



Machine Learning and Big Data - DATA622

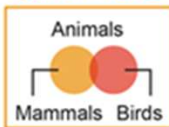
CUNY School of Professional Studies



The 5 tribes of machines learning

5 Tribes of Machine Learning

Symbolists



Use symbols, rules, and logic to represent knowledge and draw logical inference

Favored algorithm

Rules and decision trees

Bayesians

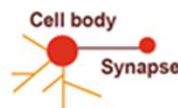


Assess the likelihood of occurrence for probabilistic inference

Favored algorithm

Naïve Bayes or Markov

Connectionists



Recognize and generalize patterns dynamically with matrices of probabilistic, weighted neurons

Favored algorithm

Neural network

Evolutionaries



Generate variations and then assess the fitness of each for a given purpose

Favored algorithm

Genetic programs

Analizers



Optimize a function in light of constraints (“going as high as you can while staying on the road”)

Favored algorithm

Support vectors

<https://learning.acm.org/techtalks/machinelearning>

How it started: Hubel & Wiesel

Mapping of the Visual cortex hierarchy

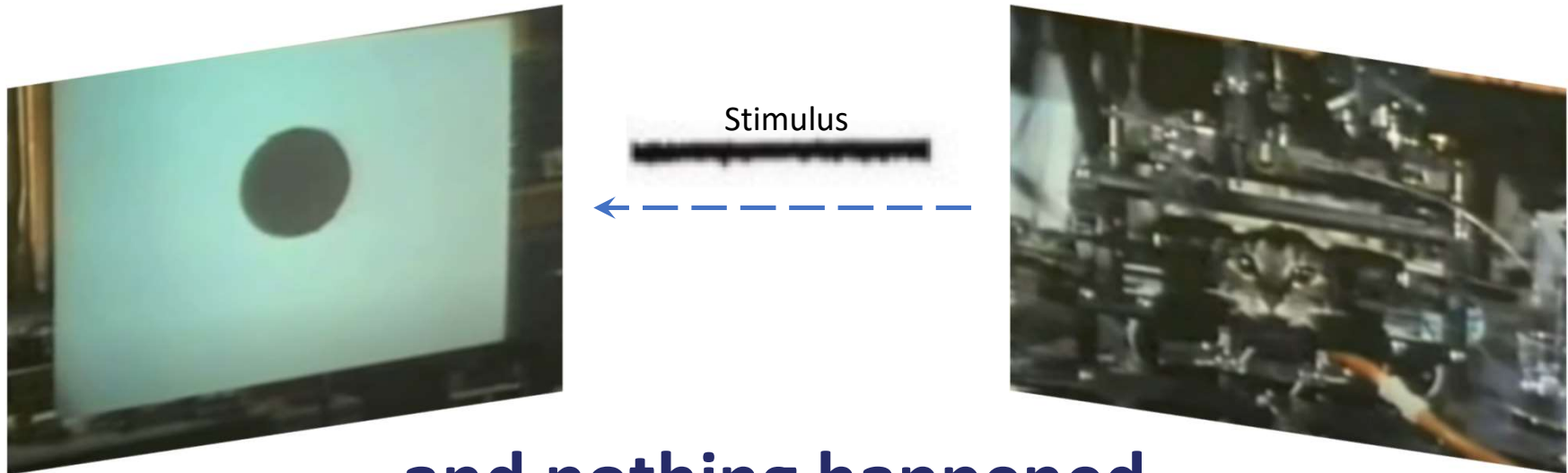
Hubel & Wiesel

David Hubel and Torsten Wiesel recorded electrical activity from individual neurons in the brains of cats. They won the Nobel Prize for Physiology & Medicine in 1981.



Visual cortex experiment

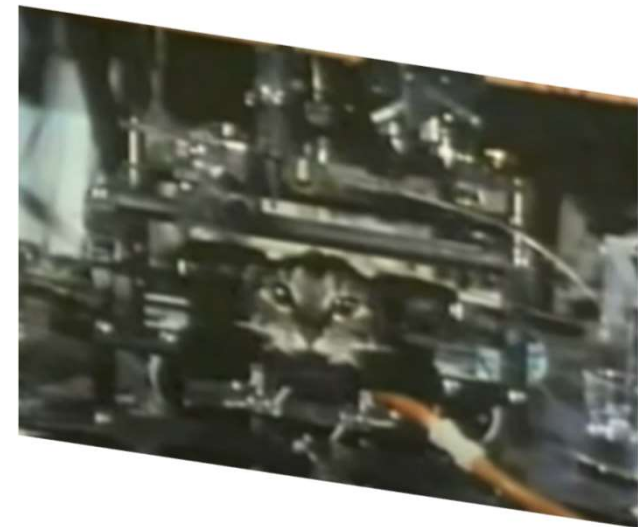
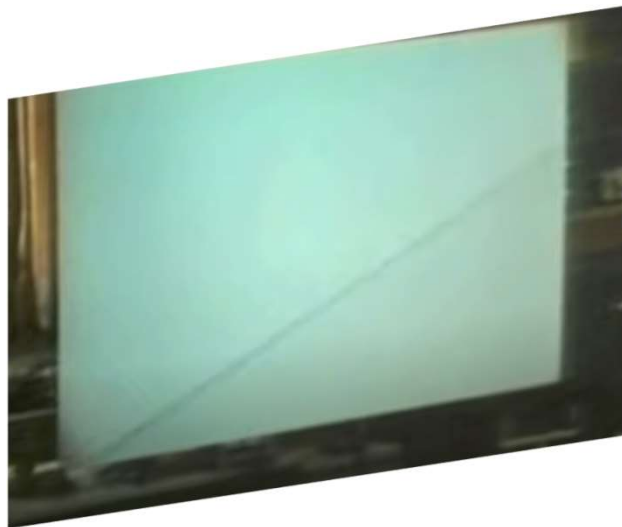
The two jabbed a small microelectrode into a brain cell in the visual cortex at the back of the brain of an anesthetized cat, and showed the cat various shapes.



...and nothing happened

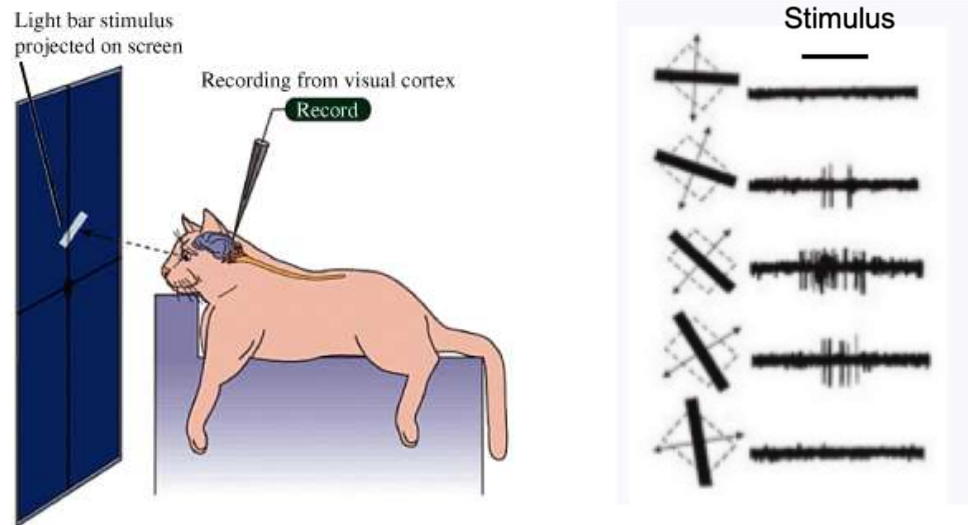
Visual cortex experiment (cont'd)

...until they changed slides in a certain way

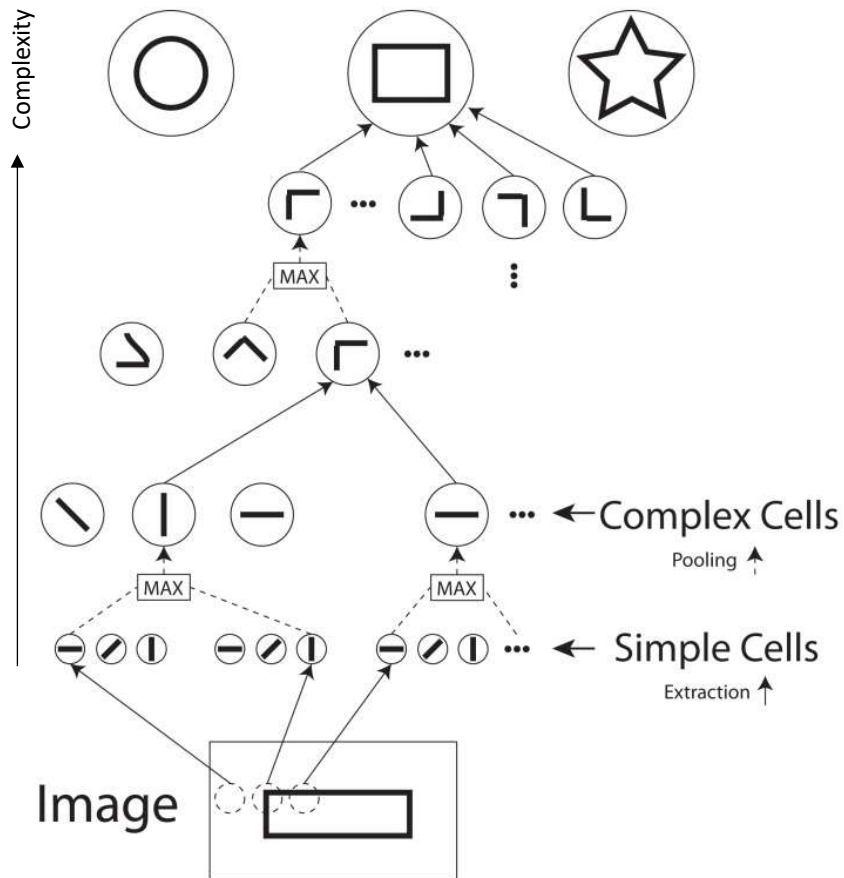


Neuron behavior (simple cells)

Neurons showed stimulus based on orientation (simple cells in visual cortex)



Cell hierarchy of the visual cortex

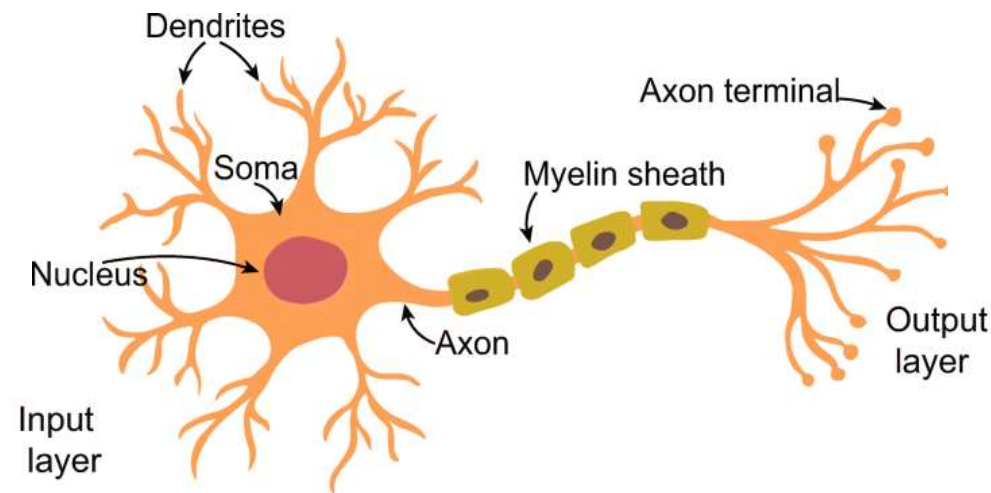


- Simple Cells: respond to points of light or bars of light in a particular orientation
- Complex cells: respond to bars of light in a particular orientation moving in a specific direction
- Hypercomplex Cells: respond to bars of light in a particular orientation, moving in a specific direction, and of a specific line length.

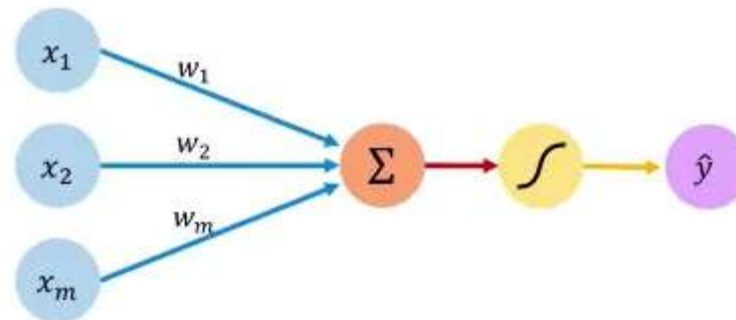
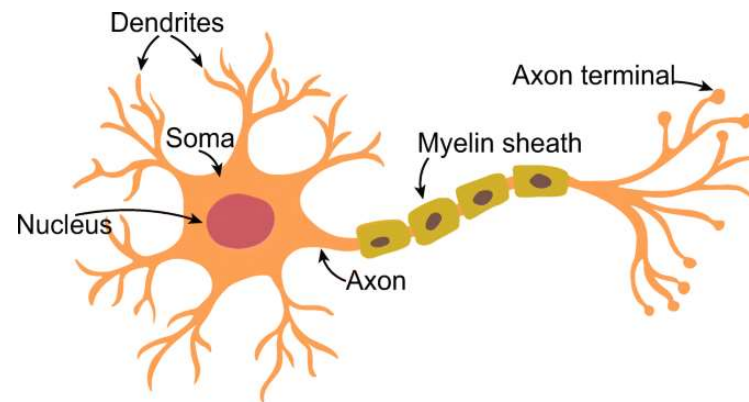
Neural networks

Neural networks were inspired by neural networks in the brain.

Neurons in the brain

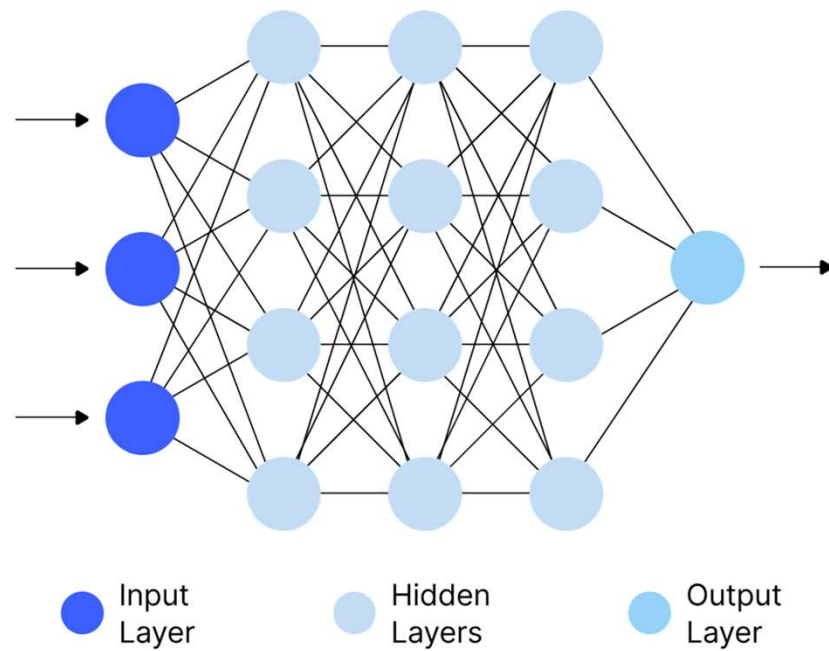


Neural networks



Inputs Weights Sum Non-Linearity Output

Neural networks



Neural networks

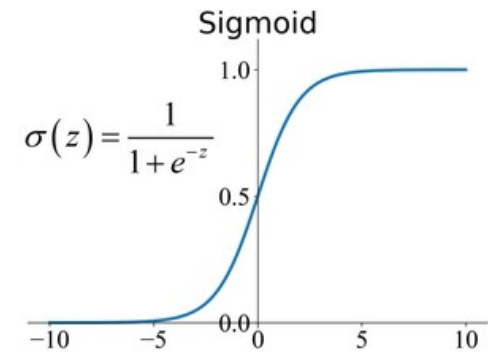
1. Resize & move data (Linear transformations)
2. Reshape & transform data (Activation Functions)
3. Dimension changes/reduction (via neurons in layer)

Activation Functions

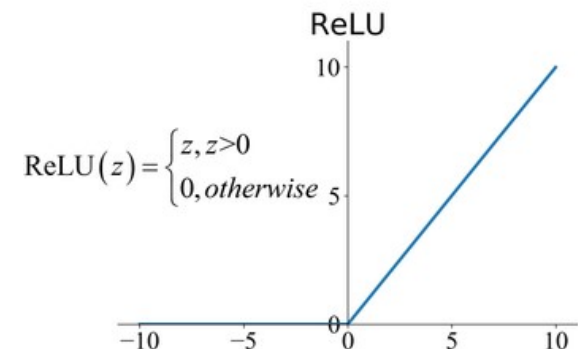
What do we do about non-linear decision boundaries?

Activation Functions

- Activation functions provide non-linear transformation (“reshape” data)
- Sigmoid is typically used for classification e.g. to predict the probability, as an output.
- Rectified Linear Unit (ReLU) is the ‘go-to’ choice for most purposes.
- There are a lot of choices of activation functions – these are only two of many – but they must always be non-linear.



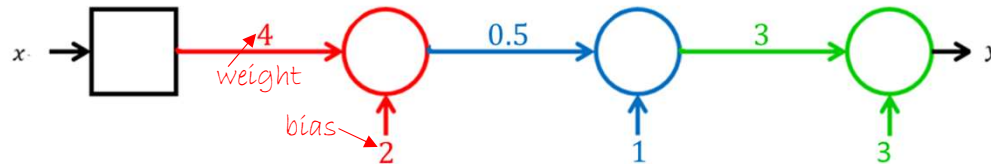
(a)



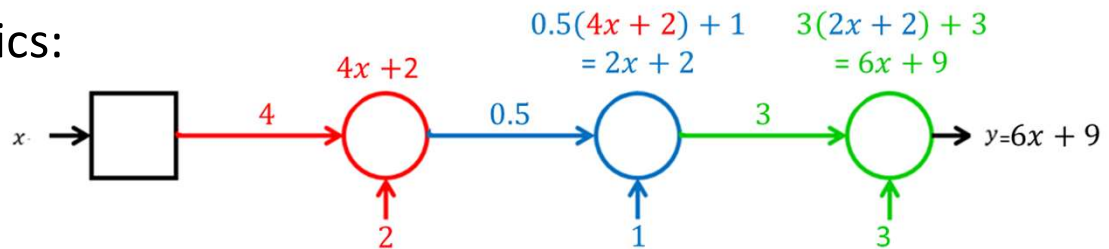
(c)

Do I really need activation functions?

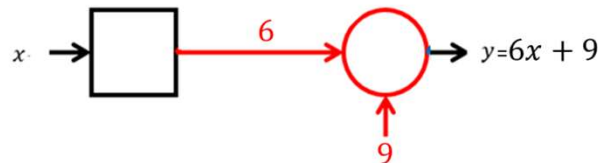
Let's consider a simple neural network:



Let's expand the mathematics:



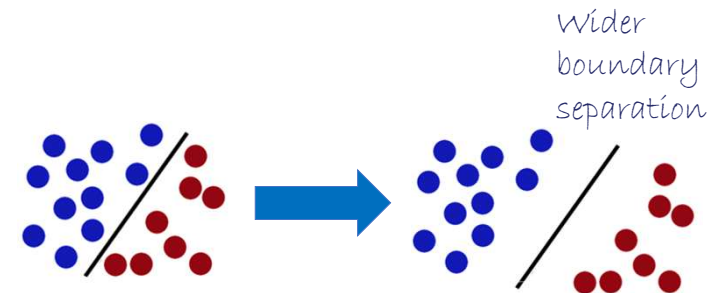
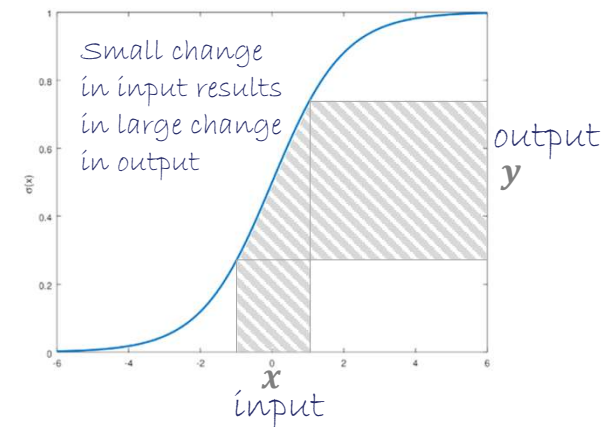
But isn't that the same as the following?



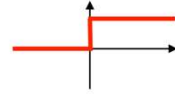
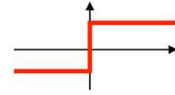

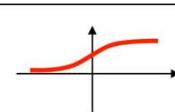
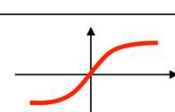
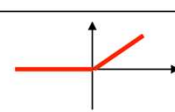
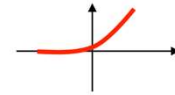
Takeaway: Without a non-linear activation the neural network will collapse

Sigmoid Activation Function

- Useful for output layer for classification problems
- A small change in input causes large change in output, at decision boundary
- Separates classes away from decision boundary
- Softmax performs a similar function for multi-class classification (3 or more classes)



More Activation Functions to choose from

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016
(<http://sebastianraschka.com>)

10

Universal Approximation Theorem

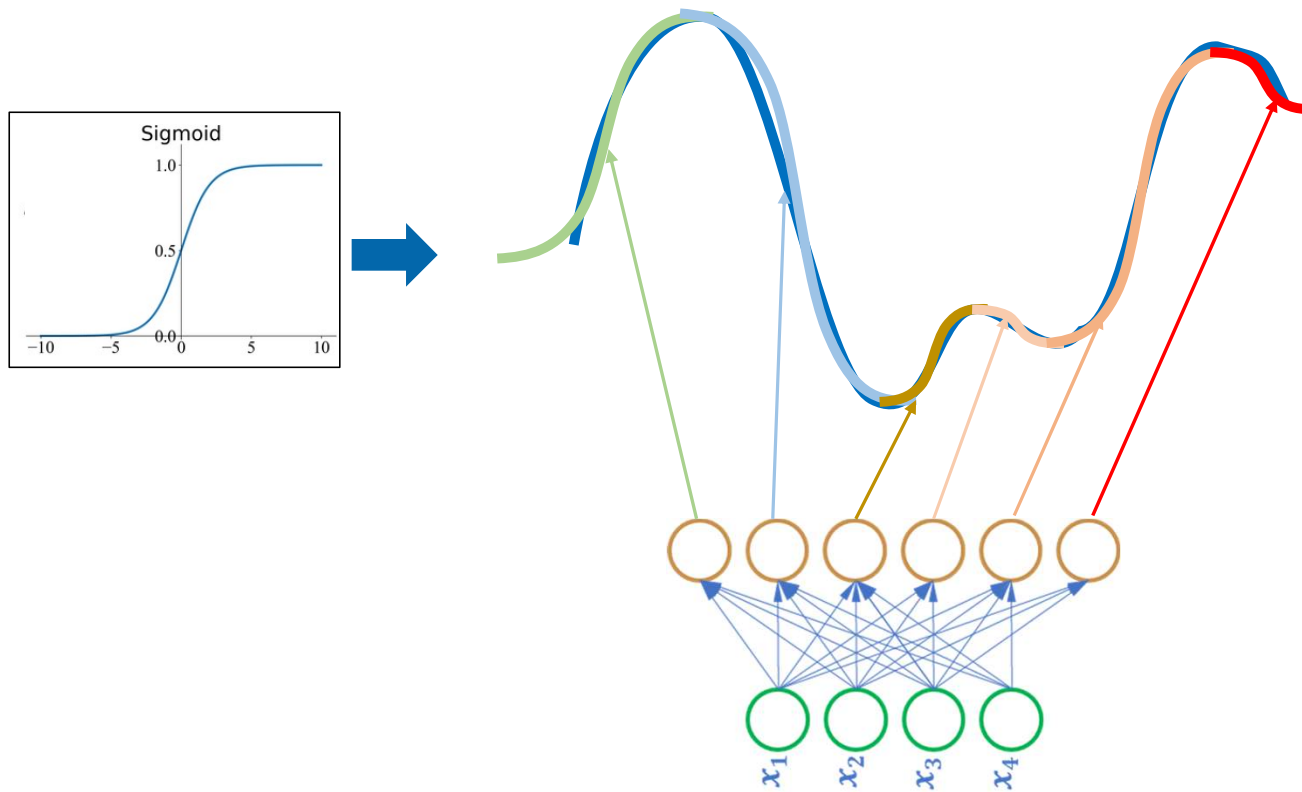
What makes neural networks so powerful?

Universal Approximation Theorem

- A neural network with only 1 hidden layer can approximate any continuous function for inputs within a specific range
- Accuracy of approximation can be controlled by number of layers & neurons
- If the function is not continuous (jumps around or has large gaps) a neural network won't be able to approximate it.

Num Hidden Layers	Result
none	Only capable of representing linear separable functions or decisions.
1	Can approximate any function that contains a continuous mapping from one finite space to another.
2	Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.
>2	Additional layers can learn complex representations (sort of automatic feature engineering) for layer layers.

Universal Approximation Theorem



Note: Simplified diagram – there is also cross-connectivity contribution.

Demo

<https://playground.tensorflow.org/>

Neural network demo

