Neural Networks

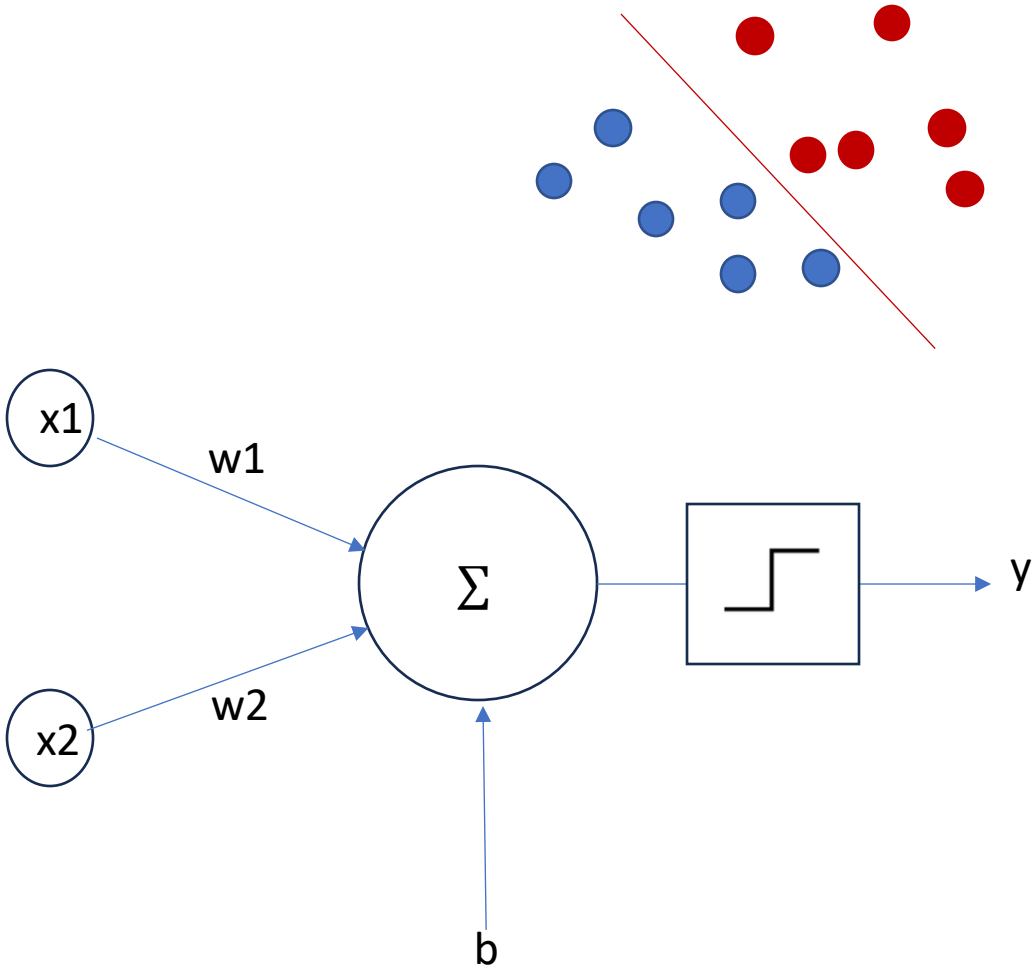
Machine Learning Theory (MLT)

Edinburgh

Rik Sarkar

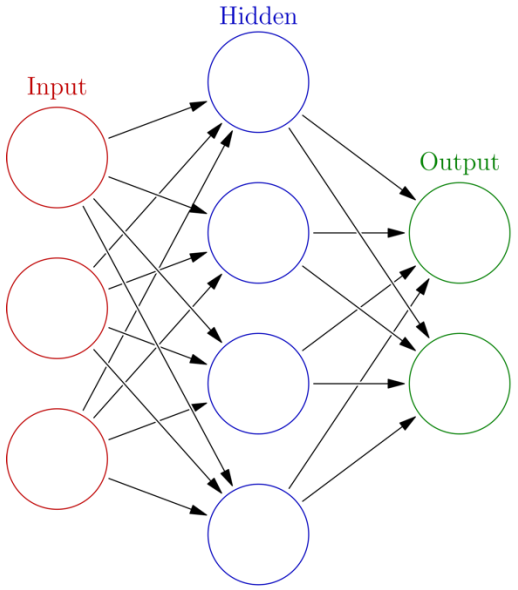
Single neuron

- Perceptron with threshold activation
  - $a_1, a_2, b \in \mathbb{R}$
- $y = (w_1 x_1 + w_2 x_2 + b \geq 0)$
  - Truth value 0/1

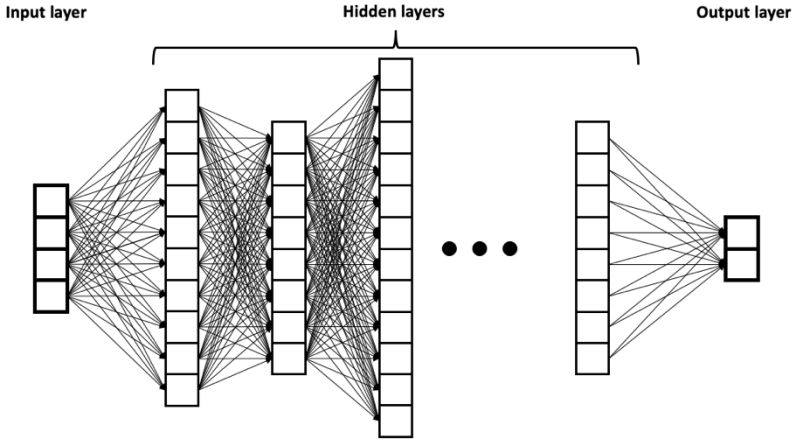


Neural network layers

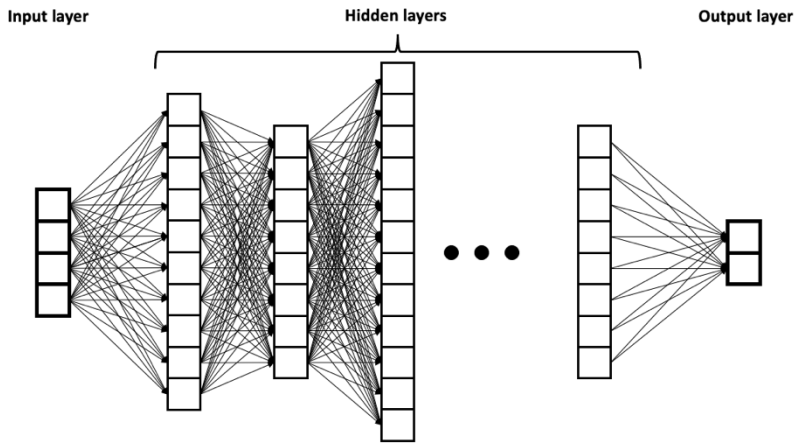
- Input, output and hidden layers
  - Input later: just the individual input variables
  - Hidden and output layers: Activation functions



- Deep Neural networks
  - Multiple hidden layers
  - Output of a layer is the input to the next layer
  - So we can get more complicated functions as output



Multilayer perceptron



Usually

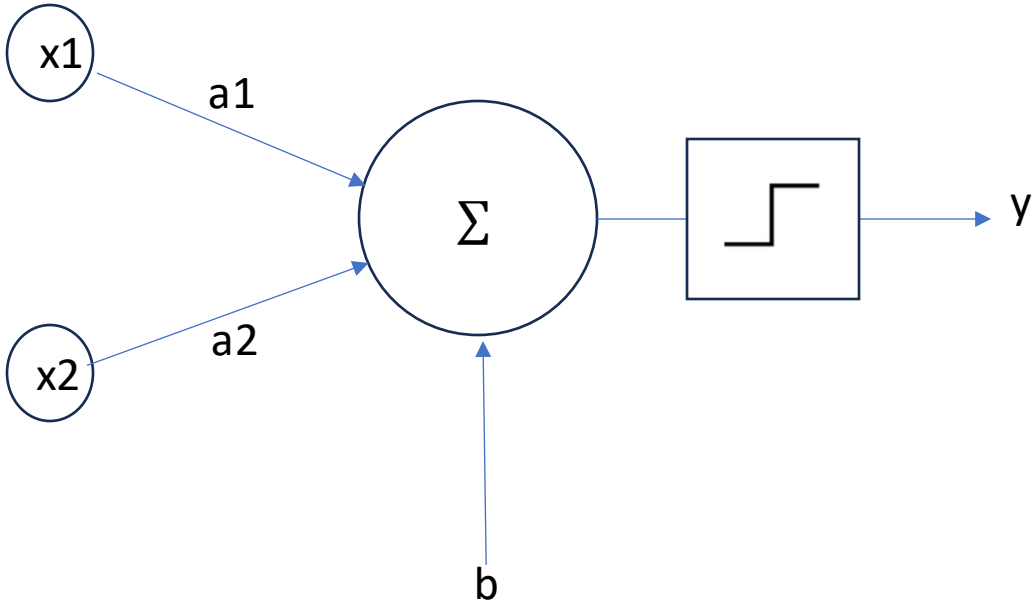
- Multiple hidden layers
- Feedforward (no arrows going backward)
- Fully connected (all outputs of layer x go into all neurons of layer x+1)
- Non-linear activation functions
  - Threshold, sigmoid, ReLU etc

Other types of deep networks

- Can have backward and skip connections, not fully connected etc

Logic using perceptron

- Let us Implement Boolean expressions using perceptrons
- Perceptron with threshold activation
  - Suppose $y, x_1, x_2 \in \{0,1\}$
  - $w_1, w_2, b \in \mathbb{R}$
- $y = (w_1 x_1 + w_2 x_2 + b \geq 0)$
  - Truth value 0/1
- How would you implement OR?
  - Select values of w_1, w_2, b



Logic gates and Boolean expressions

- Logic gates Can be implemented using perceptrons and threshold activation

- Deep neural network can implement arbitrary logic functions/expressions
  - With suitable choice of weights

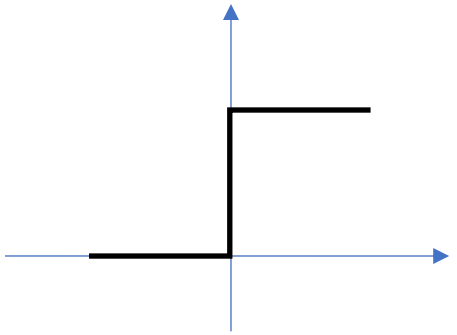
Homogeneous coordinates

- The expressions of type
  - $z = w_1 x_1 + w_2 x_2 + \cdots + b$
  - Occurs all the time

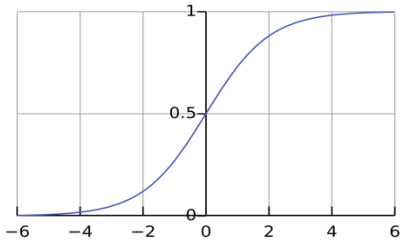
- It is easier to write them as:
  - $z = \sum w_i x_i + b$
- Let us create $x_0 = 1$, and set $w_0 = b$
  - Then we can write just $\mathrm{z} = \sum w_i x_i$
  - With suitable vectors W and x, this is dot product $\mathrm{z} = Wx$

Activation functions

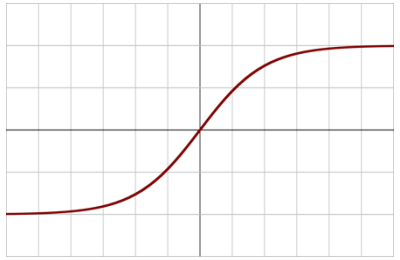
- With $z = Wx = \sum w_i x_i$
- Threshold or step function is $(z \geq 0)$
  - $y \in \{0, 1\}$, or $y \in \{-1, 1\}$



- Sigmoid, logistic or soft step
  - $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$
  - Like step, but a but a bit smoother

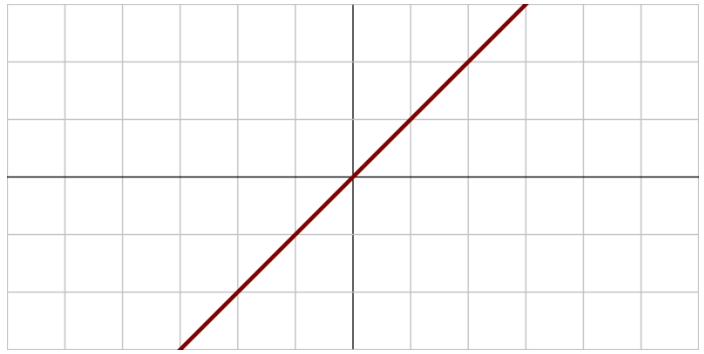


- $\tanh$: hyperbolic tangent
  - $\tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}}$



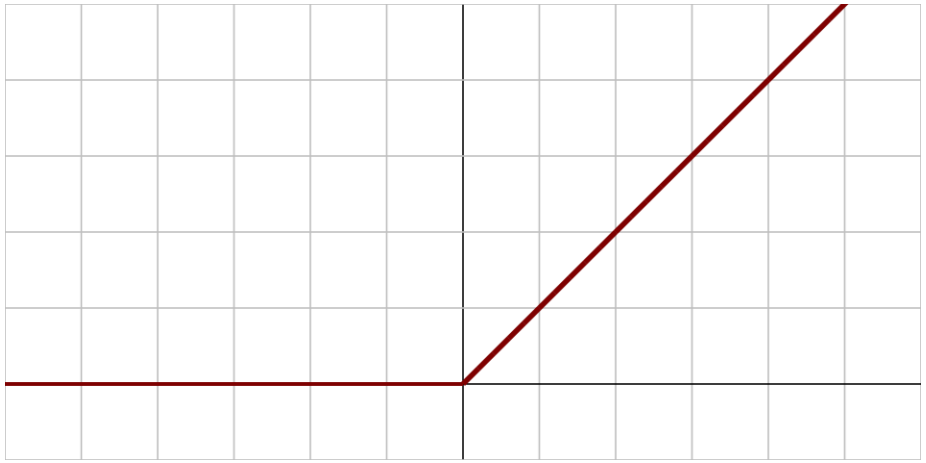
Linear activation

- $y = cz$



ReLU

- Rectified linear unit most common type of activation today
- $y = \max(0, z)$
- Variants exist with smoother curve, leaky ReLU etc
- For many examples, see https://en.wikipedia.org/wiki/Activation_function



Softmax output

- The output for classifiers often uses a different function called softmax
- Suppose the classification problem has n classes
- The the output layer has n outputs
  - $s_1, \ldots, s_n$
  - Often Called Logits – scores for each class
- We output softmax for each class
  - $sm_i = \frac{e^{s_i}}{\sum e^{s_i}}$
- The maximum among these represents the true class

Now that you know Neural networks:

- Try out:
  - https://playground.tensorflow.org/

- For one of the datasets (e.g. the circular one or the daimetrically opposite quadrants one) modify the hidden layer so that tanh activation does not find a solution, but ReLU does!
  - Post on Piazza!