Neural Networks

Learning Inspired by the Human Brain

Mehul Motani
Electrical & Computer Engineering
National University of Singapore
Email: motani@nus.edu.sg

Learning from data

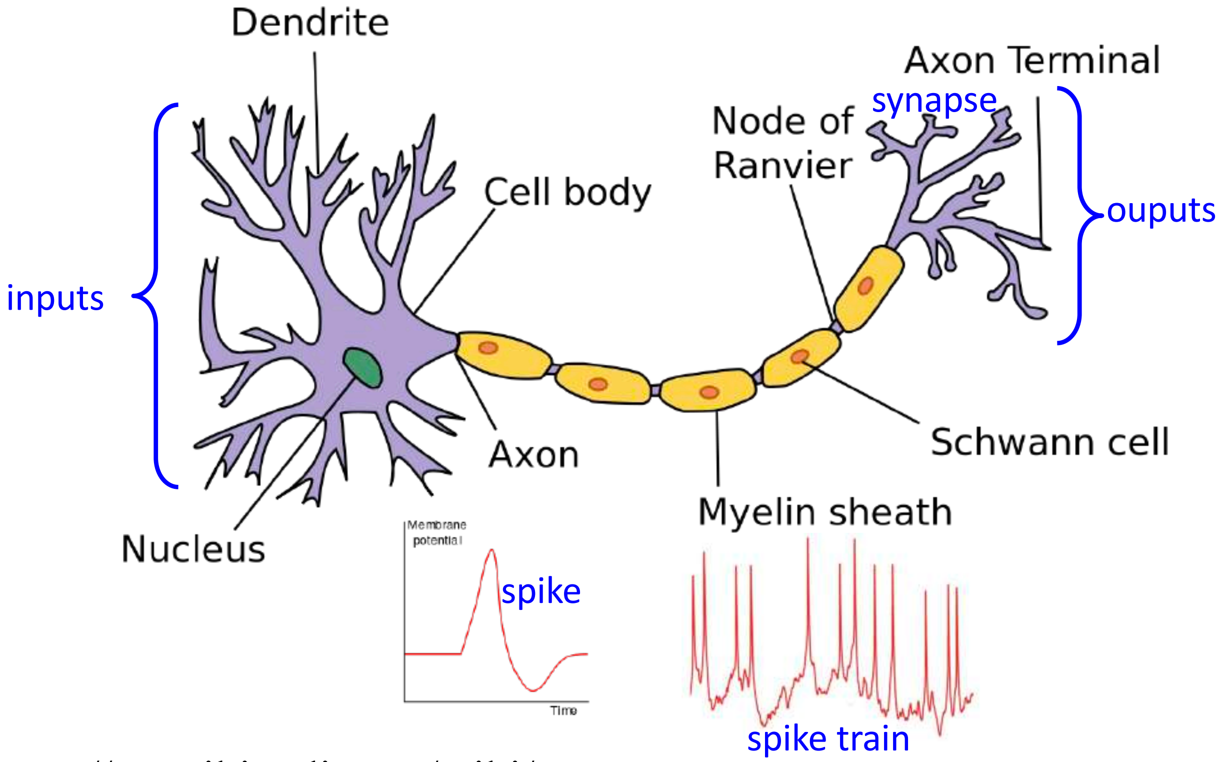
Neural networks are a way to learn from data inspired by the human brain.

Learning from data $\longleftrightarrow$ Machine Learning

Three broad approaches to machine learning

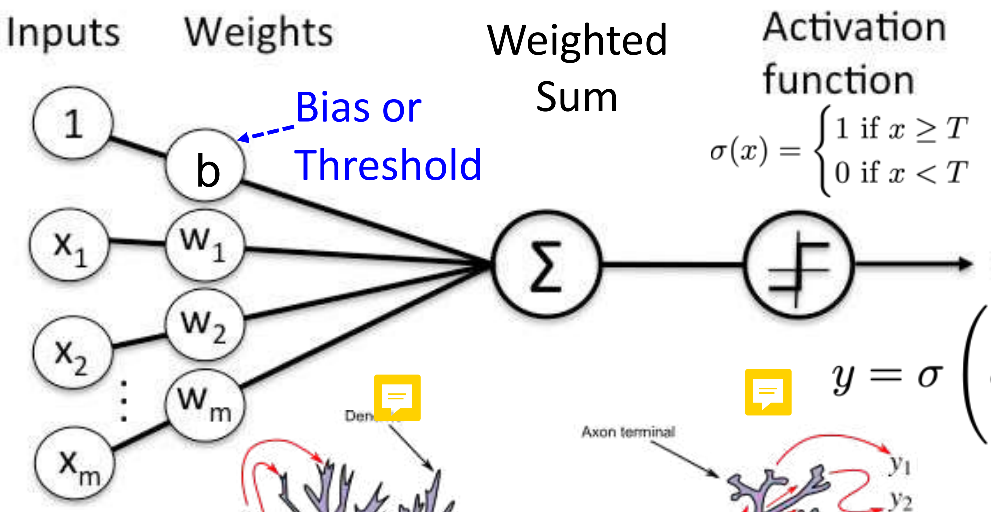
1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

Biological Neuron



https://en.wikipedia.org/wiki/Neuron

The Perceptron – Artificial Neuron

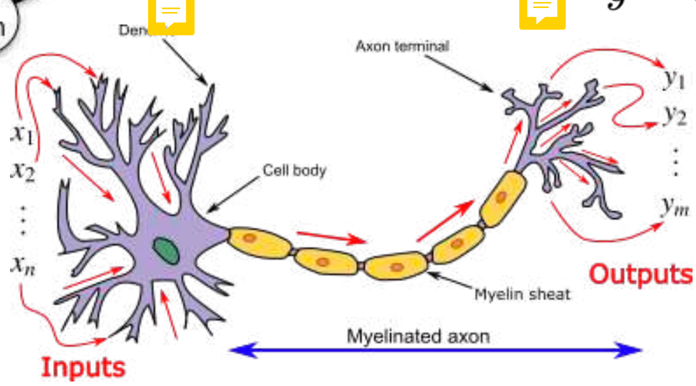


$$\sigma(x) = \begin{cases} 1 \text{ if } x \geq T \\ 0 \text{ if } x < T \end{cases}$$

$$y = \sigma\left(b + \sum_{i=1}^{m} w_i x_i\right)$$

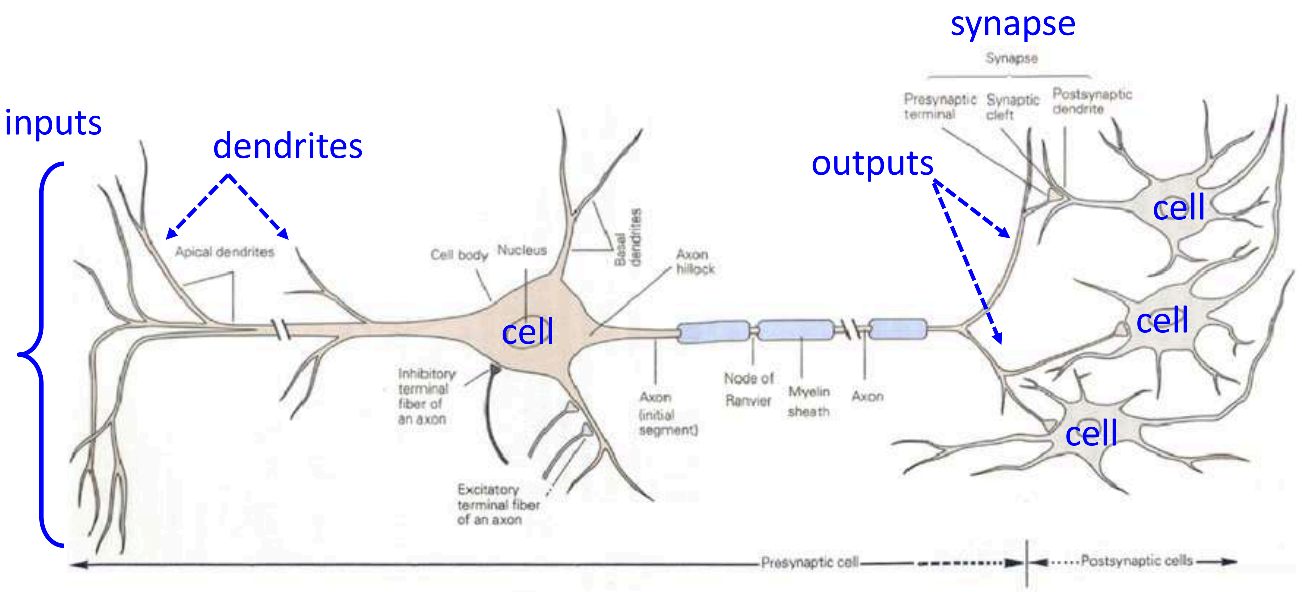
Weighted linear combination of inputs

Compare to



$\boxed{\text{The weights are bias are learned during training}}$

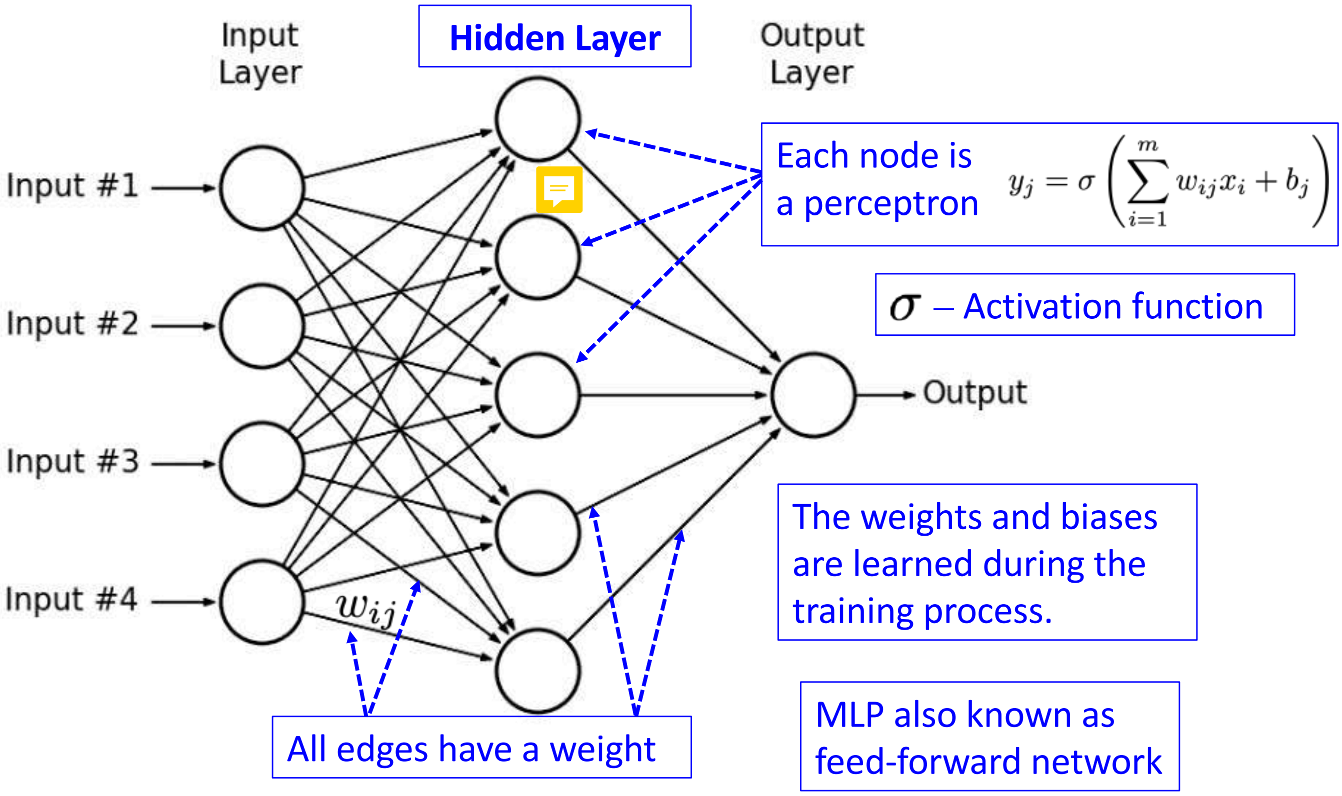
Brain is a Network of Neurons



Source: Kandel, 2000

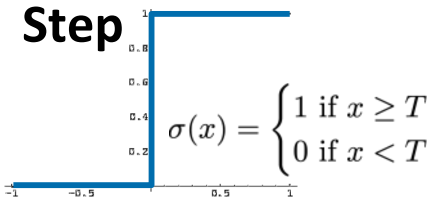
Excellent reference textbook on Neuroscience:
Kandel, E.R., Schwartz, J.H. and Jessell, T.M. eds., 2000. Principles of neural science (Vol. 4, pp. 1227-1246). New York: McGraw-hill.

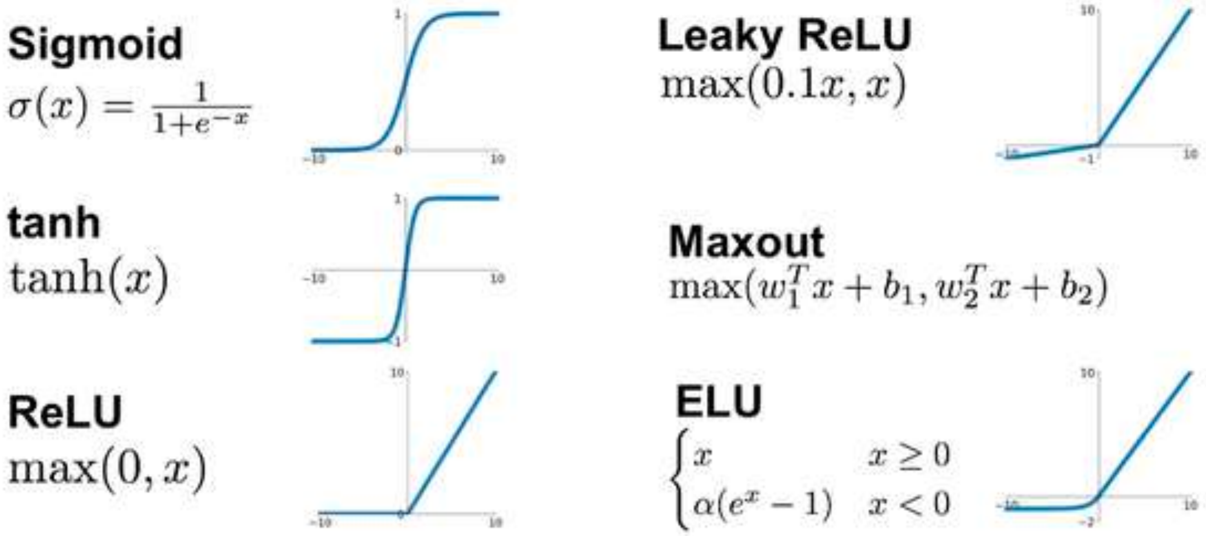
Multilayer Perceptron – Artificial Neural Network



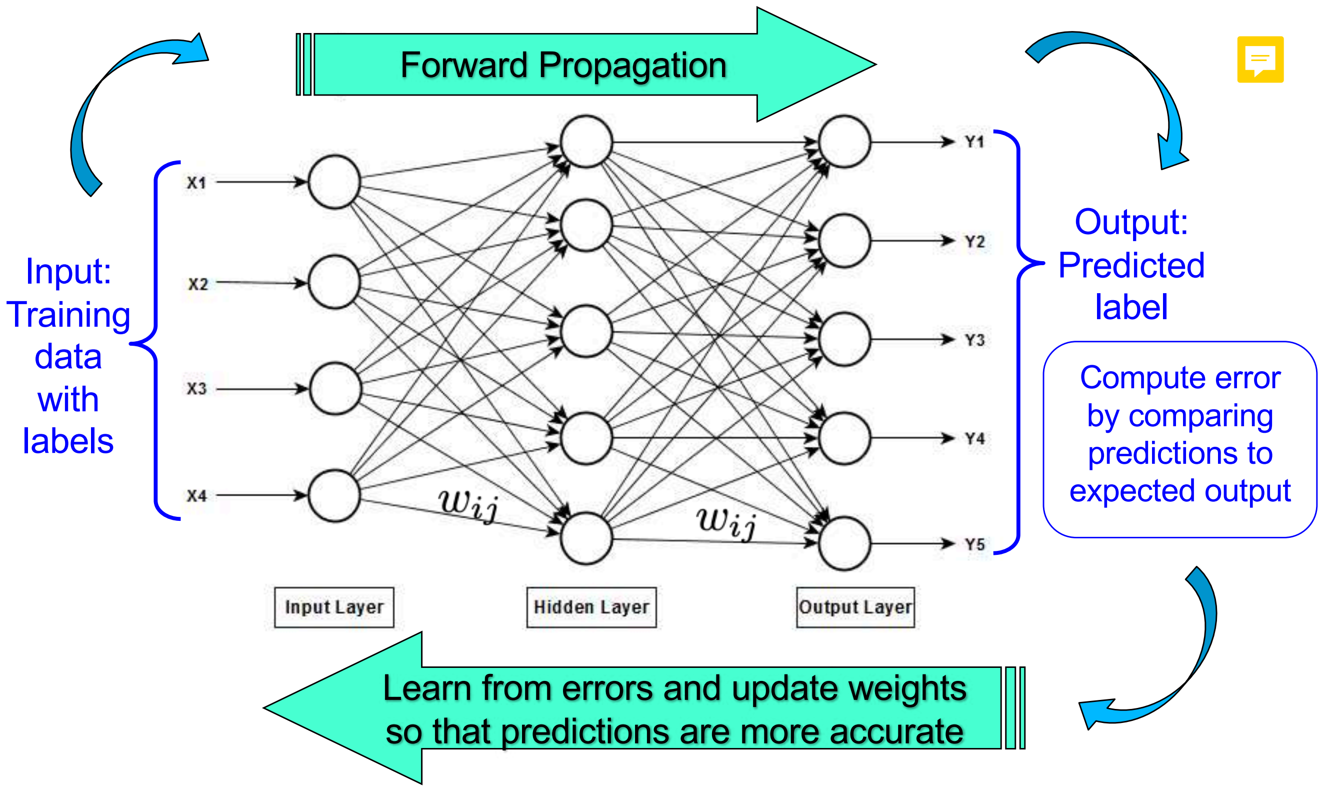
Common Activation Functions

- Activation function – Typically a function with nonlinear input output relationship
- Nonlinear functions are key and allow such networks to solve nontrivial problems
- Simplest activation is the step function, which is inspired by biological systems

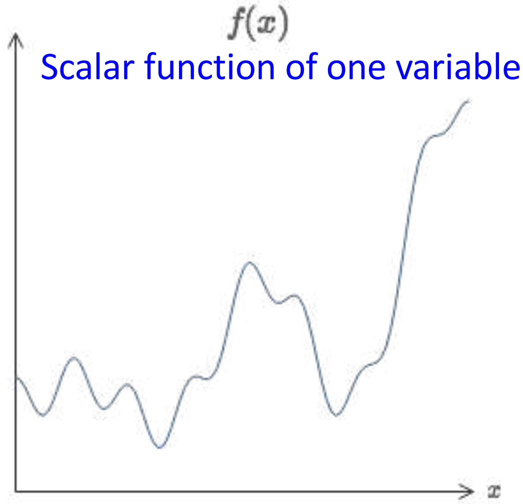


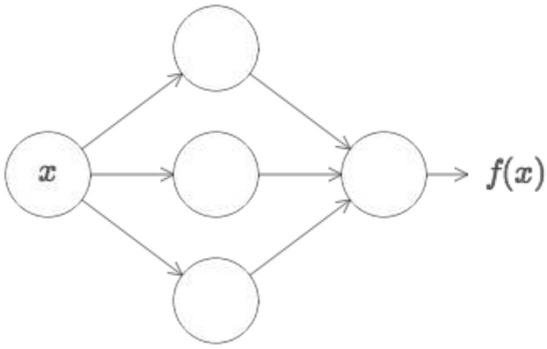


Training a Multilayer Perceptron



Neural networks are good function approximators



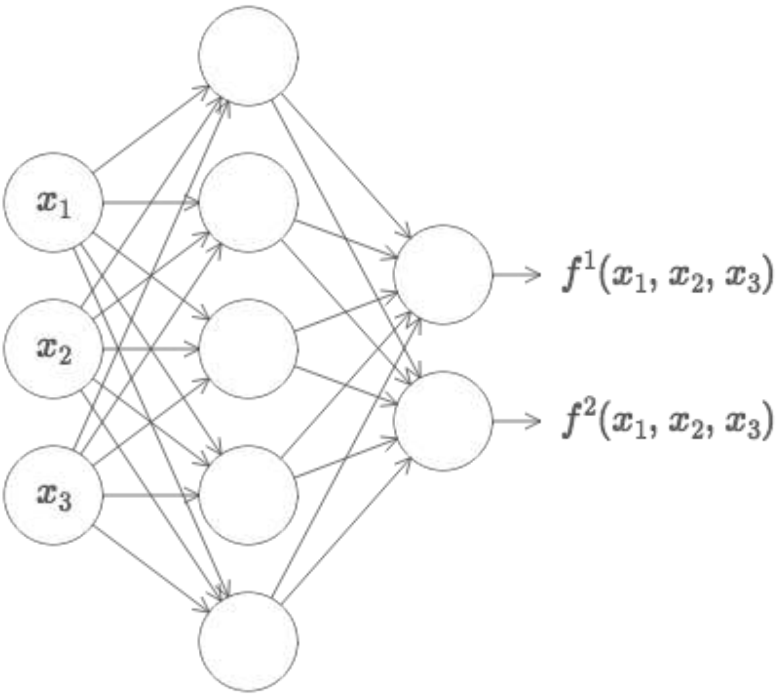


One input, one output, single hidden layer neural network

- A neural network approximates the relationship between the input (features) and the output (labels).
- A neural network can approximate almost any function, no matter how wiggly it is!
- This is a supervised learning regression problem.

Function approximation – multiple inputs and outputs

- Neural networks can also handle if the function has multiple inputs, f=f(x1,…,xm), and mutiple outputs.
- For instance, here's a multilayer perceptron for computing a function with m=3 inputs and n=2 outputs

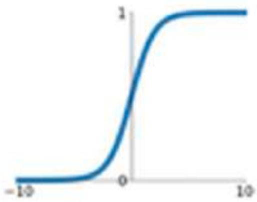


Recall: Multilayer perceptrons are called feed-forward networks.

Neural Networks = Function Approximation

- Universal Approximation Theorem:
  A feed-forward network with a single hidden layer containing a finite number of neurons can approximate any continuous function on compact subsets of R^n, under mild assumptions on the activation function.
- Works for the sigmoid activation function:

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

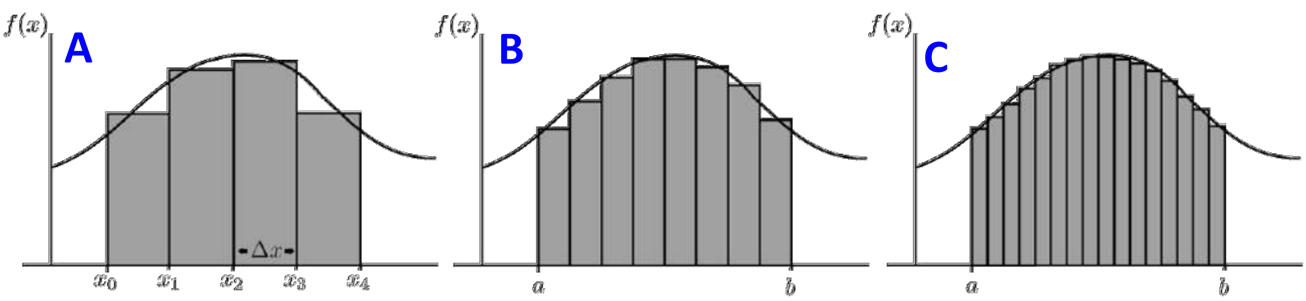


- Works for other specific activation functions as well.
- Note that the number of neurons in the hidden layer may be quite large.

https://en.wikipedia.org/wiki/Universal_approximation_theorem
http://neuralnetworksanddeeplearning.com/chap4.html

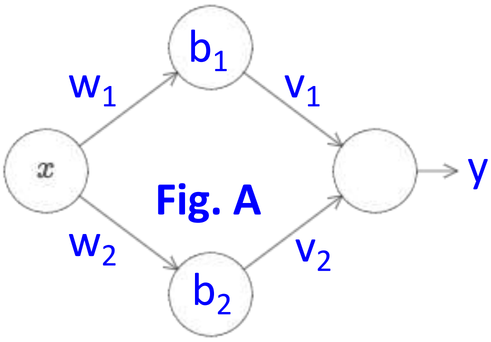
Intuition Behind Function Approximation

- In calculus, sometimes we use a finite series of rectangles to represent or approximate a given function.
- The value of the function at some point *x* is the height of the rectangle at *x*.
- As the rectangles get thinner, the approximation becomes more and more accurate.
- Neural networks build a series of rectangles to approximate functions in the same way.



Example: two-node single hidden layer MLP

- Suppose we have a single input, single output, two-node single hidden layer MLP.



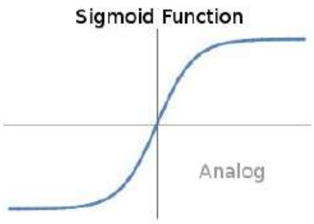
Output of Neuron 1 (on top):

$$y_1 = \sigma\left(w_1 x + b_1\right) \quad \textbf{(1)}$$

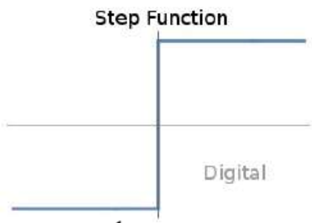
Output of Neuron 2 (on bottom):

$$y_2 = \sigma\left(w_2 x + b_2\right) \quad \textbf{(2)}$$

Example activation function **σ**



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



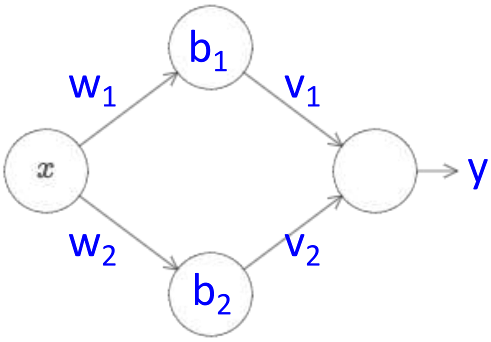
$$\sigma(x) = \begin{cases} 1 \text{ if } x \geq T \\ 0 \text{ if } x < T \end{cases}$$

Final output of MLP:

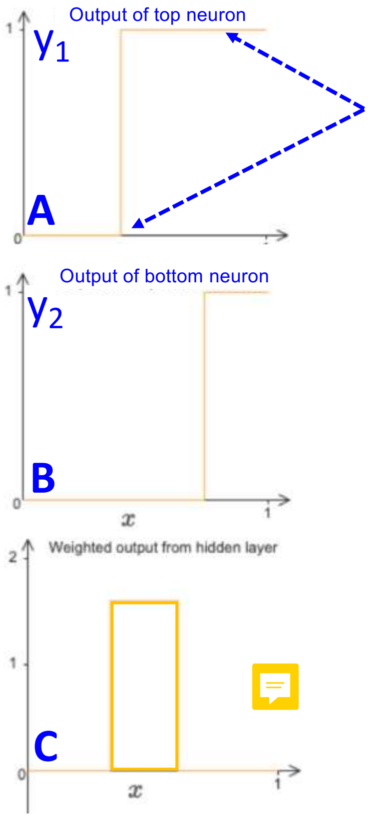
$$y = v_1 y_1 + v_2 y_2 \quad \textbf{(3)}$$

Example: two-node single hidden layer MLP

- Assume step activation.
- Works for other activation functions too.



$$y_1 = \sigma\left(w_1 x + b_1\right)$$
$$y_2 = \sigma\left(w_2 x + b_2\right)$$
$$y = v_1 y_1 + v_2 y_2$$



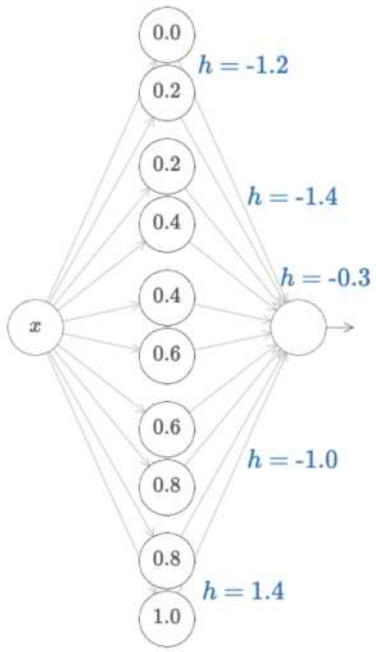
Height and threshold for y_1 depend upon w_1 and b_1.

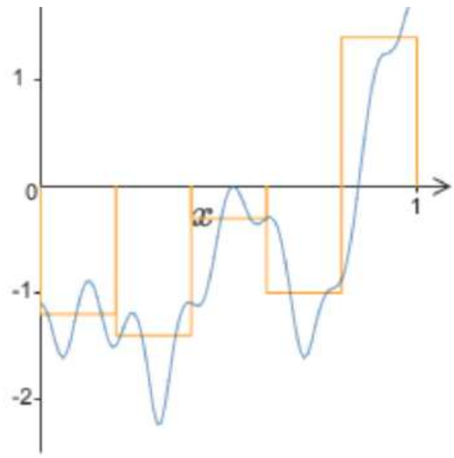
Same for y_2

Then we can choose v_1 and v_2 to make the final output a ‘rectangle’.

Function approximation with single hidden layer multilayer perceptron

- The idea then is to use multiple rectangles to approximate the given function.





- The weights and biases are learned from the labeled dataset during training.
- We get better approximations by increasing the number of nodes in the hidden layer.

Function approximation – Boolean functions

AND

Inputs		Output
A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

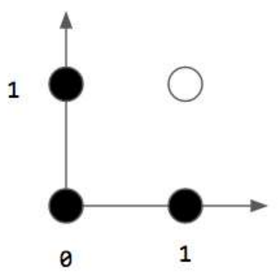
Inclusive OR

Inputs		Output
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

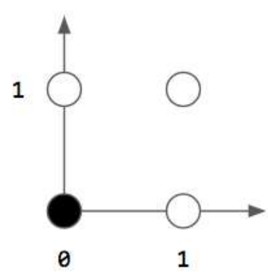
Exclusive OR

Inputs		Output
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

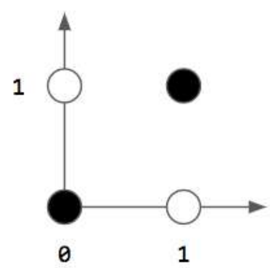
AND



OR

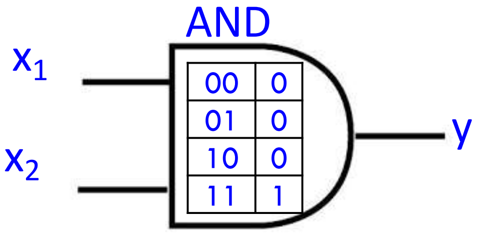


XOR

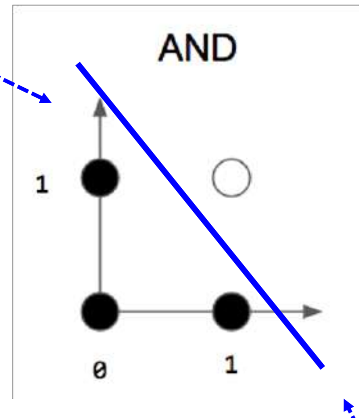


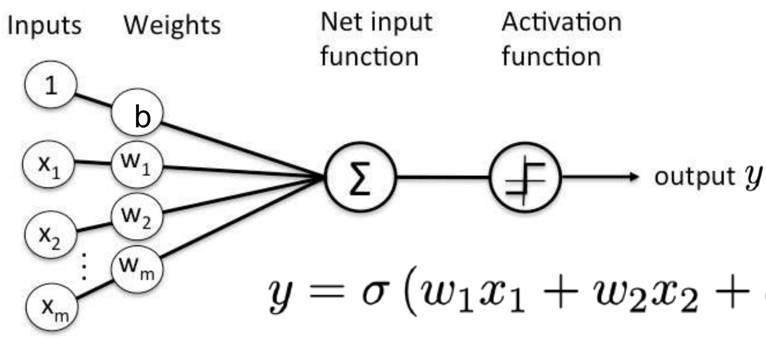
This is a supervised learning classification problem.

Perceptron for implementing the AND gate



Linearly separable set



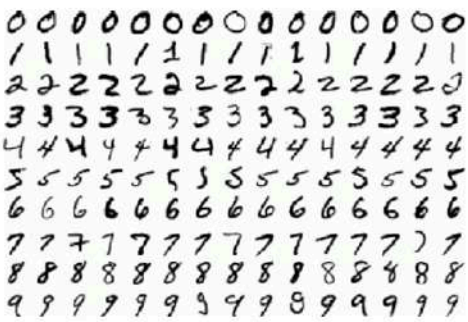


$$y = \sigma\left(w_1 x_1 + w_2 x_2 + b\right)$$ Equation of line

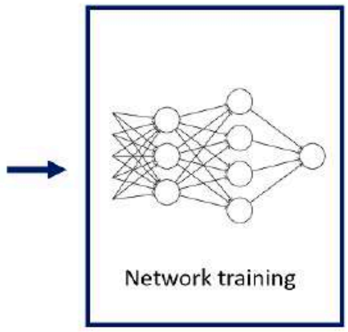
- The perceptron returns a linear separator!
- The weights and biases are learned during training.

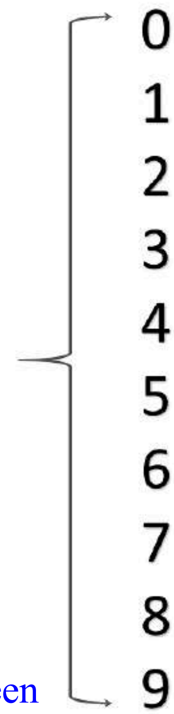
Function approximation – Image Classification

- MNIST dataset of handwritten digits
  - 10 classes (digits 0 – 9)
  - 60,000 training examples
  - 10,000 test examples



Data & Labels





- This is a supervised learning classification problem.
- The function being approximated is the mapping between image (which are the features) and the class label.
- This can be solved via MLP or CNN.

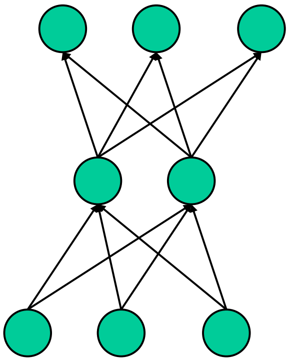
Brief History of Neural Networks

- 1943: McCulloch–Pitts "neuron" – Started the field
- 1962: Rosenblatt's perceptron
  - Learned its own weight values; convergence proof
- 1969: Minsky & Papert book on perceptrons
  - Proved limitations of single-layer perceptron networks
- 1982: Hopfield's associative memories, convergence in symmetric nets
  - Introduced energy-function concept
- 1986: Backpropagation of errors (Hinton)
  - Method for training multilayer networks
  - https://www.nature.com/articles/323533a0
- 1997: A recurrent neural network framework called Long Short-Term Memory (LSTM) was proposed by Schmidhuber & Hochreiter.
- 2000 & beyond – The Rise of Deep Learning
  - Probabilistic interpretations, Bayesian and spiking networks
  - Convolutional neural networks, Generative Adversarial Networks
  - Recurrent neural networks, Transformer networks with Attention

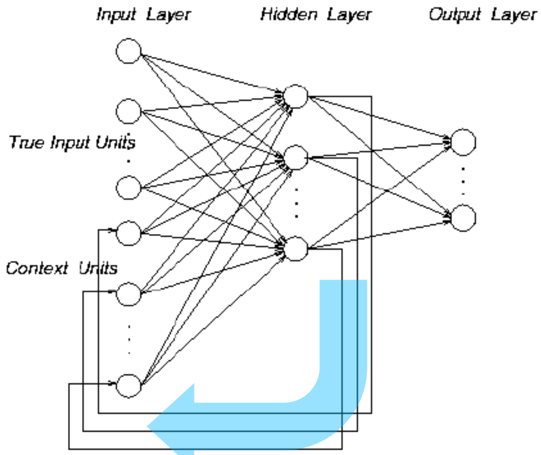
Types of Neural Networks

- Feedforward versus recurrent networks
  - Feedforward: No loops, input → hidden layers → output
  - Recurrent: Uses feedback (positive or negative)
- Continuous versus spiking
  - Spiking neural networks encode information in spike trains.
  - Continuous networks model mean spike rate (firing rate)
  - Consistent with rate-code model of neural coding
- Supervised versus unsupervised learning
  - Supervised networks use a "teacher"
    - The desired output for each input is provided by user
  - Unsupervised networks find hidden statistical patterns in input data

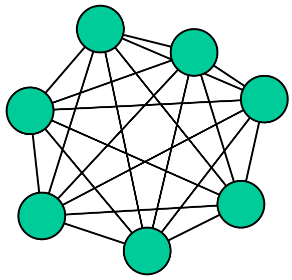
Topologies of Neural Networks



Feed-forward
(directed, a-cyclic)



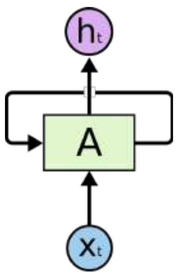
Recurrent
(feedback connections)



Fully
connected

Deep Learning – Many hidden layers and many nodes per layer
Example: VGG-16 has 41 layers, 16 layers with learnable weights, and about 100 million learnable parameters

Recurrent neural networks (RNN)

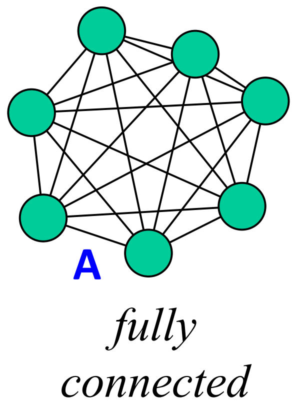


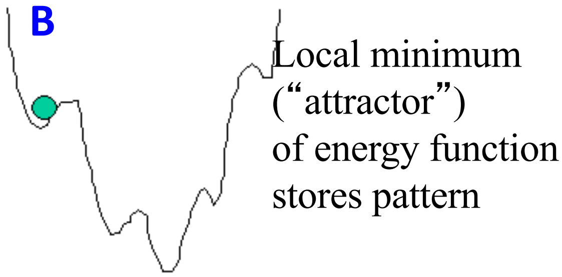
- Employ feedback (positive, negative, or both)
  - Not necessarily stable
  - Symmetric connections can ensure stability
- Why use recurrent networks?
  - Can learn temporal patterns (time series or oscillations)
  - Application to forecasting of time series data
  - Application to machine language translation
  - Biologically realistic – Majority of connections to neurons in cerebral cortex are feedback connections from local or distant neurons
- Examples of RNNs
  - Hopfield networks – associative memories
  - Boltzmann machine (Hopfield-like net with input & output units)
  - Recurrent backpropagation networks: for small sequences, unfold network in time dimension and use backpropagation learning
  - Long Short-Term Memory (LSTM)

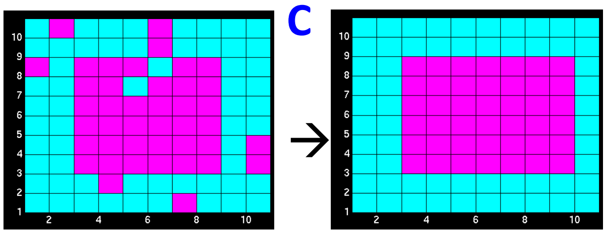
https://en.wikipedia.org/wiki/Recurrent_neural_network
https://en.wikipedia.org/wiki/Long_short-term_memory

Hopfield networks

- A type of recurrent neural network with a fully connected architecture
- Serves as content-addressable ("associative") memory system with binary threshold nodes.
  - Asynchronous updating of outputs
  - Hebbian learning rule for Hopfield networks



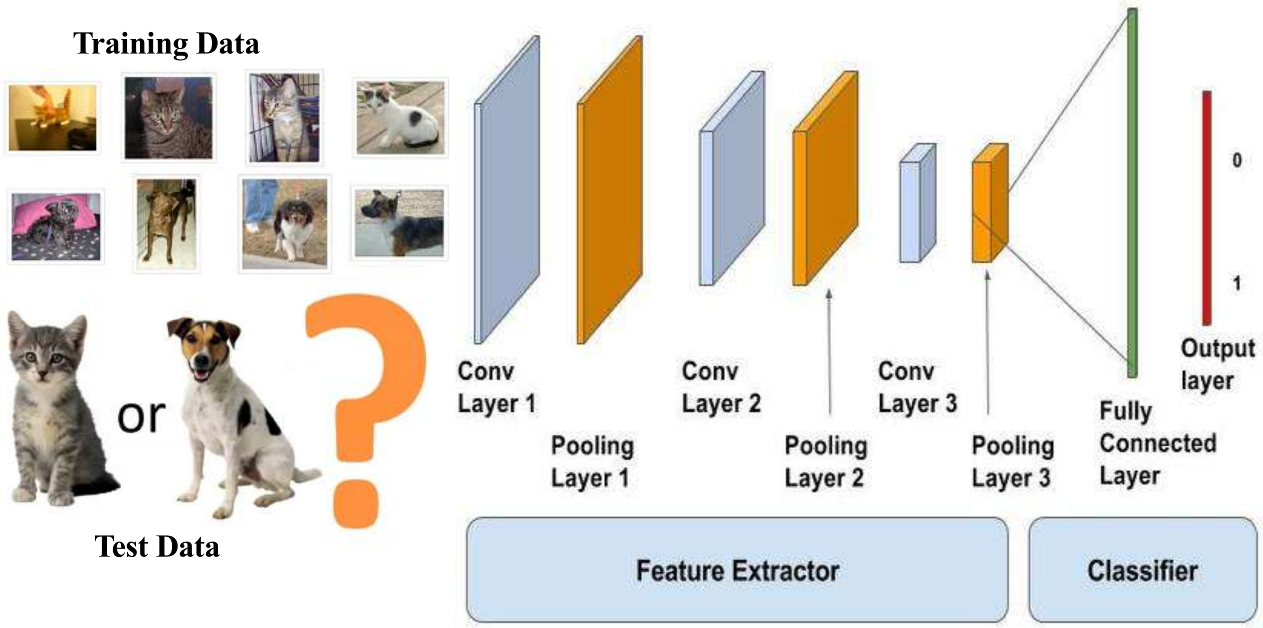




https://en.wikipedia.org/wiki/Hopfield_network

Convolutional Neural Networks (CNN)

Deep Learning Image Classification using Convolutional Neural Networks



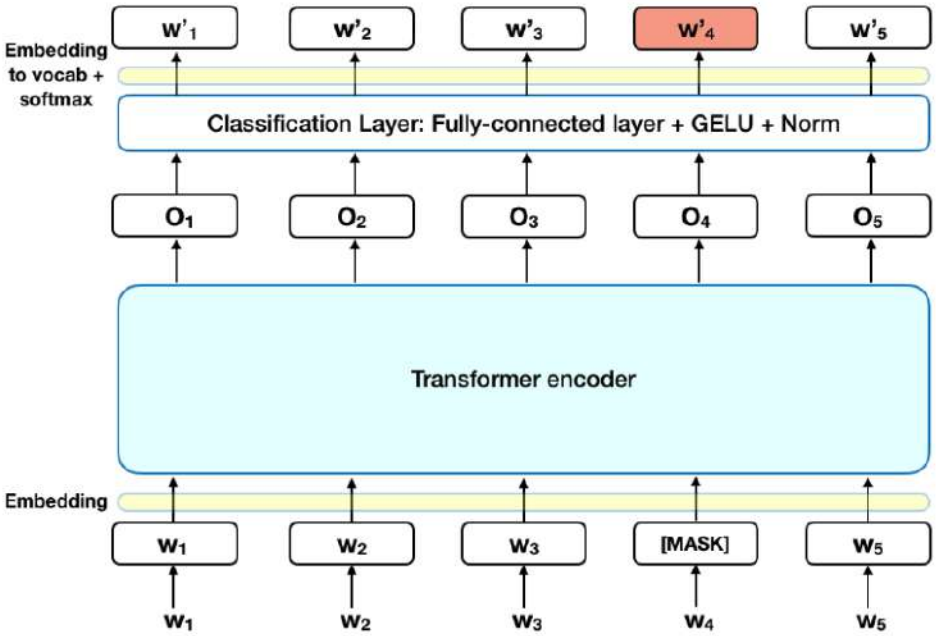
Transformer Networks

Deep Learning Natural Language Processing using BERT (Bidirectional Encoder Representations from Transformers)

Training Data

Large corpus of text data containing pairs of sentences in a certain language.

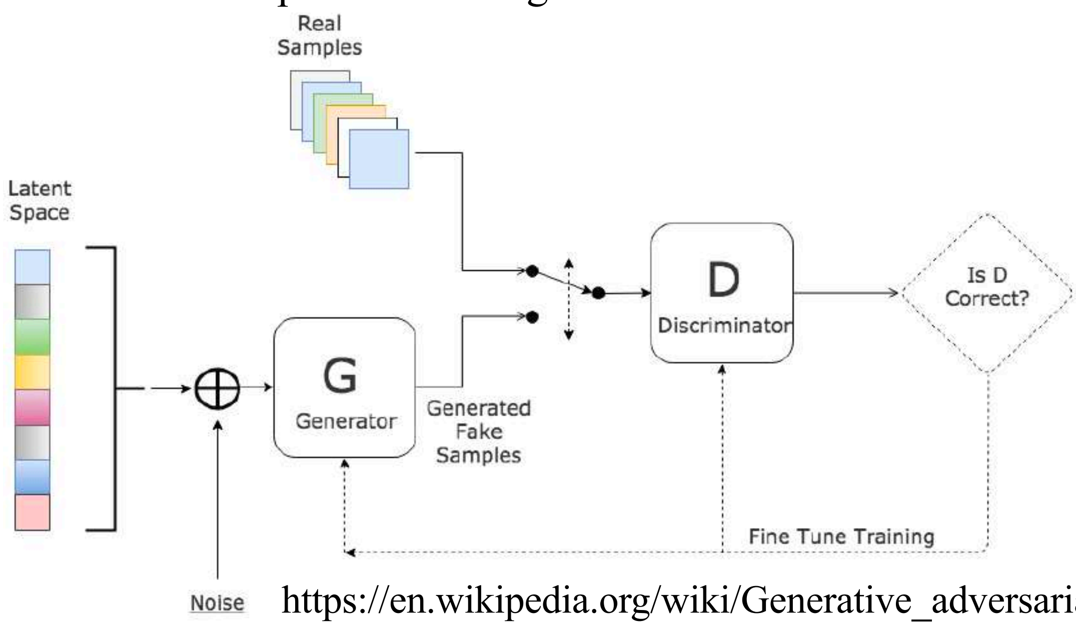
BERT is useful in many NLP tasks such as Question-Answering, Natural Language Understanding, and Machine Translation.



Source: towarddatascience.com

Generative Adversarial Network (GAN)

- GANs allow you to learn to generate new fake data with the same statistics as the given training set.
- GANs consist of two neural networks that compete with each other and result in improved learning for both networks.



https://en.wikipedia.org/wiki/Generative_adversarial_network

Deep Fakes Images via GANs

These faces are all fakes!



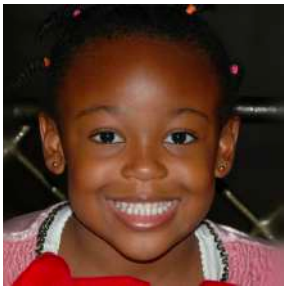

https://thispersondoesnotexist.com/

Deep Fakes Images via GANs

In each pair, one face is real and the other is fake.
Can you tell which face is real?

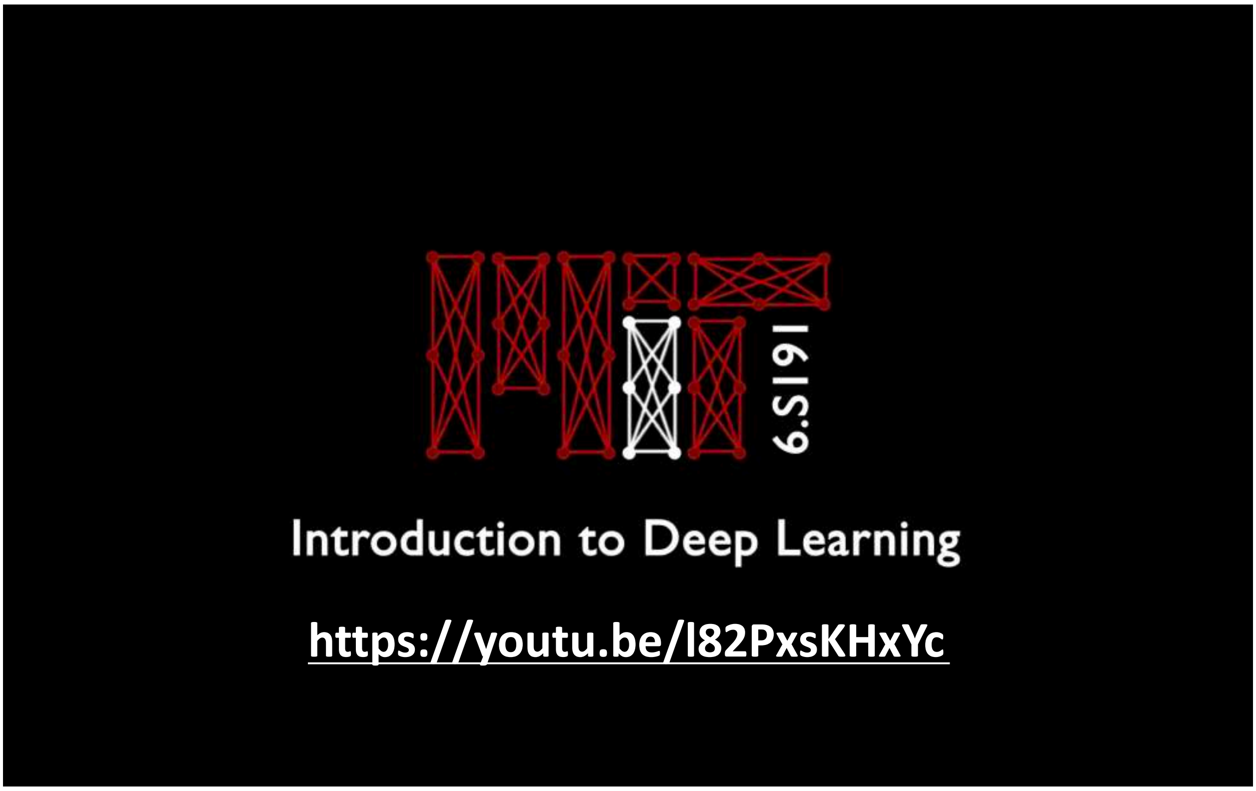

OR



OR


http://www.whichfaceisreal.com

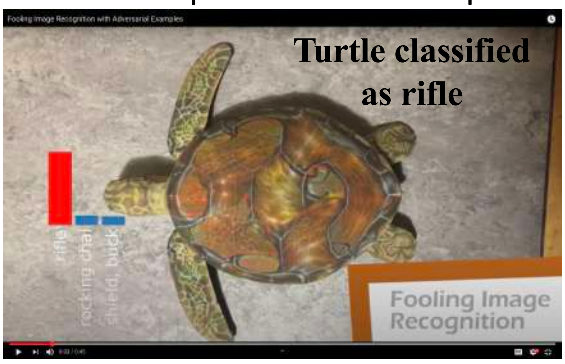
Deep Fakes Videos via GANs (watch)



Adversarial Attacks to Fool Neural Networks

1. These carefully modified stop signs are interpreted as speed limit signs by AI algorithms in self driving vehicles. (Eykholt et al, CVPR 2018)

2. More examples that fool deep learning AI. (Athalye et al, ICML 2018)



Watch: https://youtu.be/piYnd_wYlT8

Adversarial attacks on AI (watch)

https://youtu.be/Exd6CLAYOh0

Unsupervised Neural Networks

- No feedback to say how output differs from desired output (no error signal) or even whether output was right or wrong
- Network must discover patterns in the input data by itself
  - Only works if there are redundancies in the input data
  - High dimensional data → Reduce dimensionality → Clustering
- Self organizing map
  - Uses unsupervised learning to produce a low-dimensional discretized representation of the input space of the training samples
- Autoencoder
  - Model is trained to generates a compact representation of the input data and use it to reconstruct the input with as high fidelity as possible.

https://en.wikipedia.org/wiki/Unsupervised_learning
https://en.wikipedia.org/wiki/Self-organizing_map
https://en.wikipedia.org/wiki/Autoencoder

Thank you!

- Please send me your feedback and any questions you may have.
- The best way to contact me is via email:

  mehul.motani@gmail.com
- Thanks for listening!