Neural Networks

An Introduction

Lovedeep Gondara

February 28, 2018

Inspiration

- Brain has about 100 billion neurons
- Sparsely connected with 10^4 connections per neuron
- Massive parallel processing

Inspiration

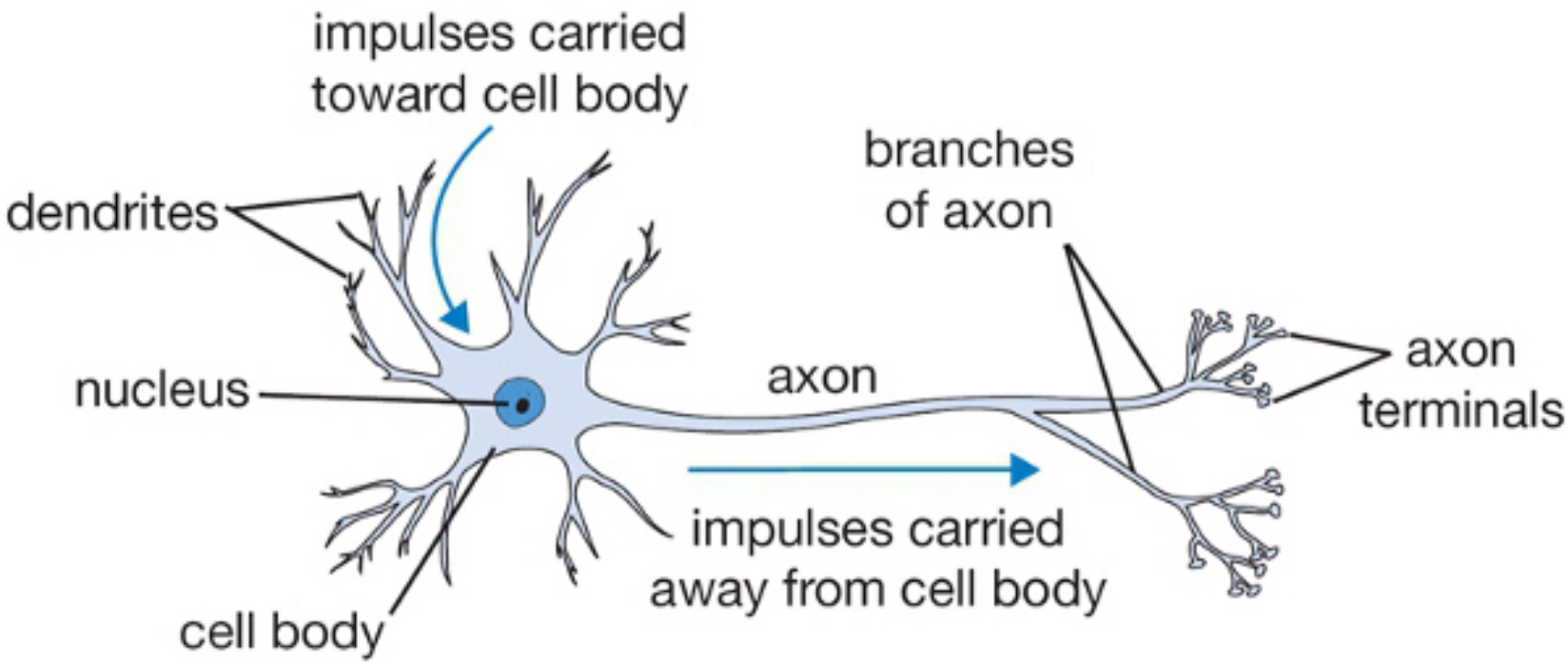


Figure: Biological neuron

Realization

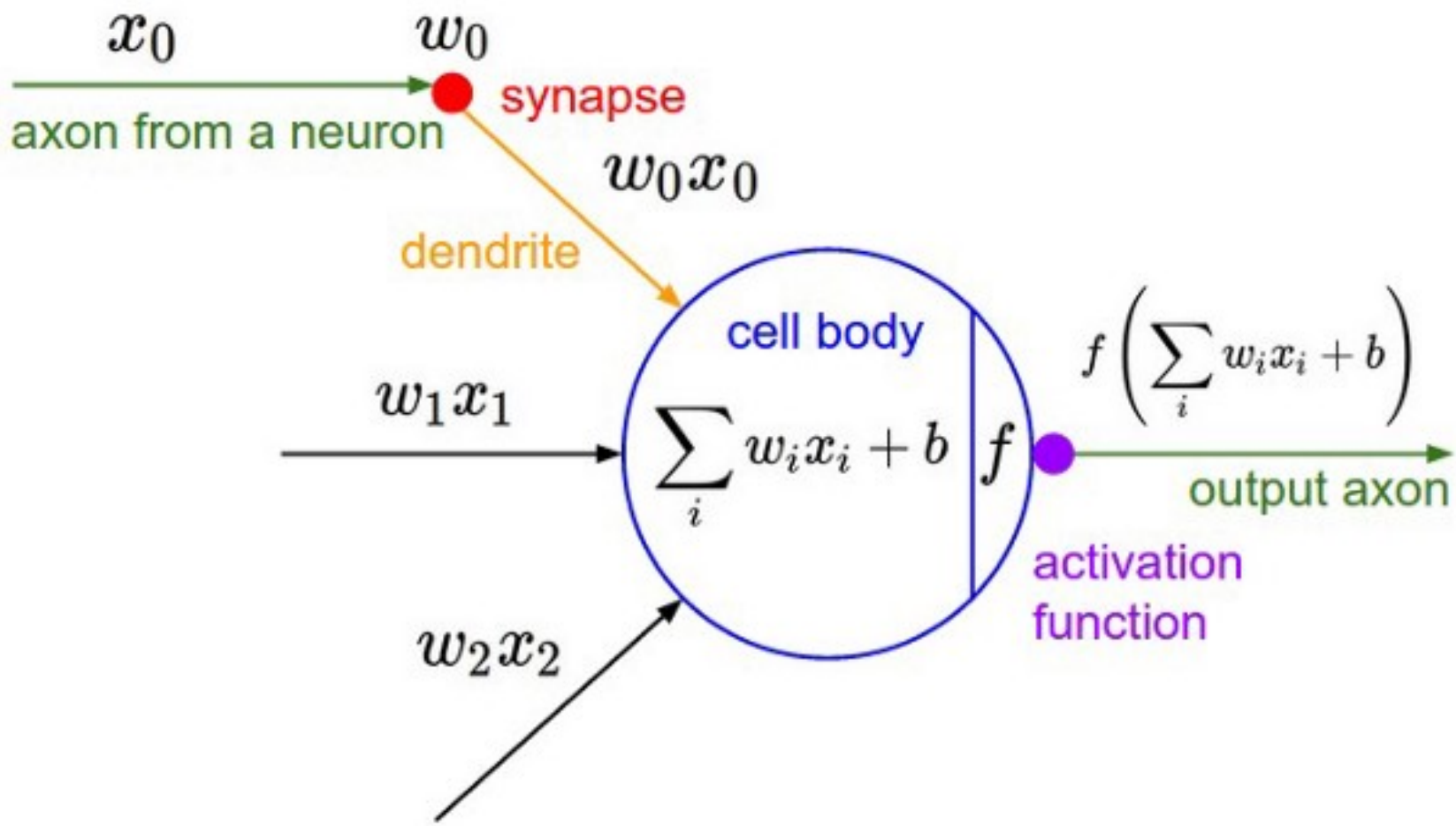


Figure: Computational neuron

Implementation

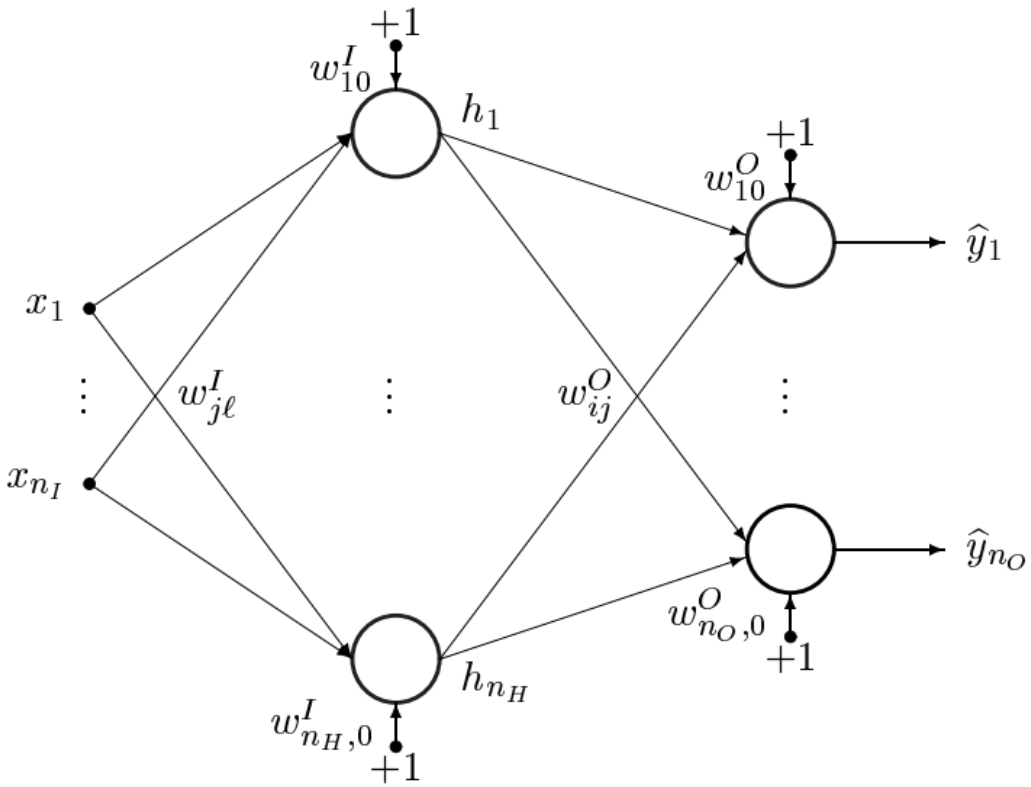


Figure: Simple two layer NN

Implementation

- Graphical representation of layered computation

$$h_j(x) = \psi\left(\sum_{i=1}^{n} wx + b\right) \tag{1}$$

- x, input vector
- w, weights
- b, bias
- ψ, an activation function

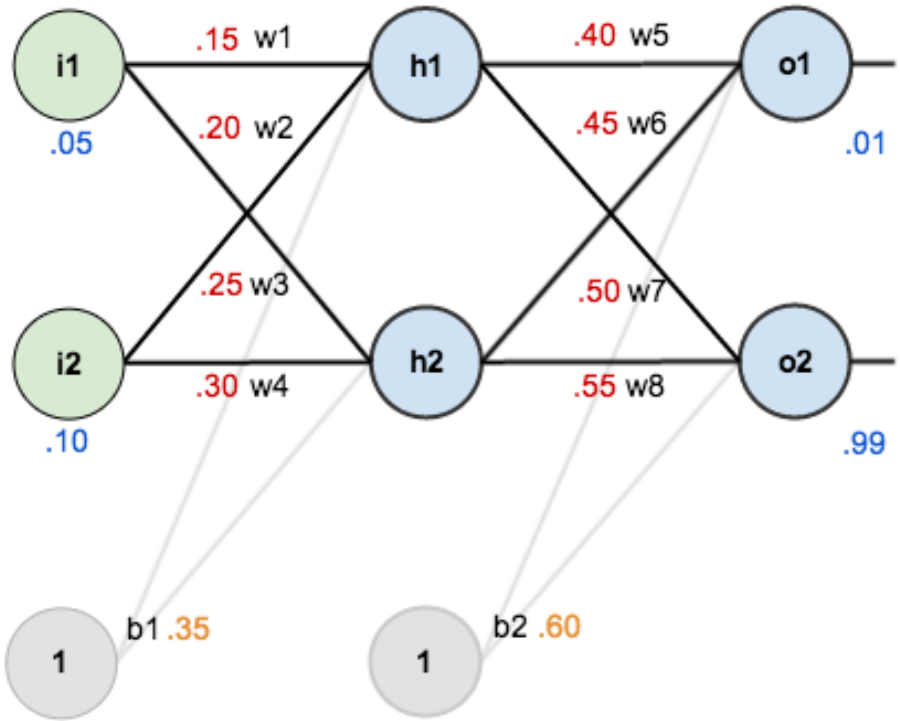
Implementation

- Learn weights w to minimize a cost function, C
- Use backpropagation
- Assumption: Cost function can be written as average $C_x = \sum_x C_x$
- Assumption: Can be written as function of outputs from neural network

Implementation

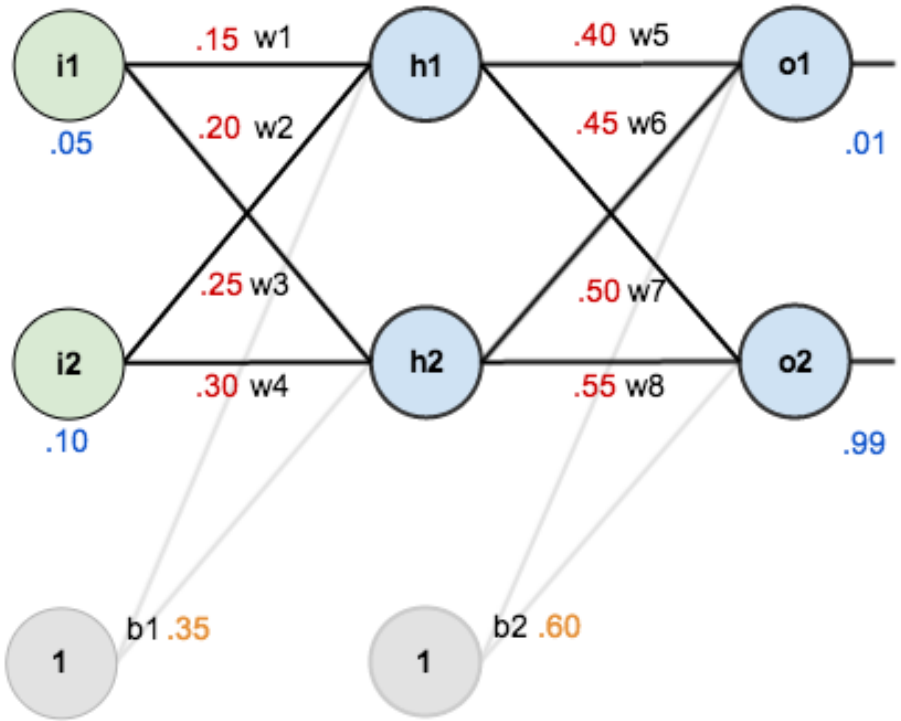
- Initialize weights and biases with some value.
- Start forward pass

Implementation



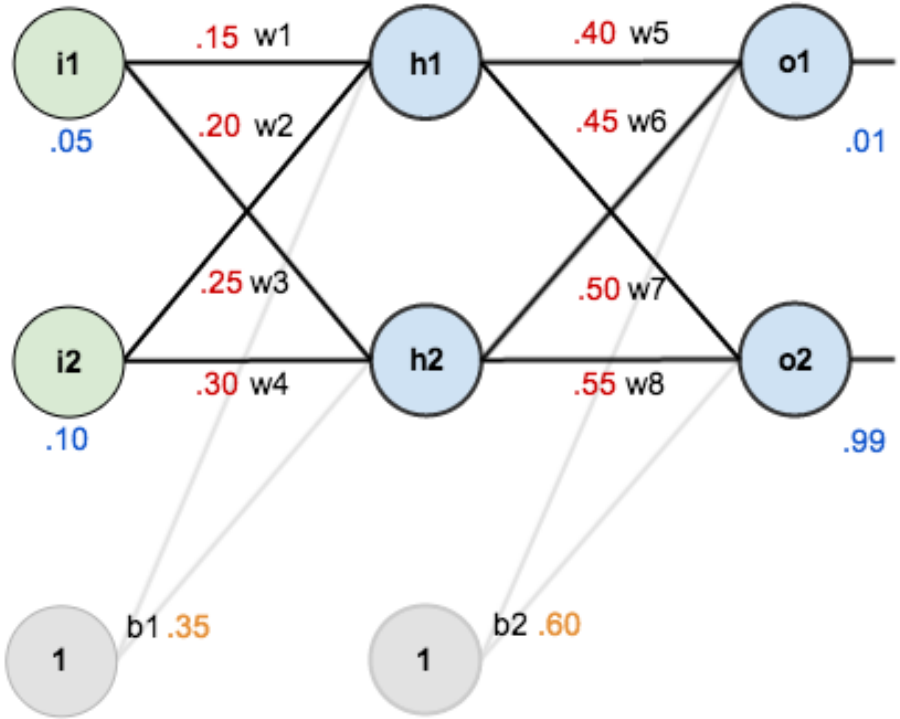
$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

Implementation



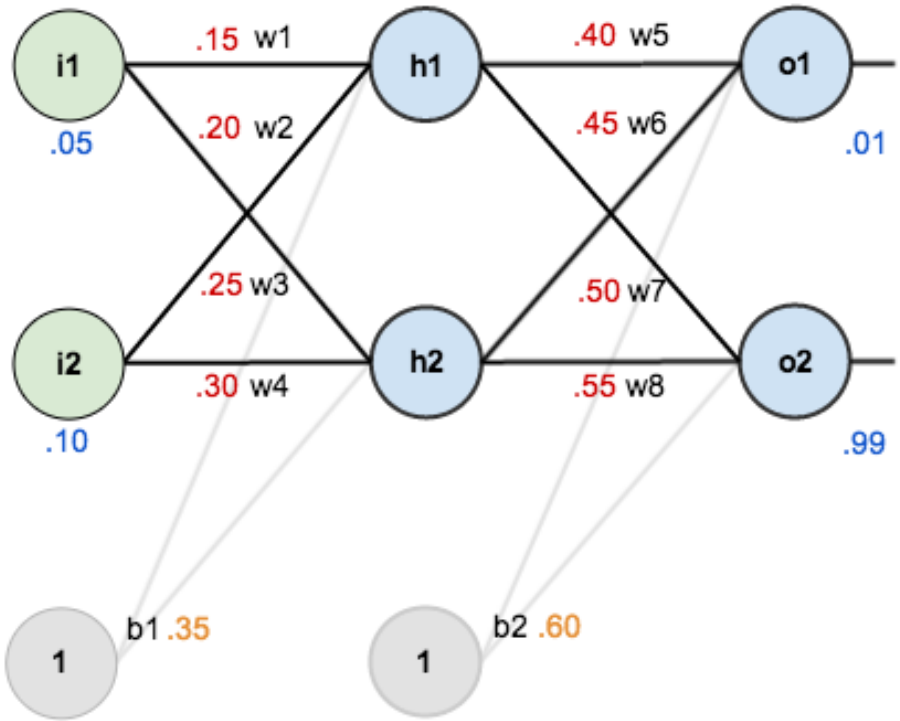
$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

Implementation



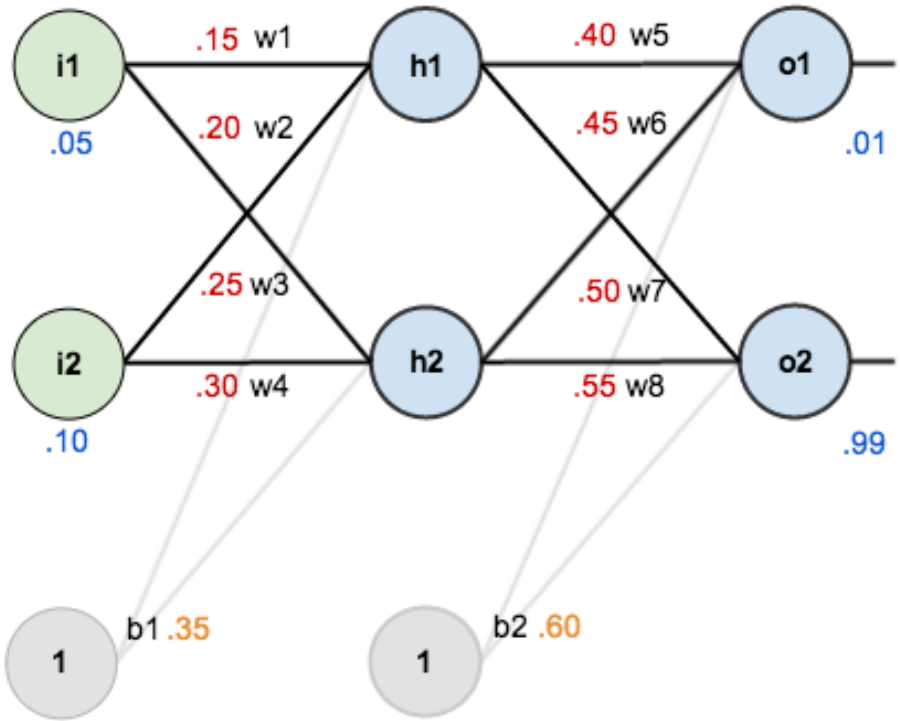
$$out_{h2} = 0.596884378$$

Implementation



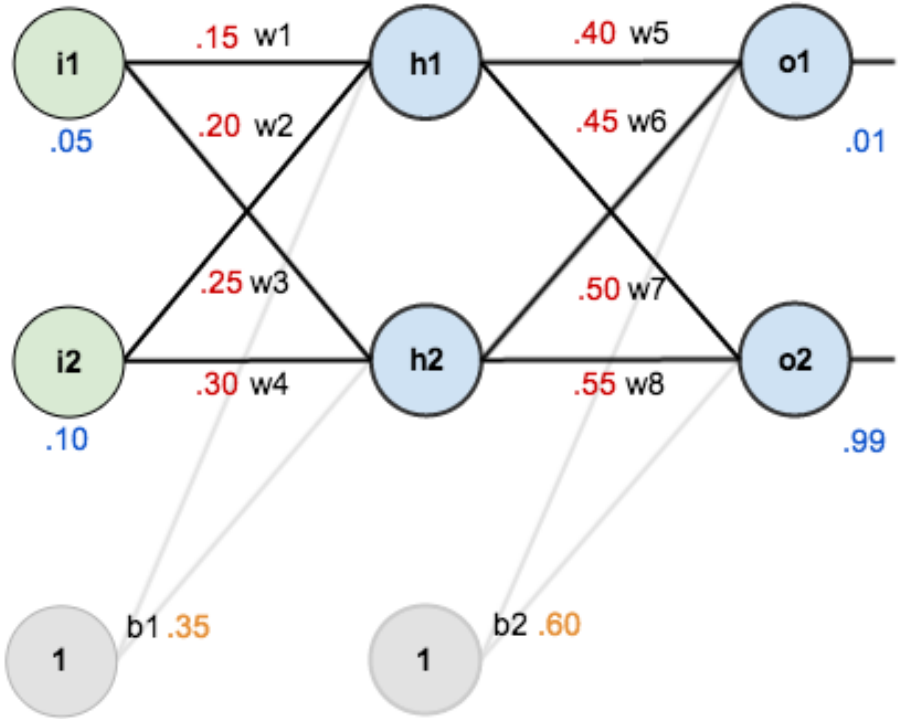
$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

Implementation



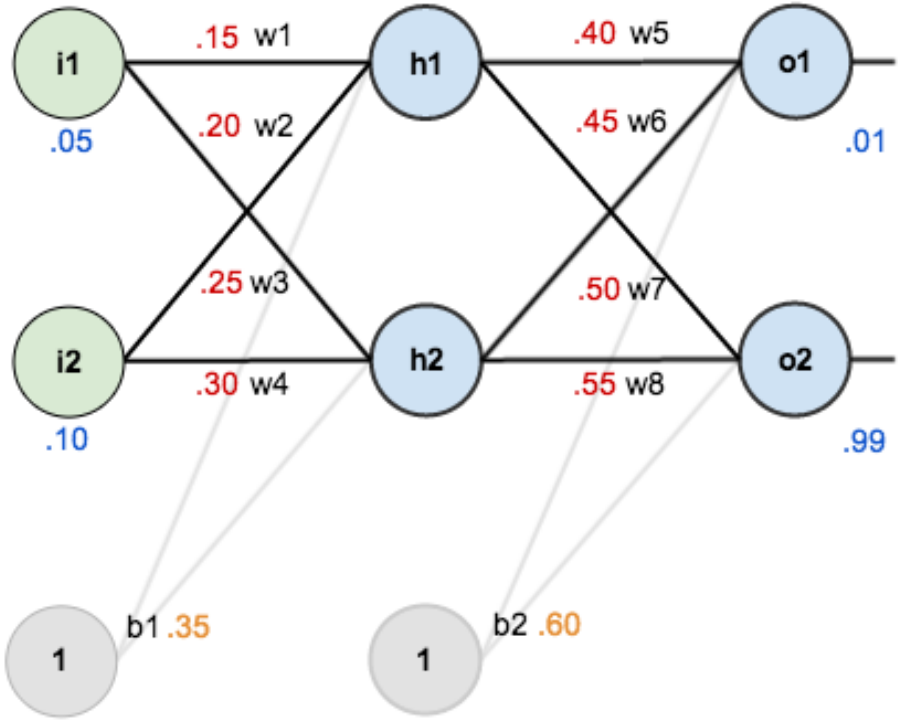
$$out_{o2} = 0.772928465$$

Implementation



$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Implementation



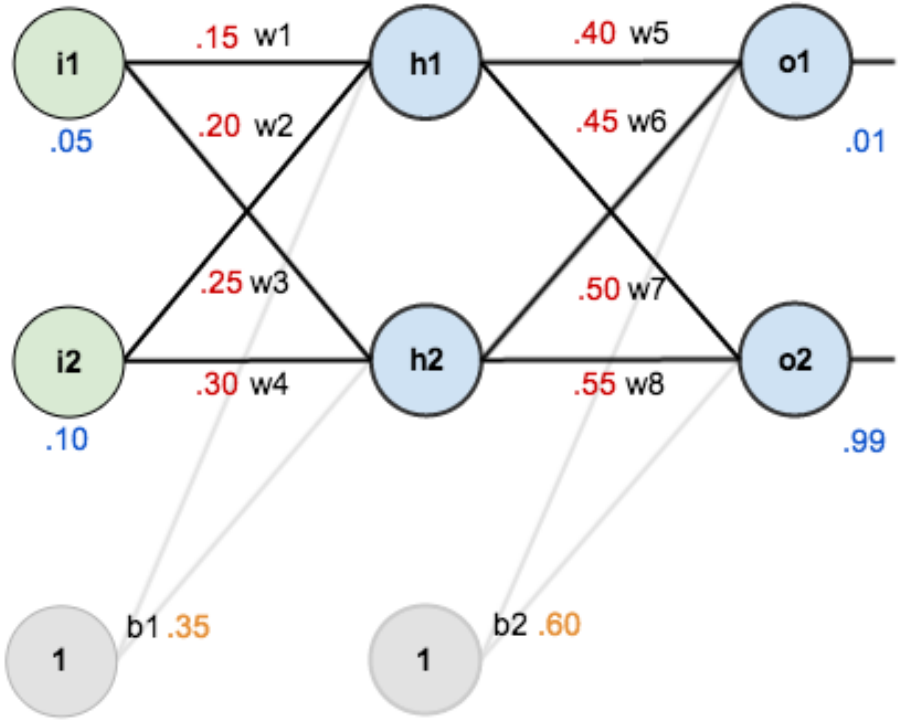
$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

Implementation

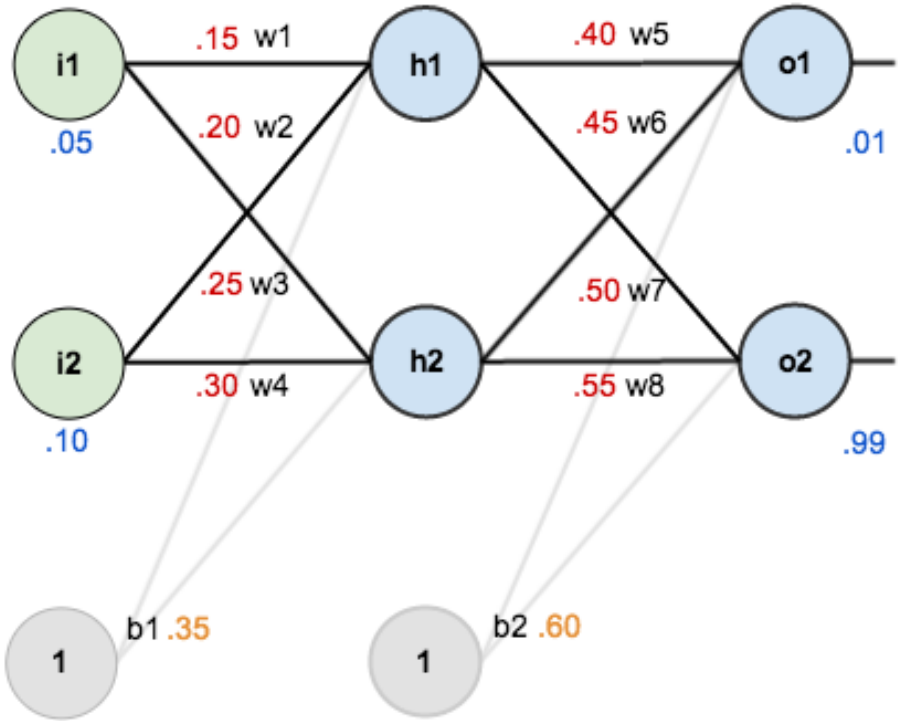
- Start backward pass
- Goal: Update each of the weights in the network so the model output is closer to the real output.
- Minimize error for each neuron and network as a whole
- We want to know how much change in a single weight affects total error

Implementation



$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

Implementation



$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

Implementation

- Neural networks are powerful
- Proven to be universal approximators

Implementation

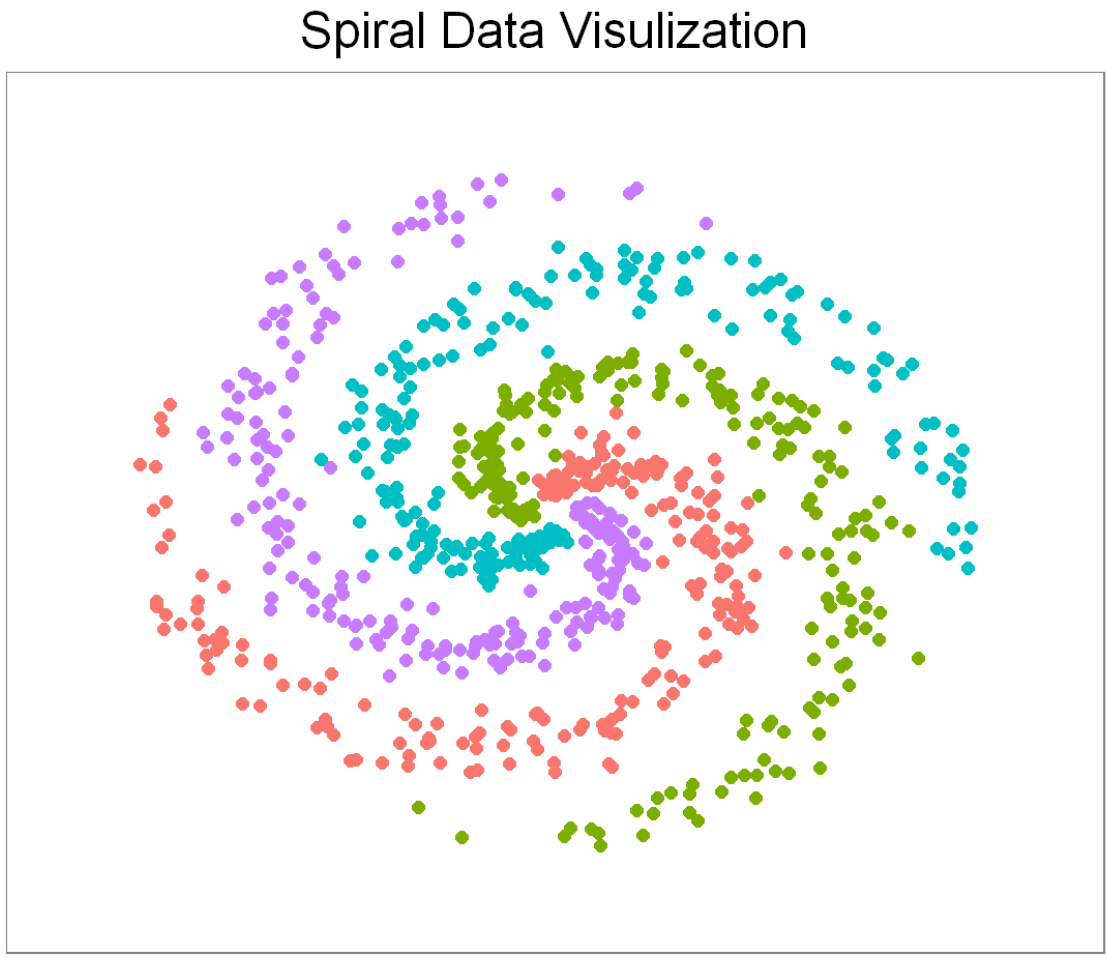


Figure: Spiral dataset

Implementation

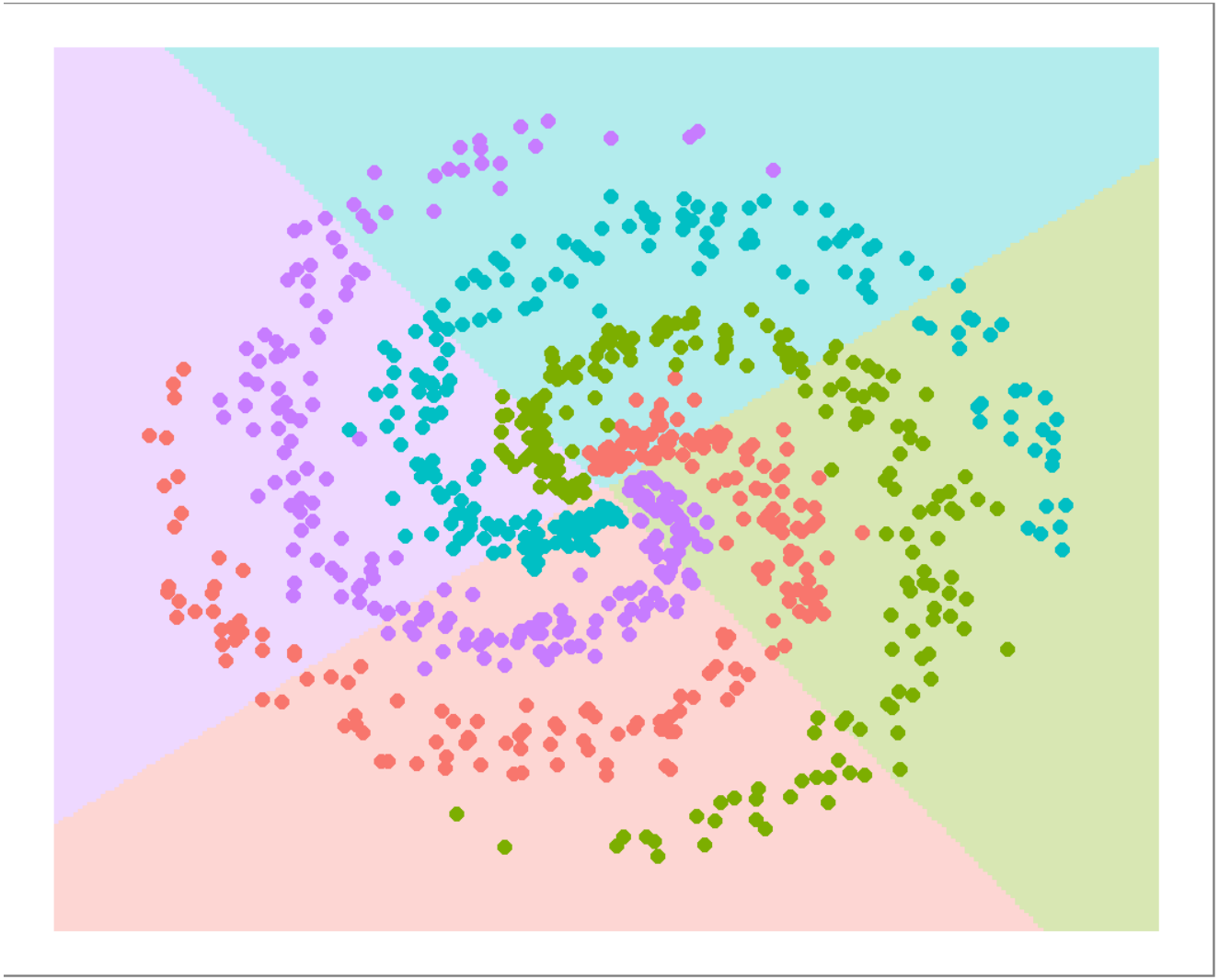


Figure: One layer NN, linear activation

Implementation

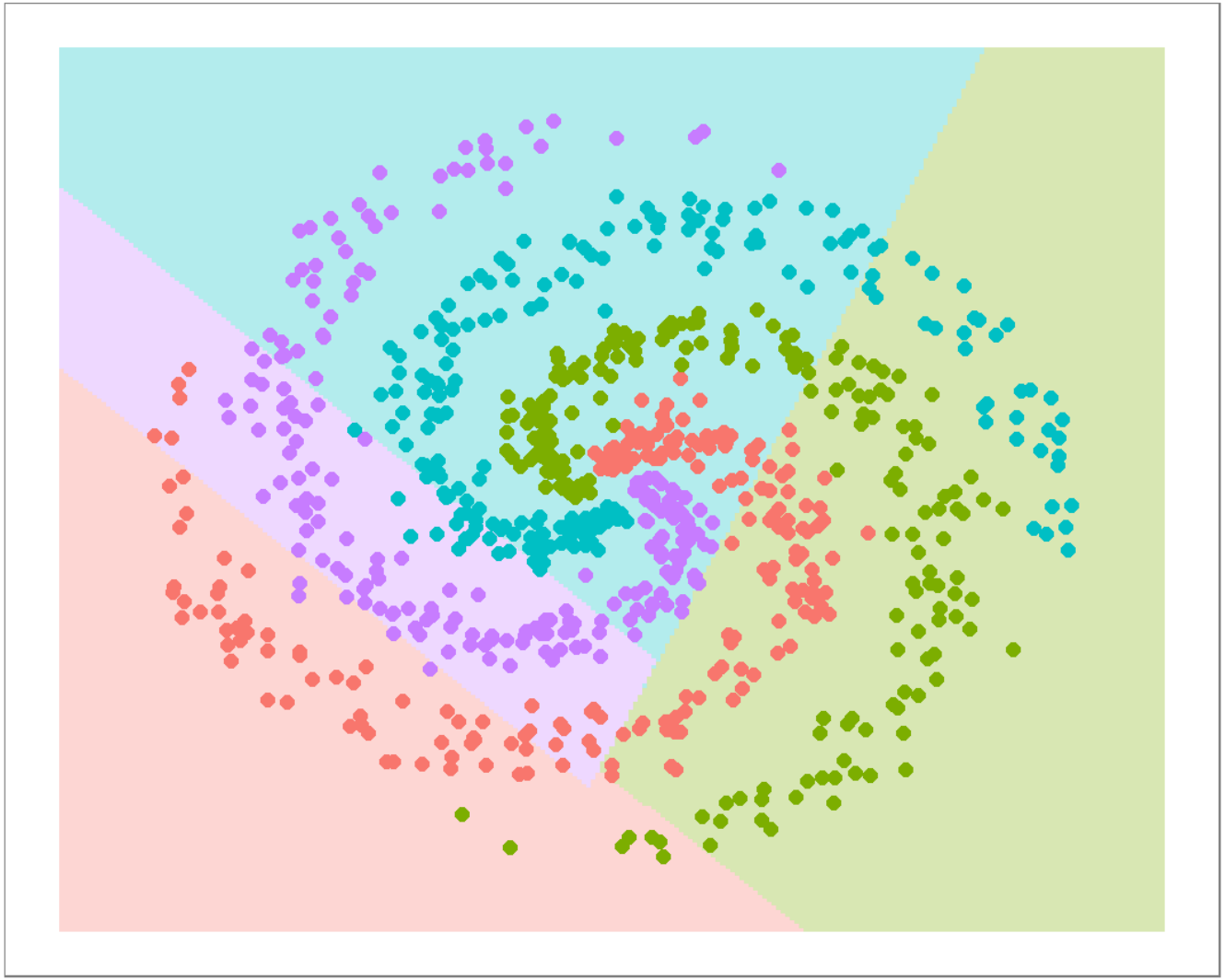


Figure: Two layer NN, linear activation

Implementation

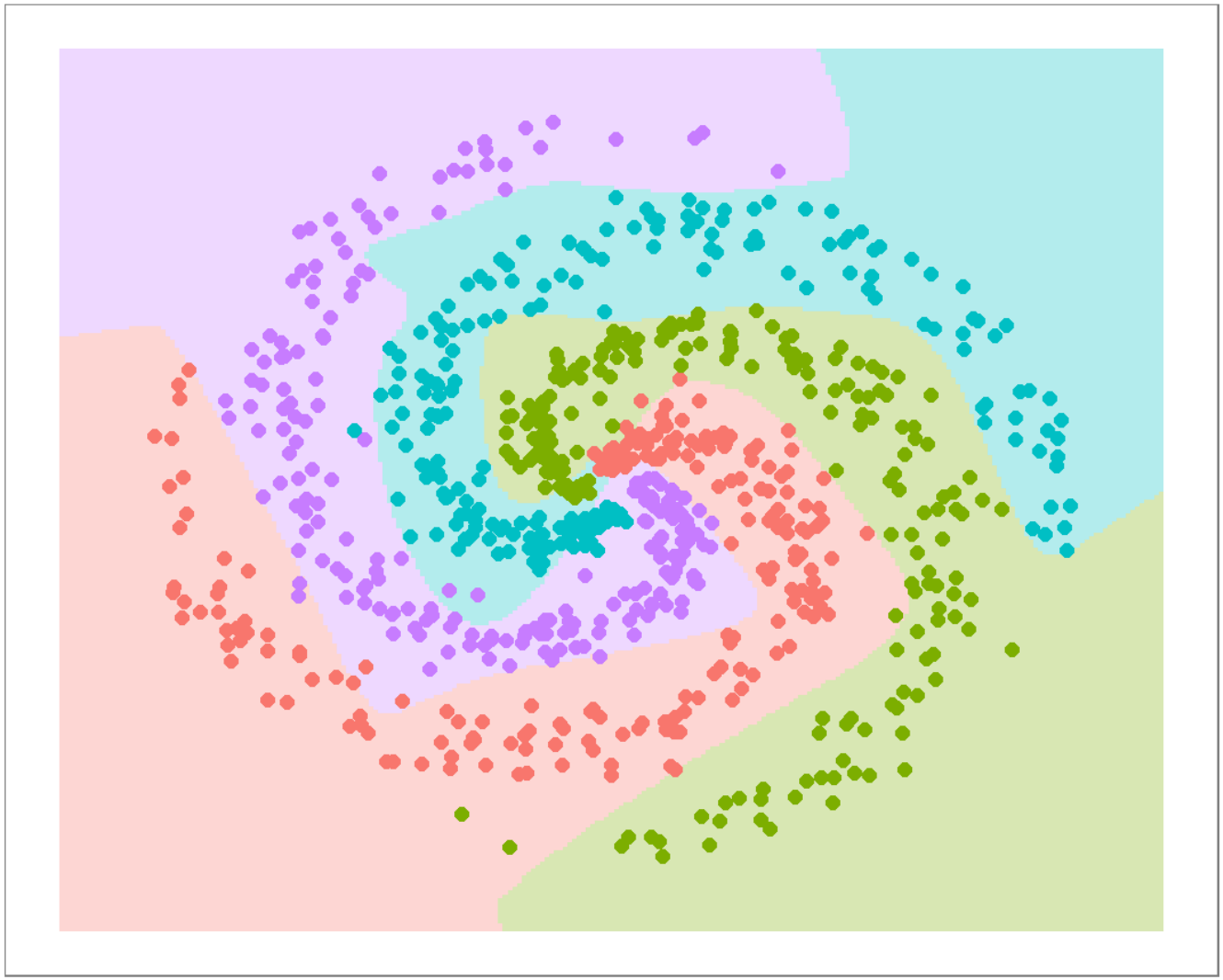


Figure: Ten layer NN, sigmoid activation

Activation functions

- Step (1 or 0)
- Sigmoid [0,1]
- Tanh [-1,1]
- Relu [0,inf]

References I

Backprop example taken from "https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/"