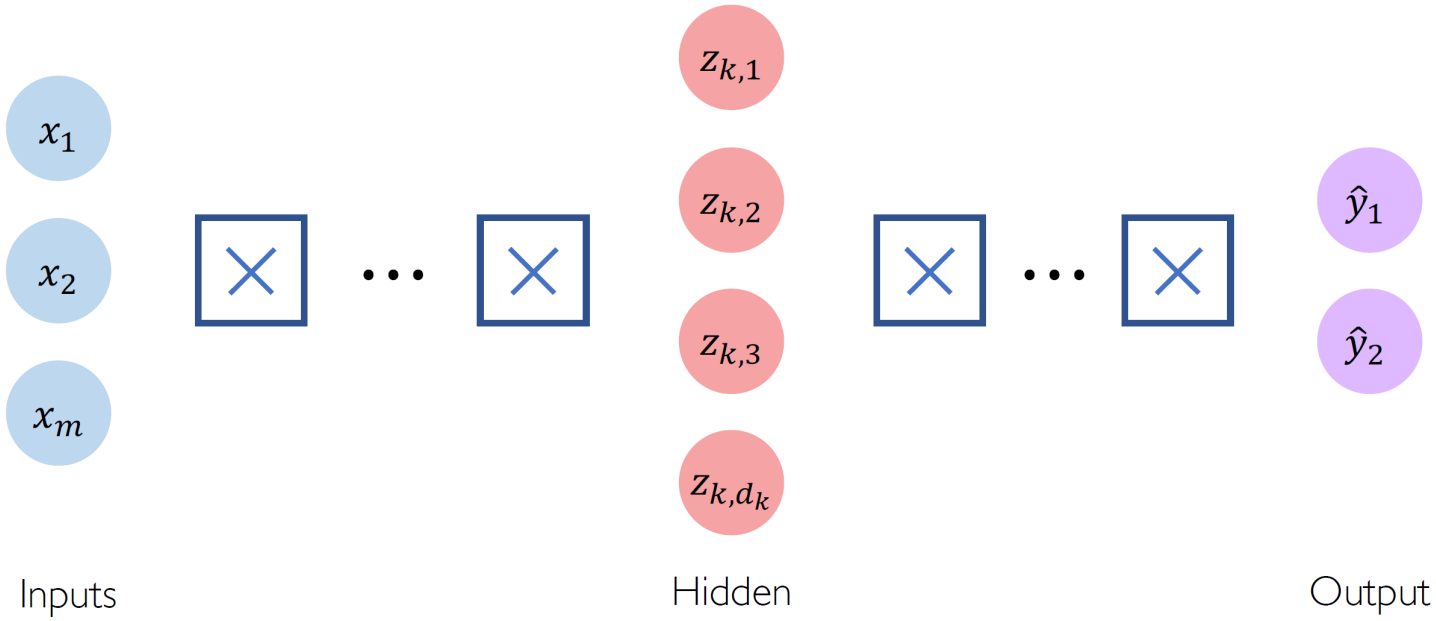


Deep Neural Network



$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{d_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

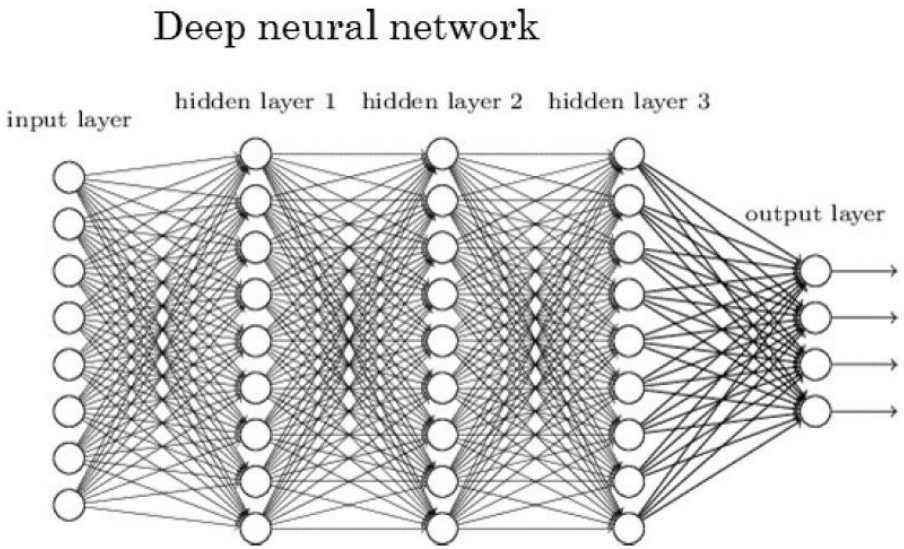
Deep Neural Network

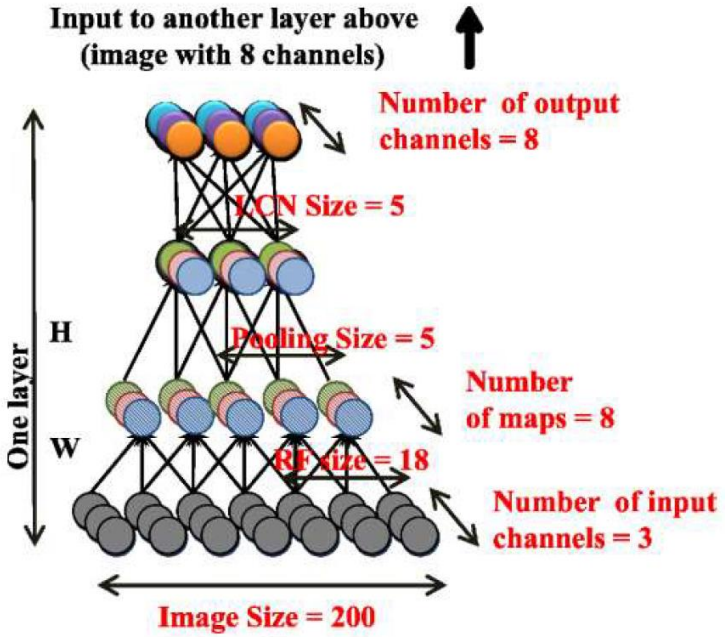


$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{d_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

Deep Structures

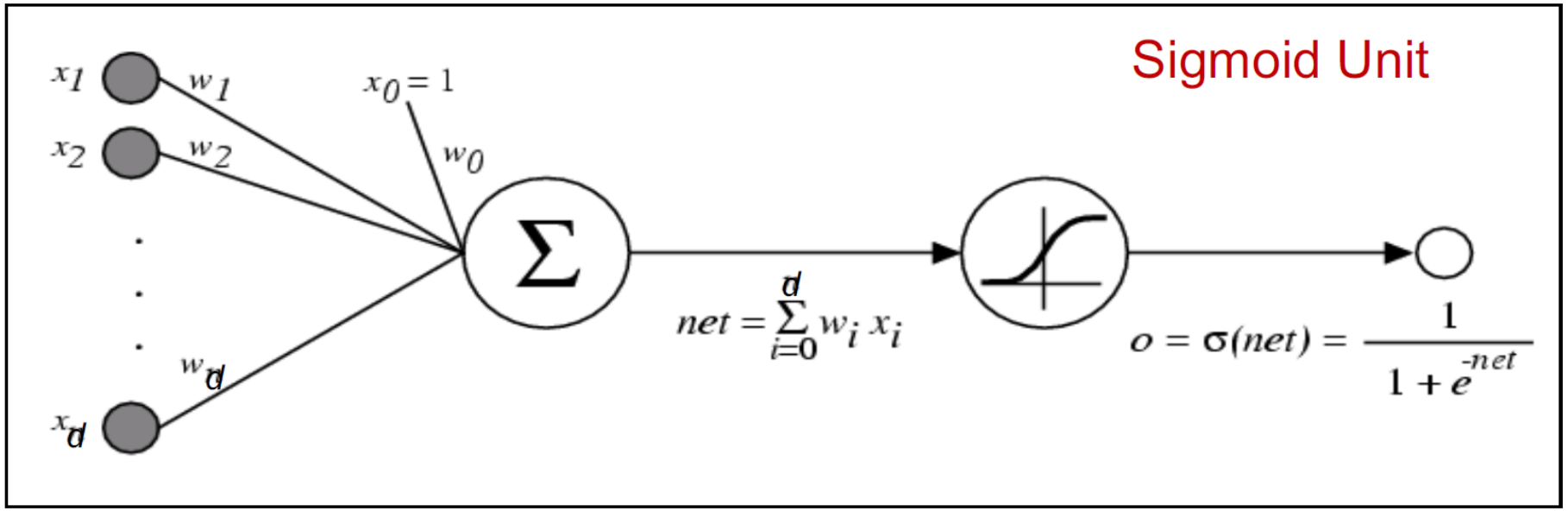
- "depth" is the length of the longest path from a source to a sink
- Layered deep structure
- "Deep" $\mapsto$ Depth > 2



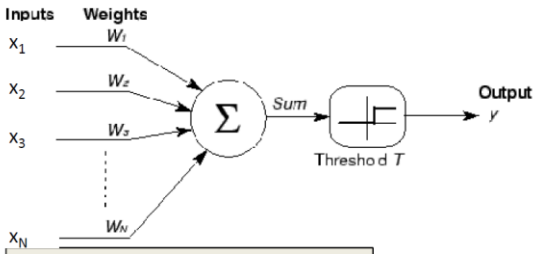


Logistic function as a graph

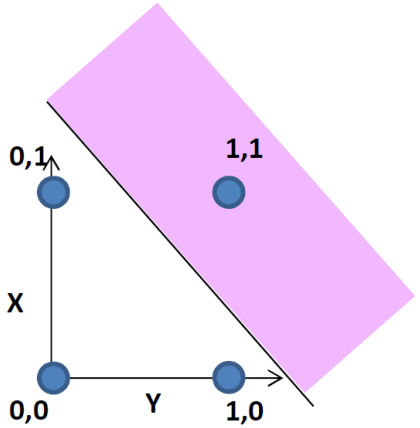
$$\text{Output, } o(\mathbf{x}) = \sigma(w_0 + \sum_i w_i X_i) = \frac{1}{1 + \exp(-(w_0 + \sum_i w_i X_i))}$$

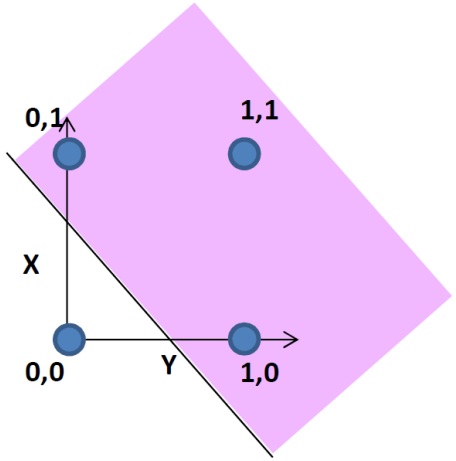


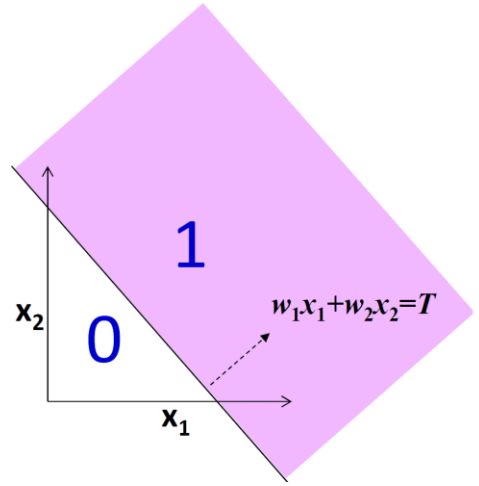
A Perceptron on Reals is Linear classifier

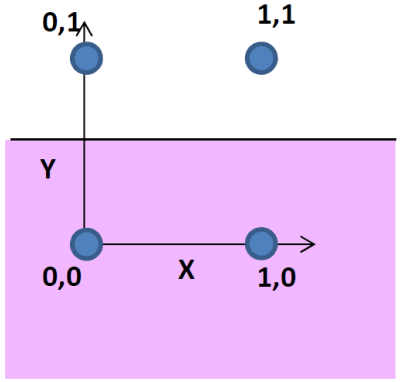


$$y = \begin{cases} 1 \text{ if } \sum_i w_i x_i \geq T \\ 0 \text{ else} \end{cases}$$





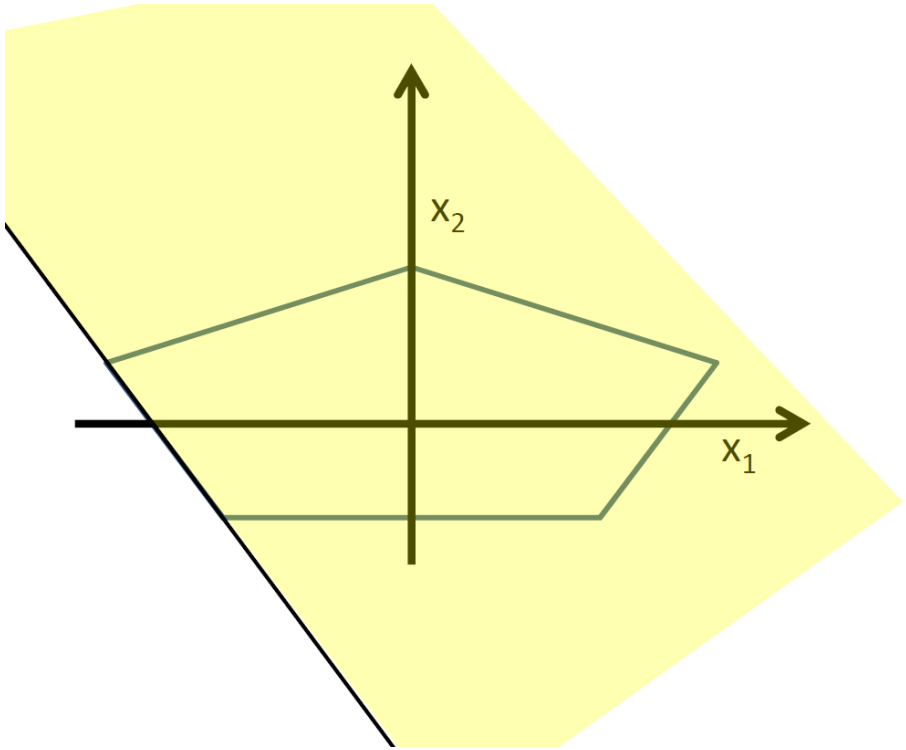


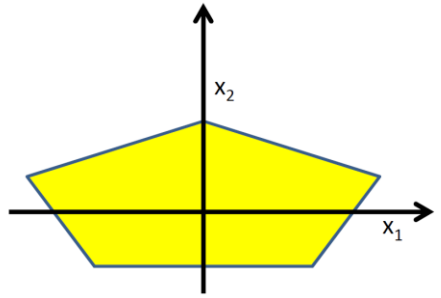


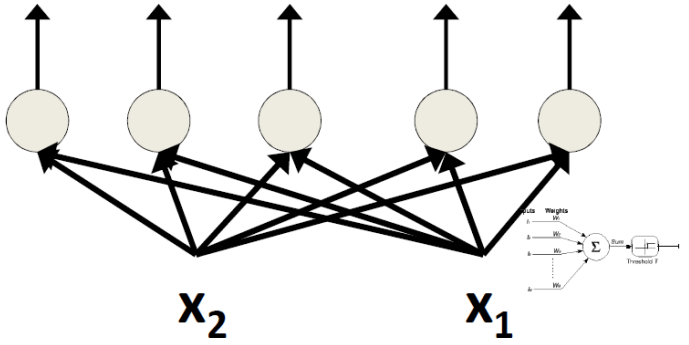
- Boolean perceptrons are also linear classifiers
  - Purple regions have output 1 in the figures
  - What are these functions
  - Why can we not compose an XOR?

Composing complicated "decision" boundaries

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

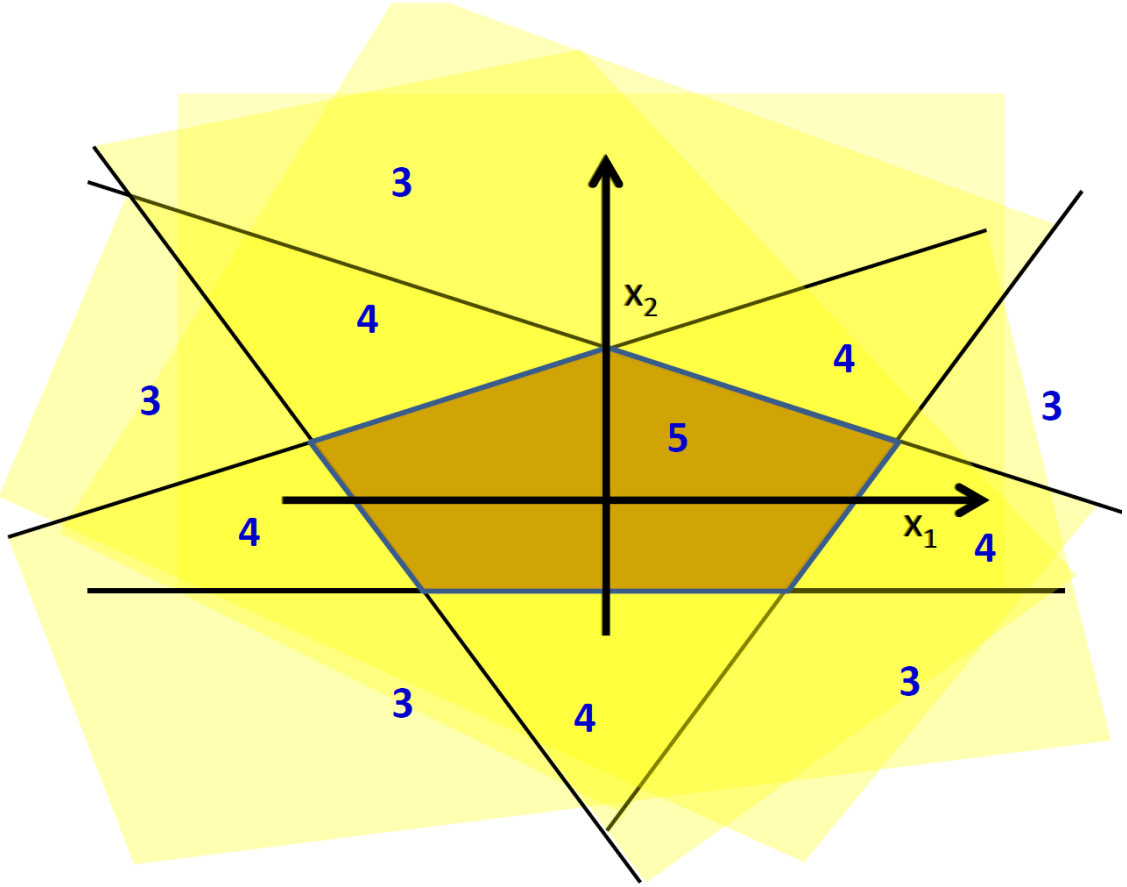


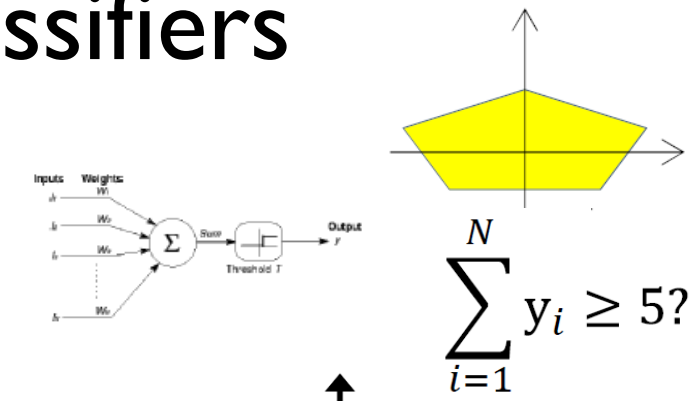




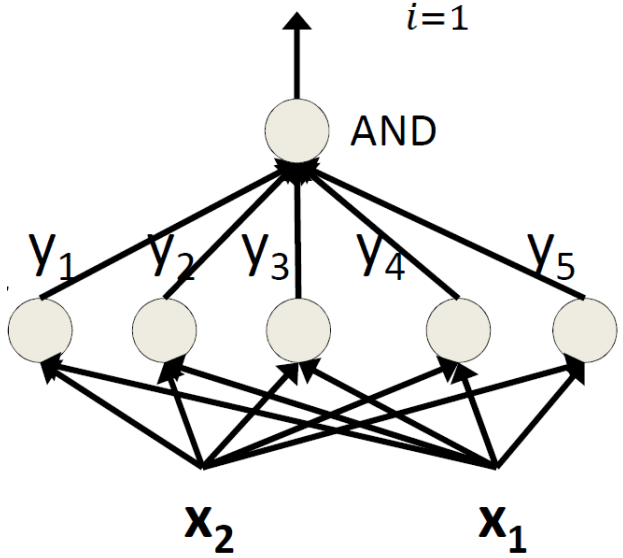
- The network must fire if the input is in the coloured area

Booleans over the reals; MLPs as universal classifiers



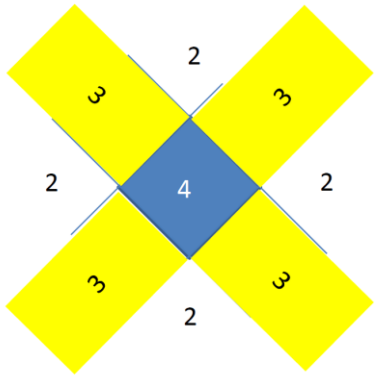


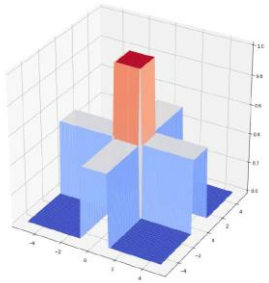
$$\sum_{i=1}^{N} y_i \geq 5?$$

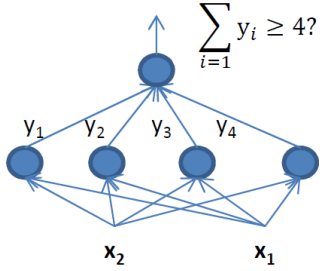


The network must fire if the input is in the coloured area

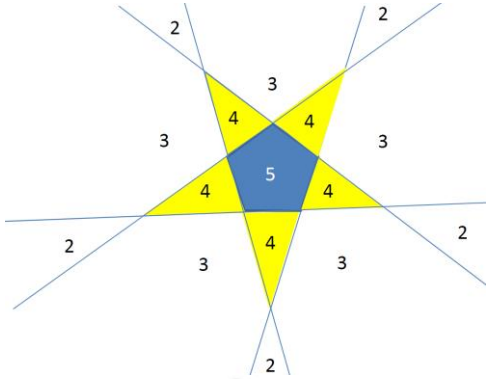
Composing a Square decision boundary

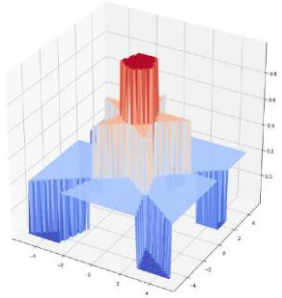


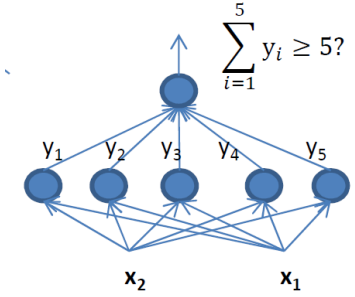




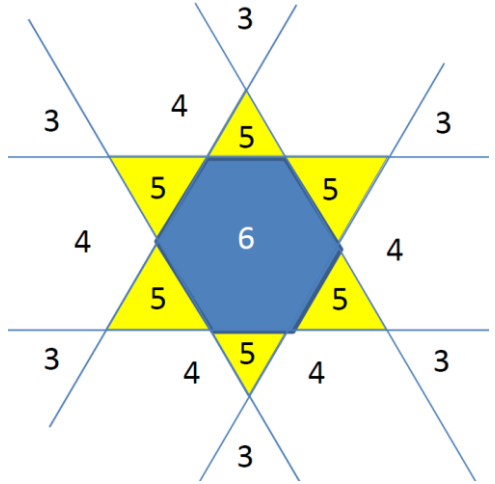
Composing a pentagon

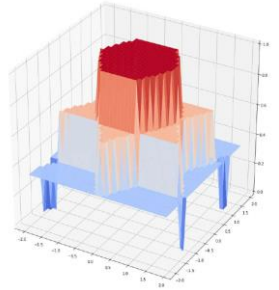


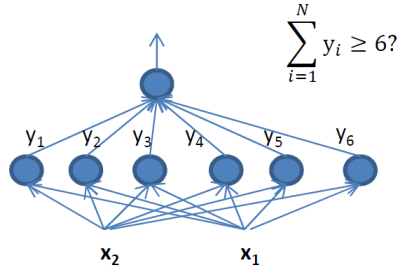




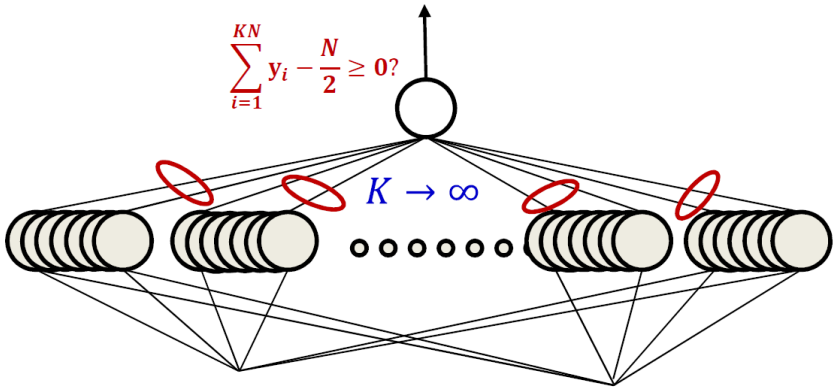
Composing a hexagon

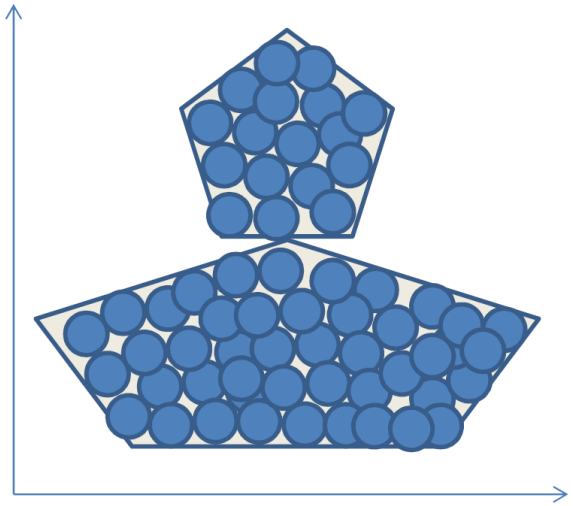






Composing an arbitrary figure



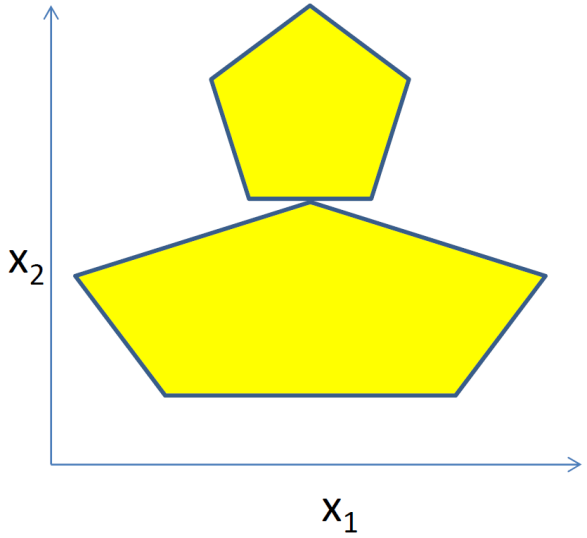


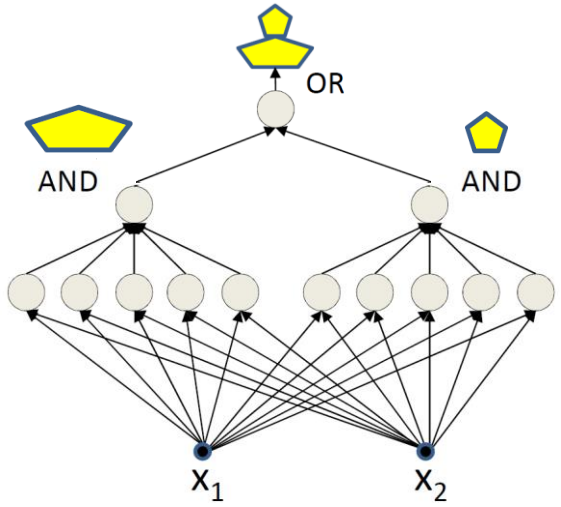
- Just fit in an arbitrary number of circles
  - More accurate approximation with greater number of smaller circles
  - Can achieve arbitrary precision

MLP: Universal classifier

More complex decision boundaries

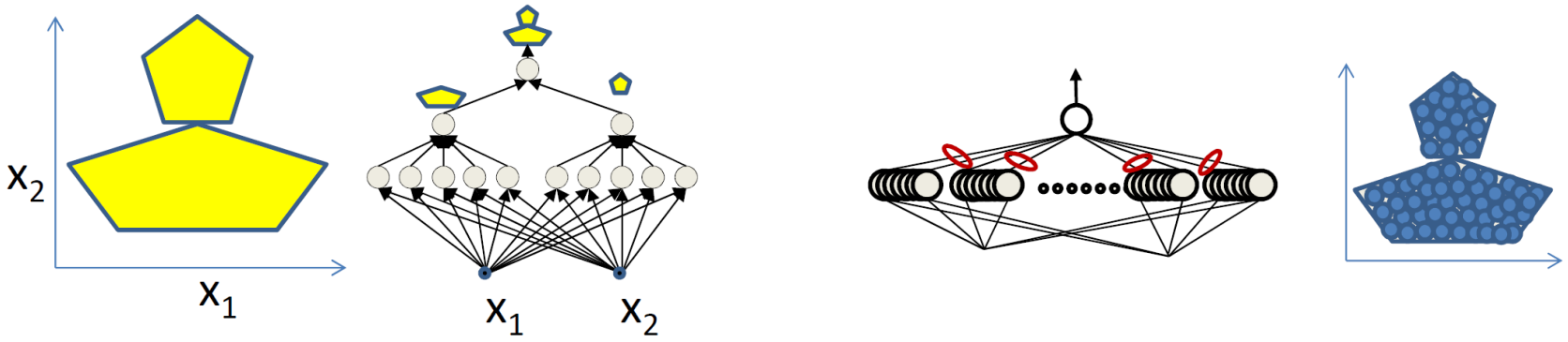
The need for depth





- Network to fire if the input is in the yellow area
  - "OR" two polygons
  - A third layer is required
- Can compose very complex decision boundaries
- Classification problems: **finding** decision boundaries in high-dimensional space
  - Can be performed by an MLP
- MLPs can classify real-valued inputs

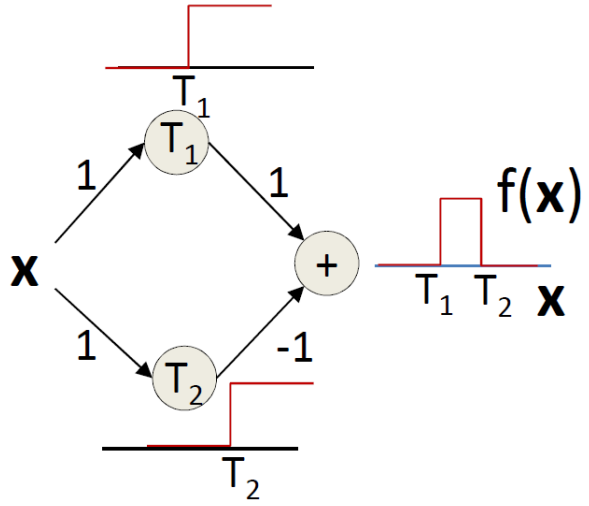
Depth vs. width

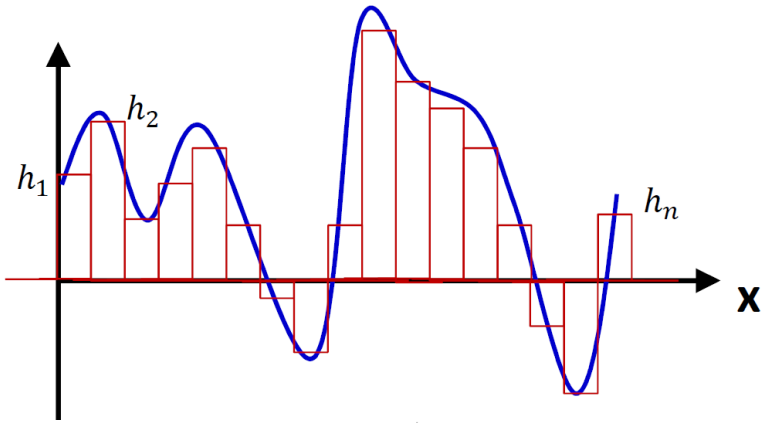


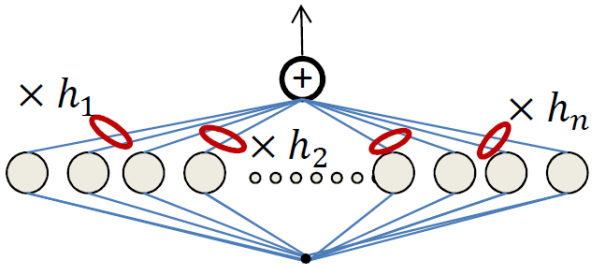
- Deeper networks can require far fewer neurons
- A polynomial of degree n requires a network of depth $\log_2(n)$
- The number of neurons required in a shallow network is potentially exponential in the dimensionality of the input
  - (this is the worst case)

MLP as a continuous-valued regression; MLPs as universal approxi

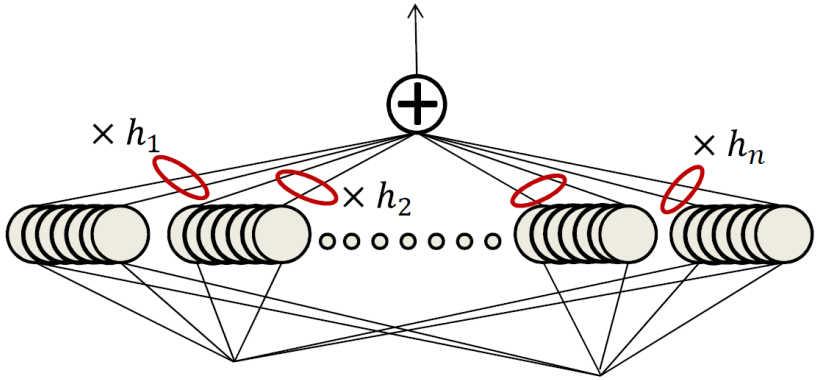
- A simple 3-unit MLP with a "summing" output unit can generate a "square pulse" over an input
  - Output is 1 only if the input lies between T1 and T2
  - T1 and T2 can be arbitrarily specified
- An MLP with many units can model an arbitrary function over an input
  - To arbitrary precision
- Simply make the individual pulses narrower
- This generalizes to functions of any number of inputs

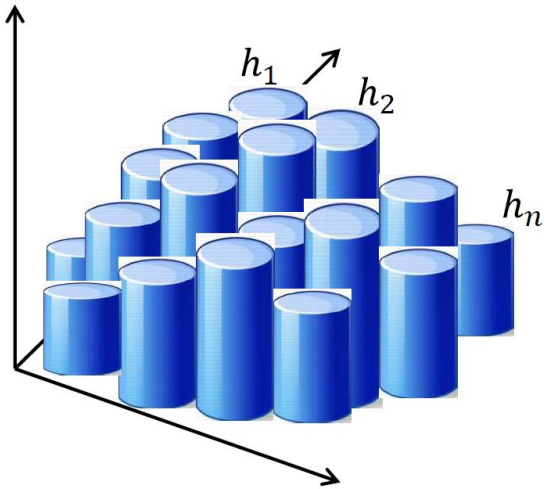






MLP as a continuous-valued function

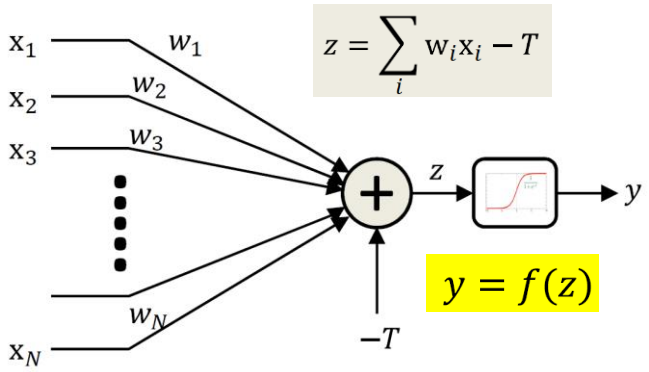


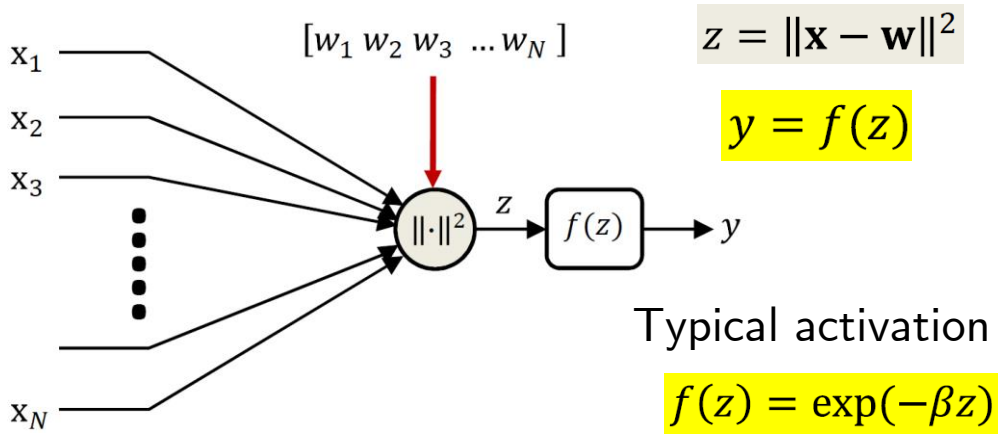


- MLPs can actually compose arbitrary functions in any number of dimensions!
  - Even with only one layer
- As sums of scaled and shifted cylinders
  - To arbitrary precision
- By making the cylinders thinner
  - The MLP is a universal approximator!

Brief segue: RBF networks

Perceptrons so far

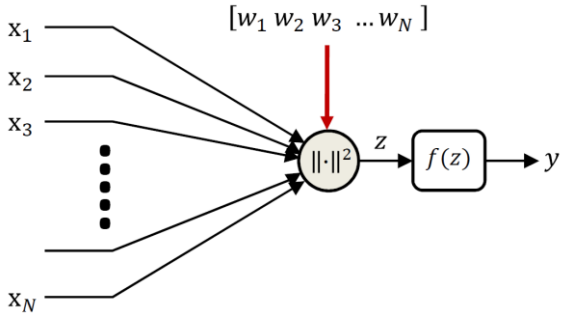


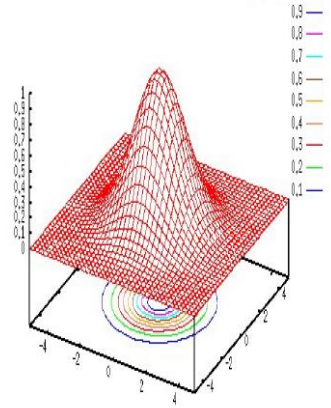


- The output is a **function of the distance** of the input from a "center"
  - The "center" is the parameter specifying the unit
  - The most common activation is the **exponent**
- β is a "**bandwidth**" parameter
  - But other similar activations may also be used
- Key aspect is **radial symmetry**, instead of linear symmetry

RBF networks as universal approximators

- Radial basis functions can compose cylinder-like outputs with just a single unit with appropriate choice of **bandwidth** (or activation function)
  - As opposed to $\mathbf{N} \rightarrow \infty$ units for the linear perceptron





- RBF networks are more effective approximators of continuous-valued functions
  - A one-hidden-layer net only requires one unit per "cylinder"