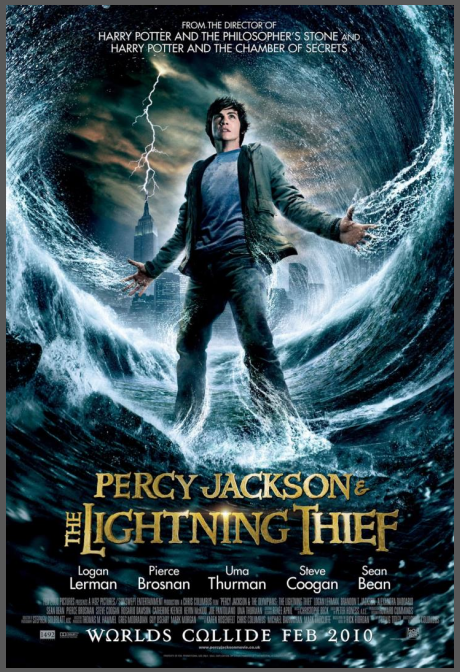
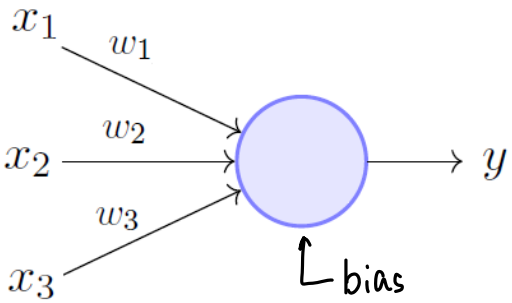
Discussion
Cingin' in the Rain



Percy-eptron



Perceptron Model (Minsky-Papert in 1969)

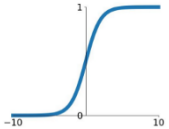
$$\text{Output} = \text{Activation}\left(w_1 x_1 + w_2 x_2 + w_3 x_3 + b\right)$$

(Bl)ac(k)-tivation Functions

Activation Functions

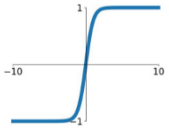
Sigmoid

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



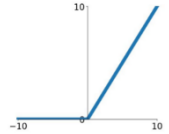
tanh

$\tanh(x)$



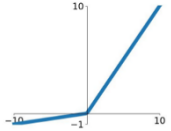
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

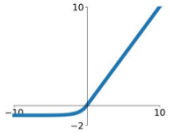


Maxout

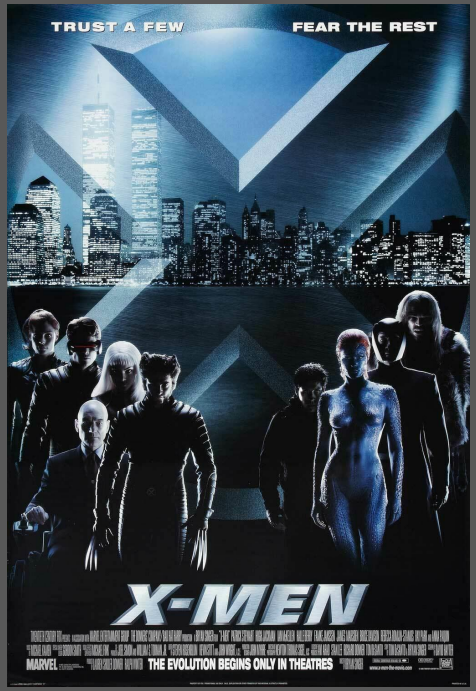
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Let’s work on an X-ample



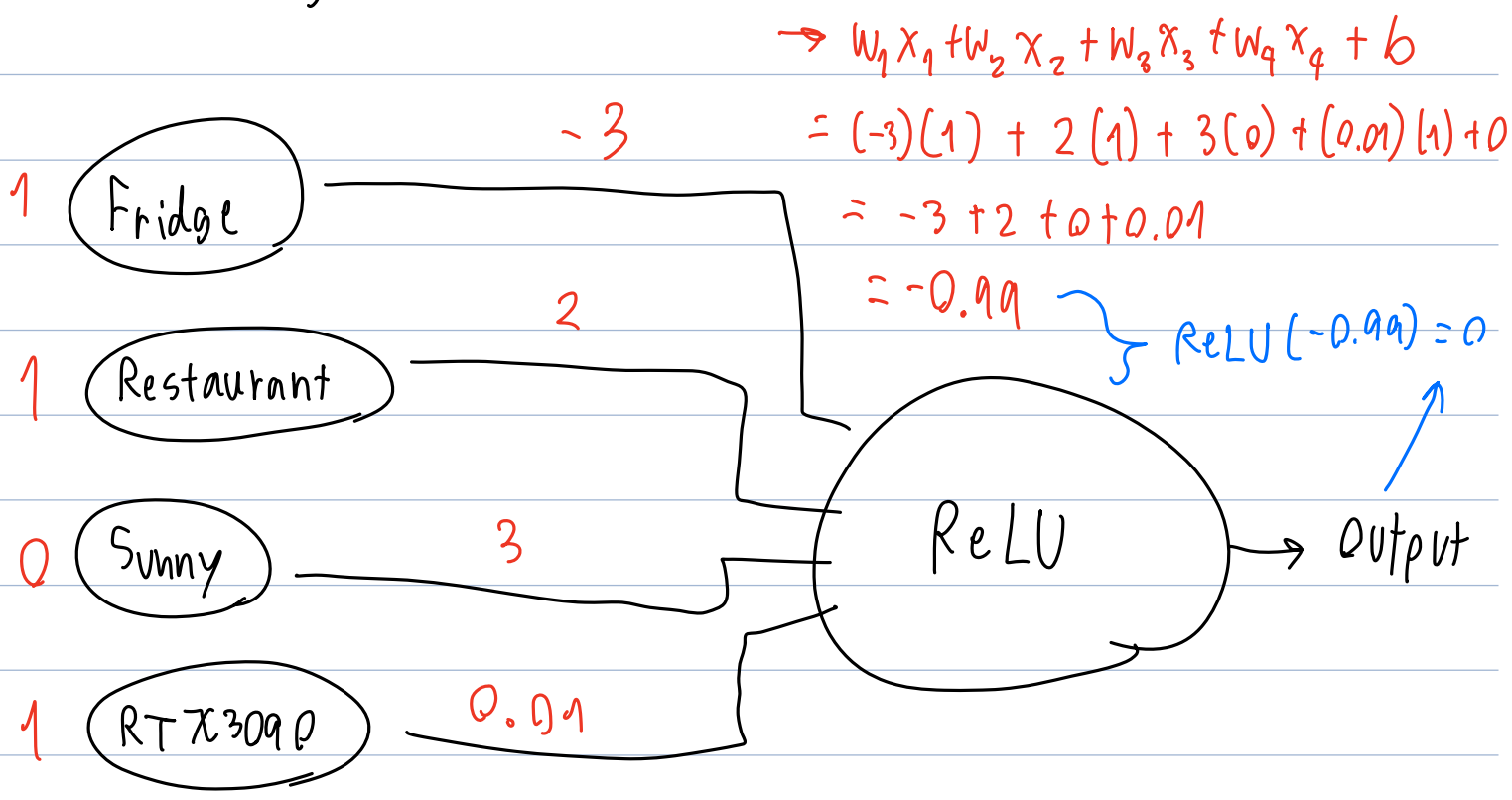
Ex Decide whether to eat outside.

1) Anything in the fridge? (1 = Yes)

2) Any restaurant nearby? (1 = Yes)

3) Is it sunny now? (1 = clear sky)

4) Using RTX3090?



Let bias = 0 in this case, and activation function is ReLU

→ Output = 0, so eat leftover food!

(A) New (Hope) Ral Network

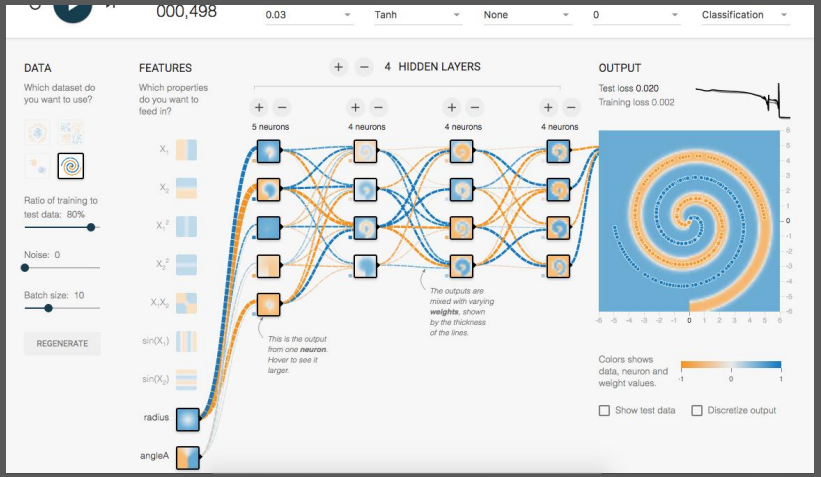
Long Definition (optional):
Artificial neural networks, usually simply called neural networks (NNs), are based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it.

Short Definition:
Neural Network is a bunch of perceptron connected together

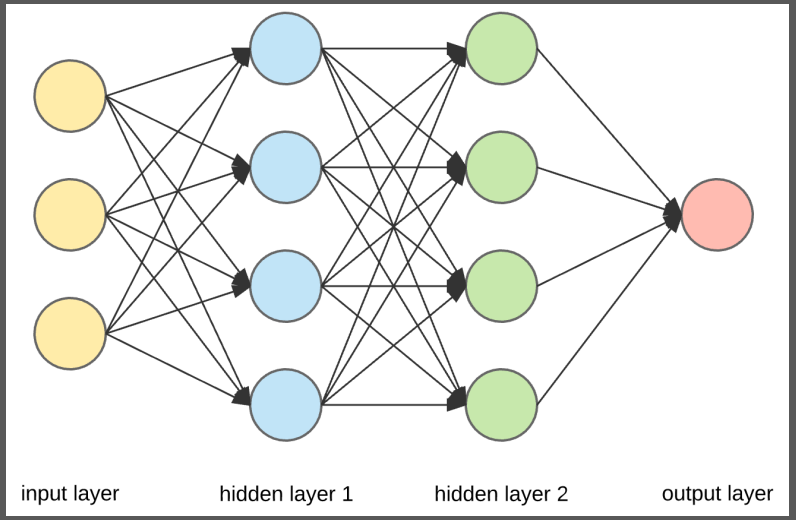
Motivation:

The perceptron is not complex enough to classify non-linear data. We then introduced neural network which consists of many perceptron to handle non-linear classification problem.

https://playground.tensorflow.org/



Definition & Terminology

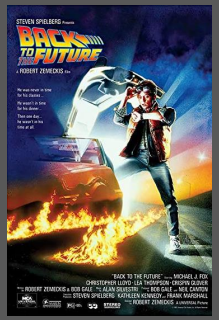


input

hidden

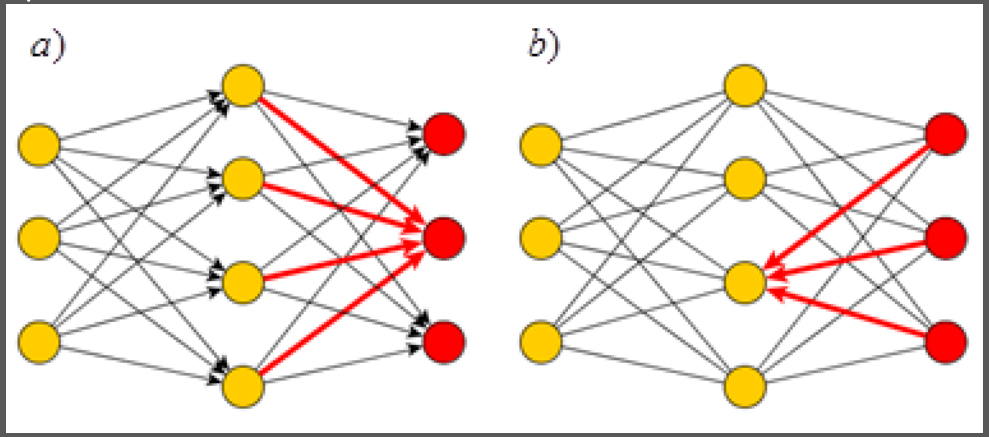
output

(Demon S)Layers

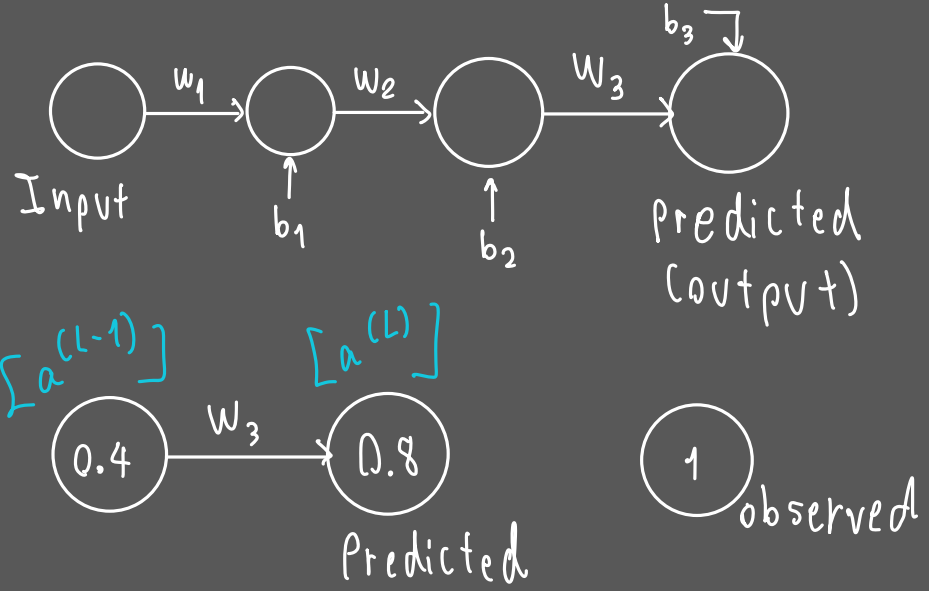


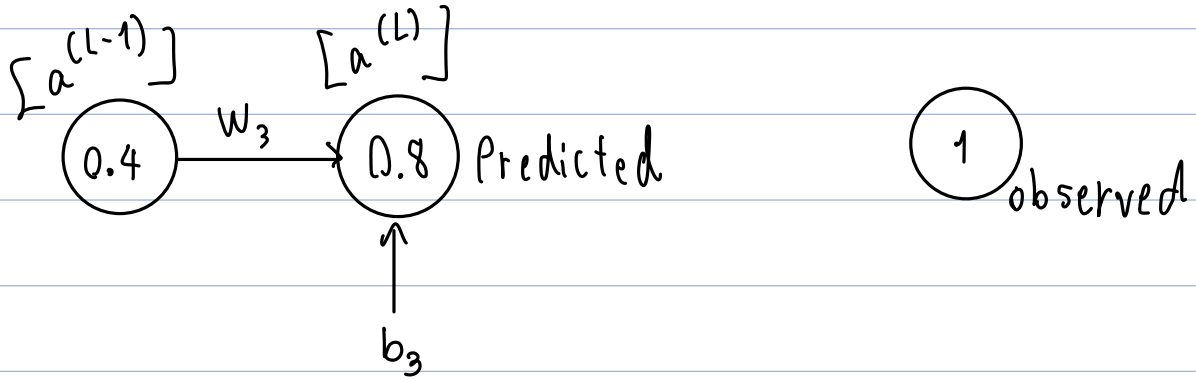
Back Propagation (backpropagation)

// given the cost function, we use chain rule to update each weight & bias from output layer to input layer



(Django Un)chain(ed) rule is backpropagation backbone





$$\text{Predicted} = \text{activation}^{(L)}\left(w^{(L)} \cdot \text{activation}^{(L-1)} + b^{(L)}\right)$$

$$\text{Cost} = (\text{predicted} - \text{observed})^2$$

Gradient Descent : $w_{3new} = w_{3old} - \lambda \frac{\partial(\text{cost})}{\partial w_3}$

That's The End (of Evangelion) for today