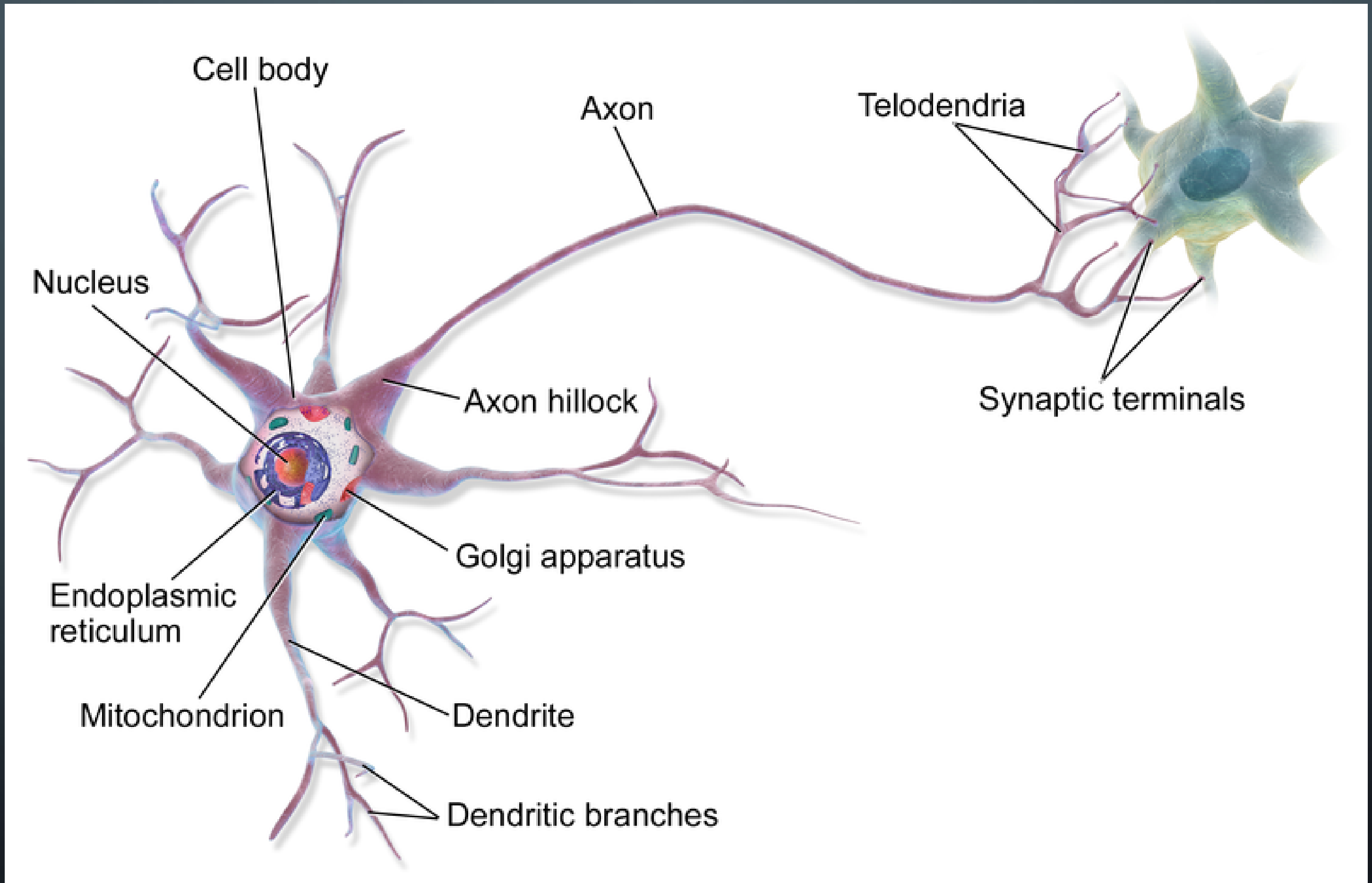
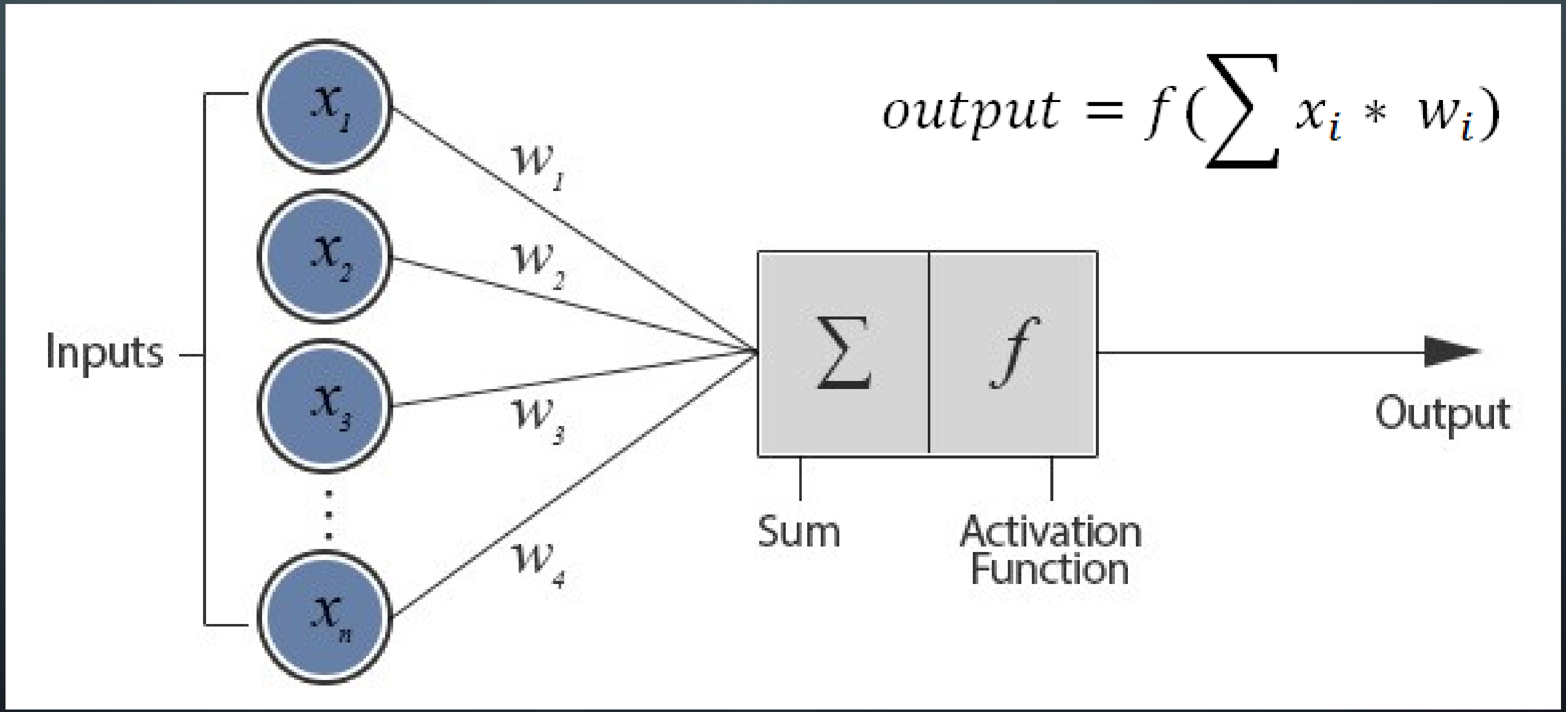
# NEURAL NETWORKS

MOHAMMAD GHODDOSI

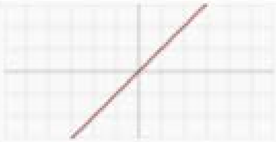
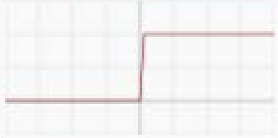
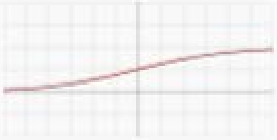
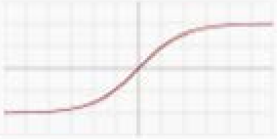
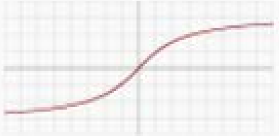
# BIOLOGICAL NEURON

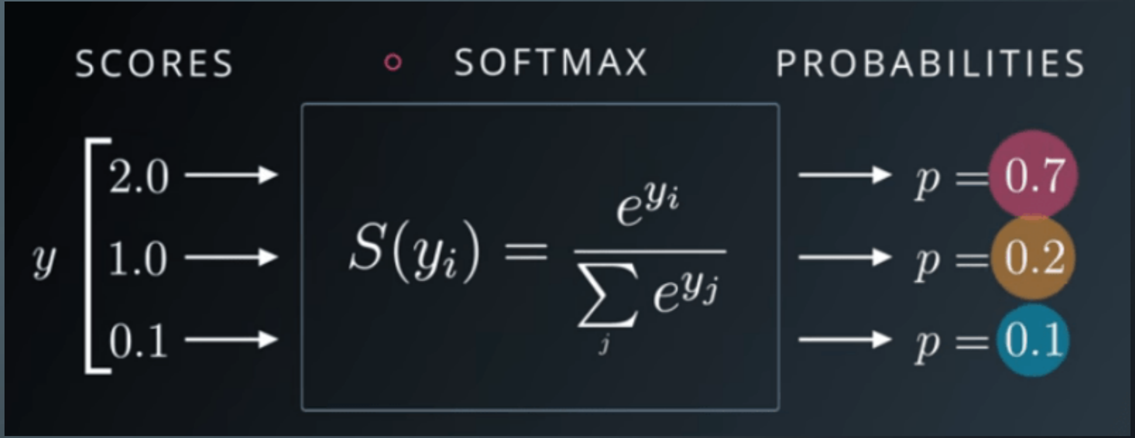


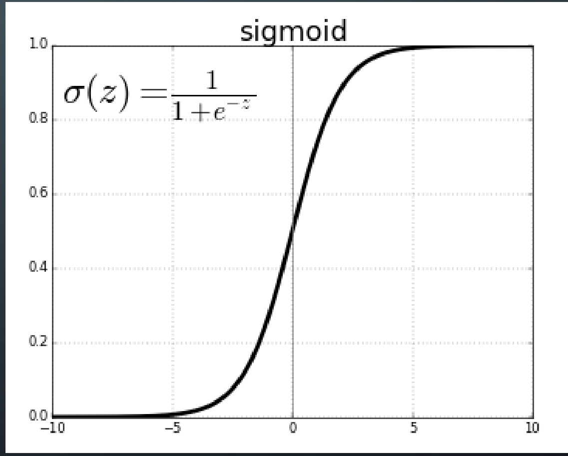
# COMPUTATIONAL NEURON

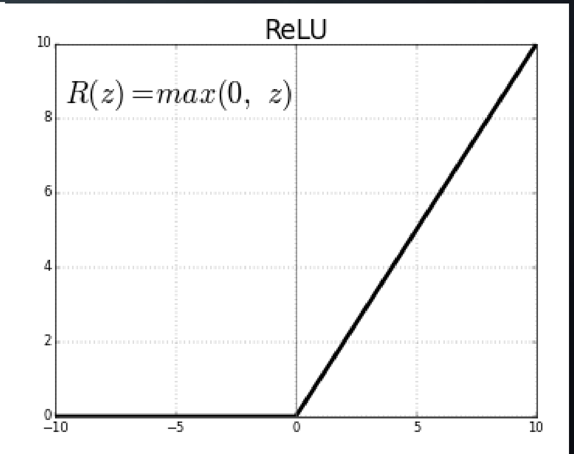


# ACTIVATION FUNCTIONS

| Name | Plot | Equation |
| --- | --- | --- |
| Identity |  | $f(x) = x$ |
| Binary step |  | $f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$ |
| Logistic (a.k.a Soft step) |  | $f(x) = \frac{1}{1 + e^{-x}}$ |
| TanH |  | $f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$ |
| ArcTan |  | $f(x) = \tan^{-1}(x)$ |



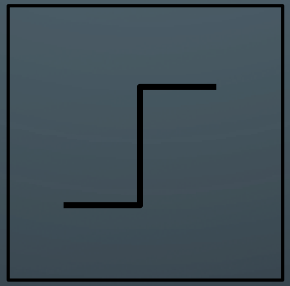



# PERCEPTRON

- Old algorithm (1958)
- Perceptron is a basic algorithm for neural networks
- Much like logistic regression
- linear problems
- Hebbian learning rule

# PERCEPTRON OUTPUT

- Activation function = step function
  - $f(z) = \begin{cases} 1 & z > 0 \\ -1 & z \leq 0 \end{cases}$



# PERCEPTRON LEARNING RULE

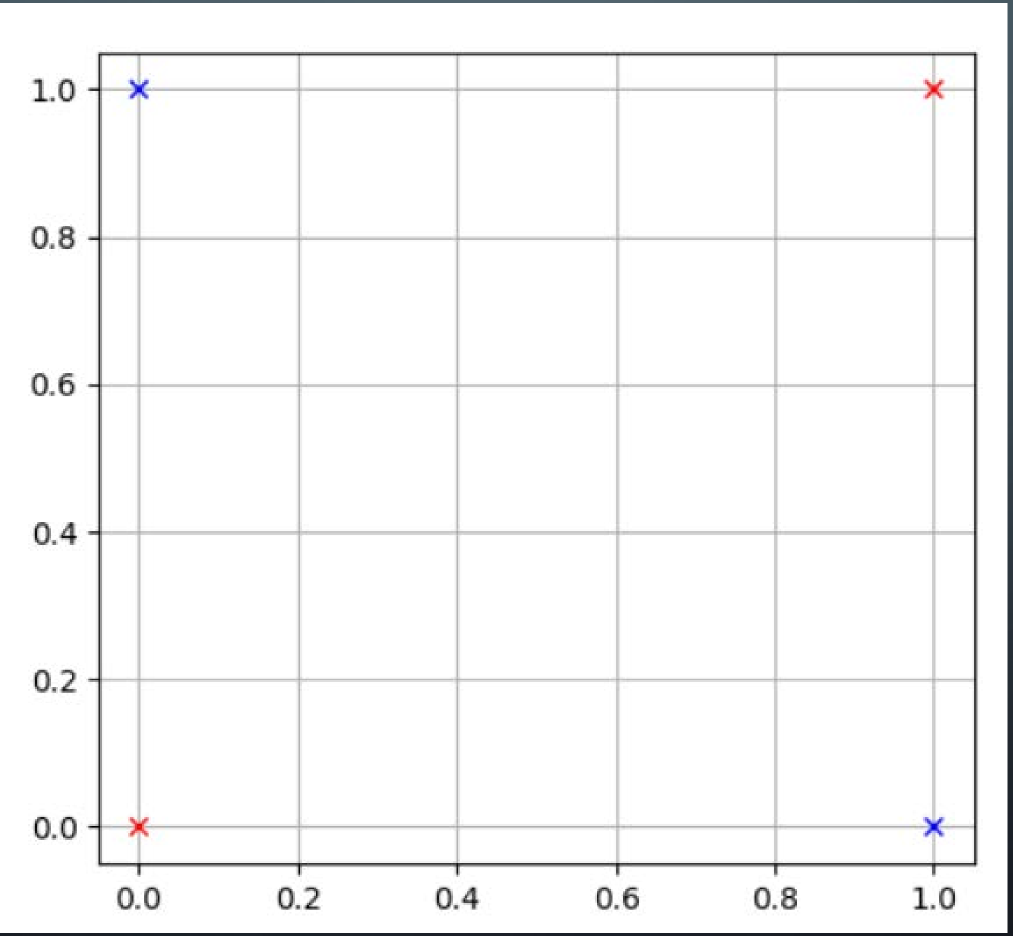
- Learning rate: $\alpha=1$
- Perceptron input: $x^{(i)}$ and label: $y^{(i)}$
- Perceptron output: $h^{(i)} = f(z^{(i)})$
- If $y^{(i)} = h^{(i)}$ then do nothing
- Else
  - If $y^{(i)} = -1 \; and \; h^{(i)} = 1$ then $w(t+1) = w(t) - x^{(i)}$
  - If $y^{(i)} = 1 \; and \; h^{(i)} = -1$ then $w(t+1) = w(t) + x^{(i)}$

# PERCEPTRON LEARNING RULE

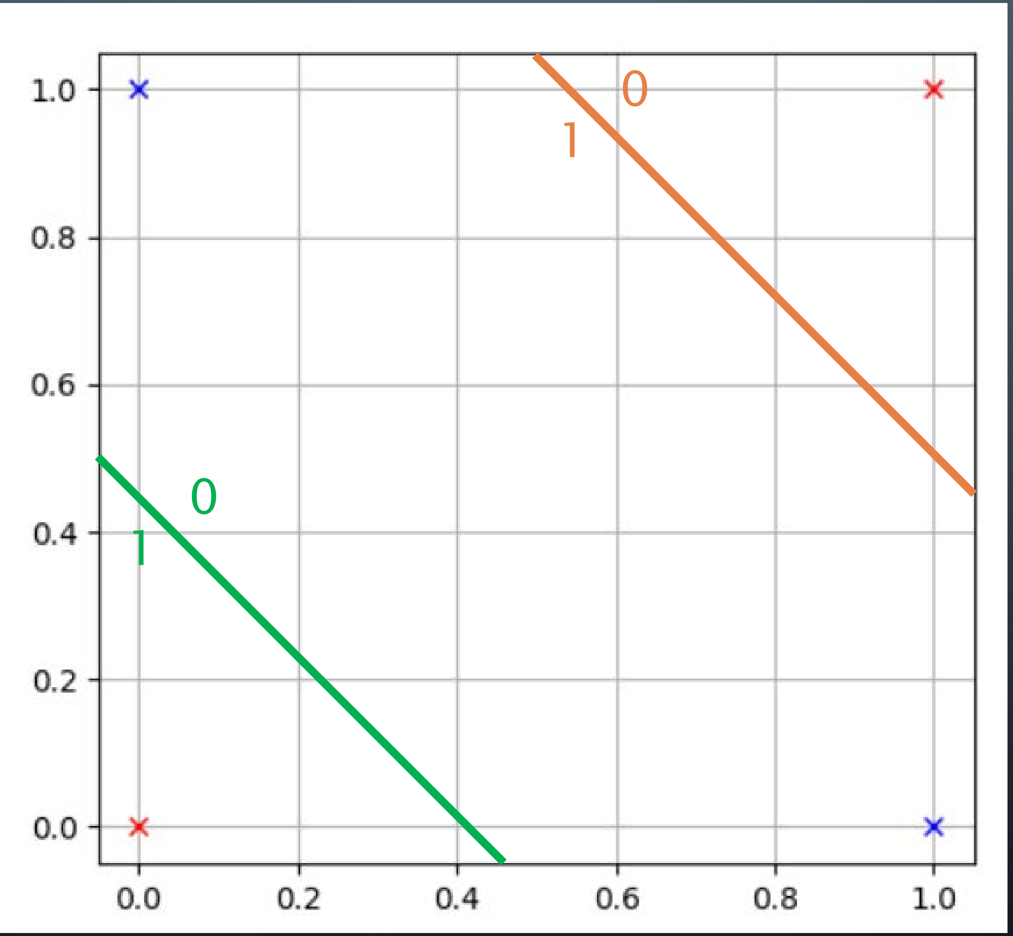
- Learning rate: $\alpha = 1$
- Perceptron input: $x^{(i)}$ and label: $y^{(i)}$
- Perceptron output: $h^{(i)} = f(z^{(i)})$
- If $y^{(i)} = h^{(i)}$ then do nothing
- Else
  - If $y^{(i)} = -1$ *and* $h^{(i)} = 1$ then $w(t+1) = w(t) - x^{(i)}$
  - If $y^{(i)} = 1$ *and* $h^{(i)} = -1$ then $w(t+1) = w(t) + x^{(i)}$

$$\boxed{w(t+1) = w(t) + y^{(i)} x^{(i)}}$$

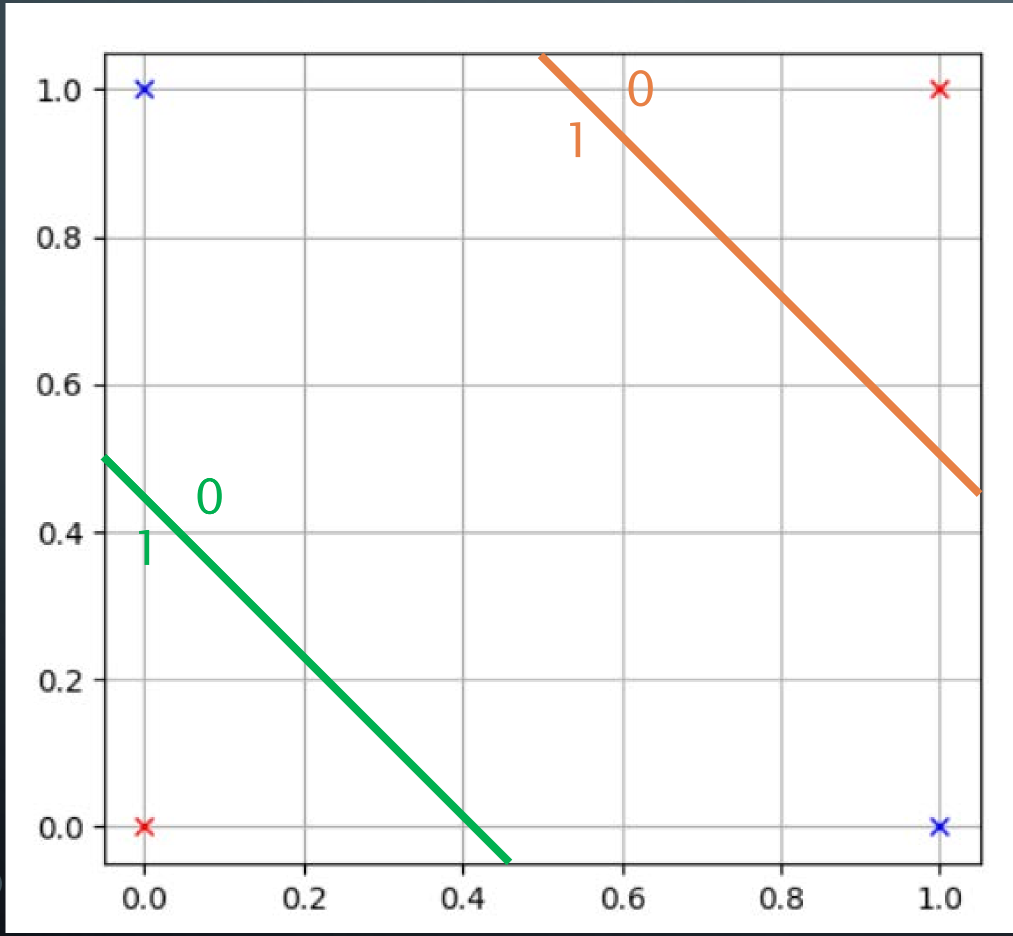
# PROBLEMS WITH PERCEPTRON

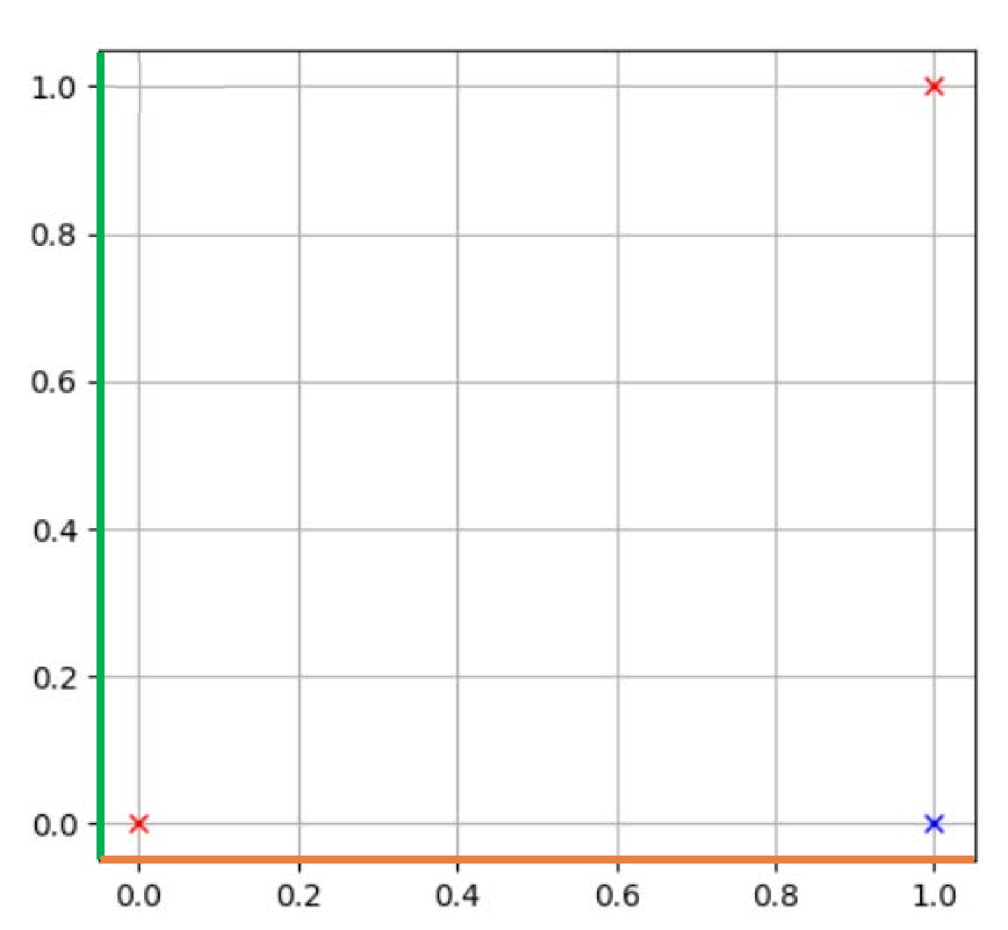


# PROBLEMS WITH PERCEPTRON

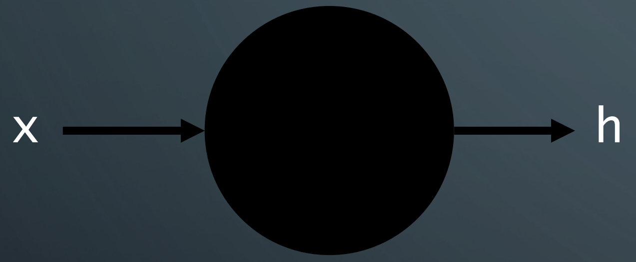


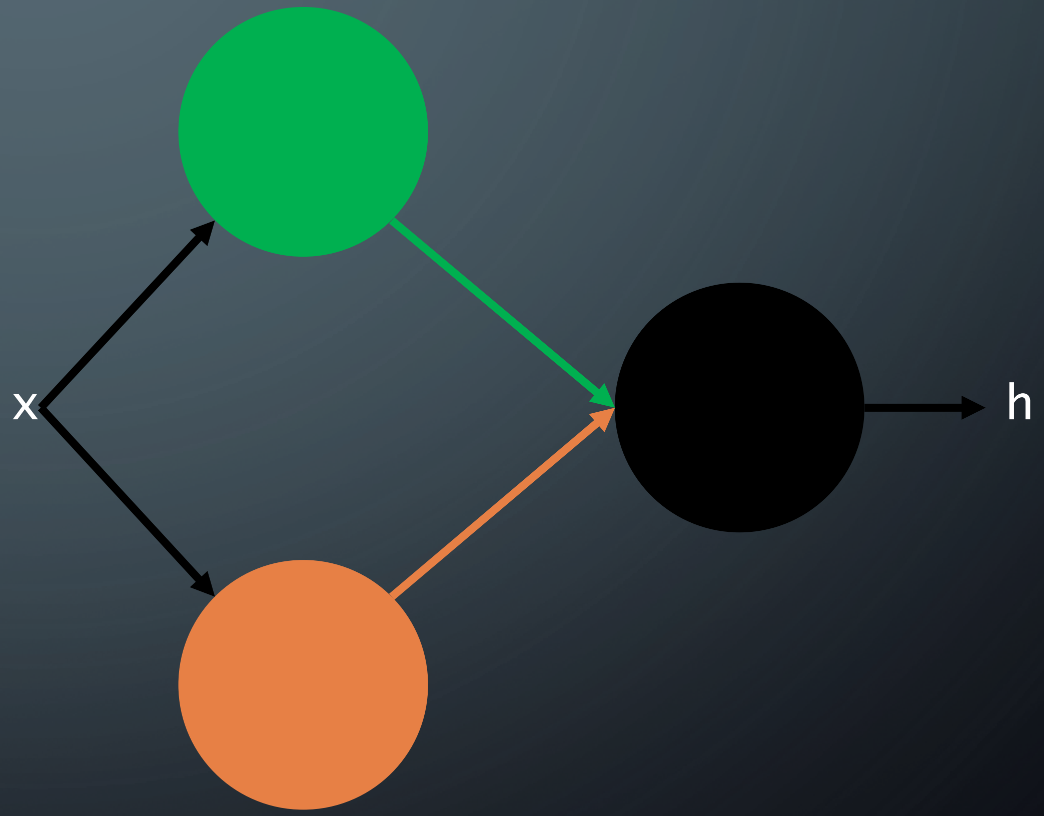
# PROBLEMS WITH PERCEPTRON



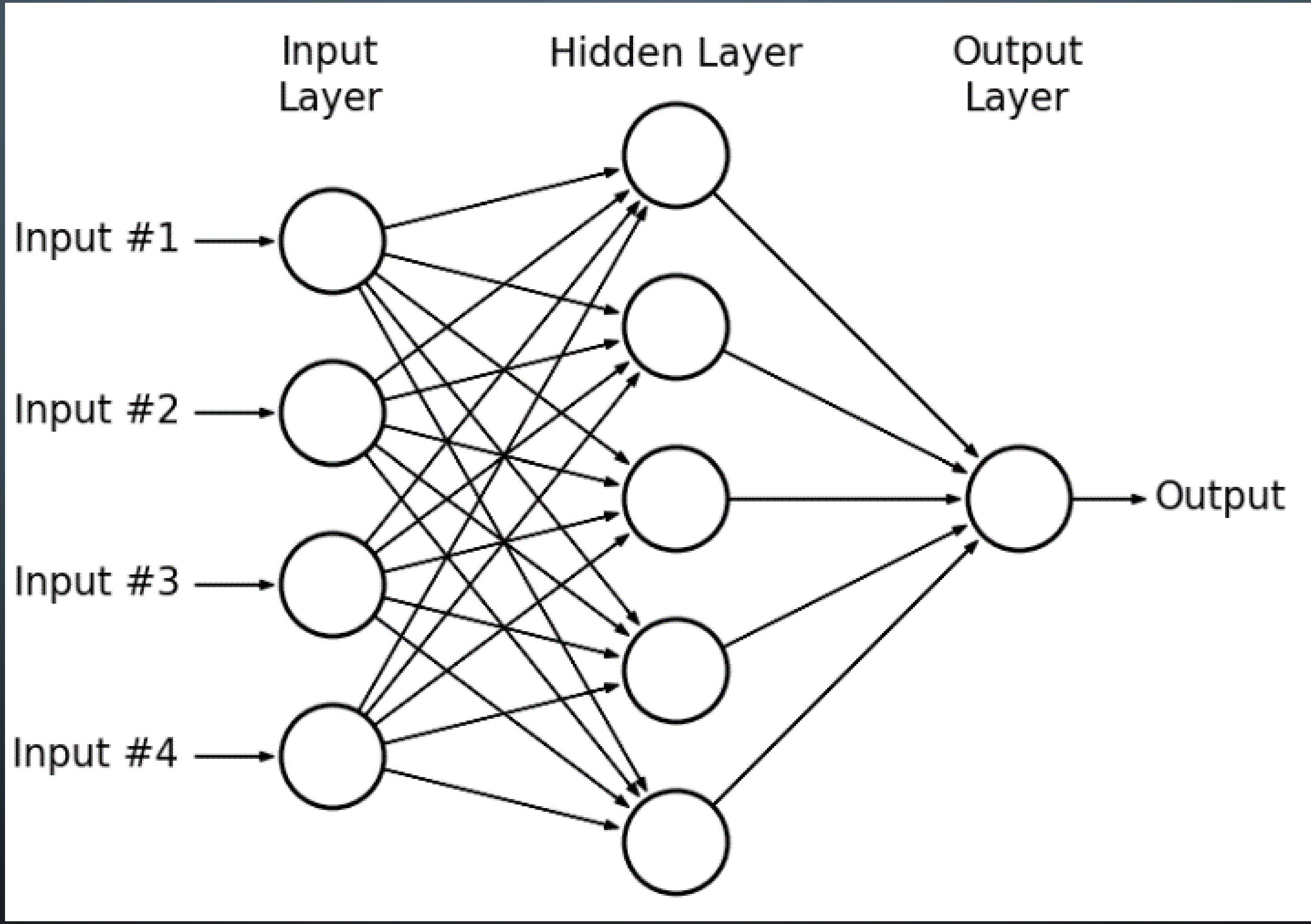



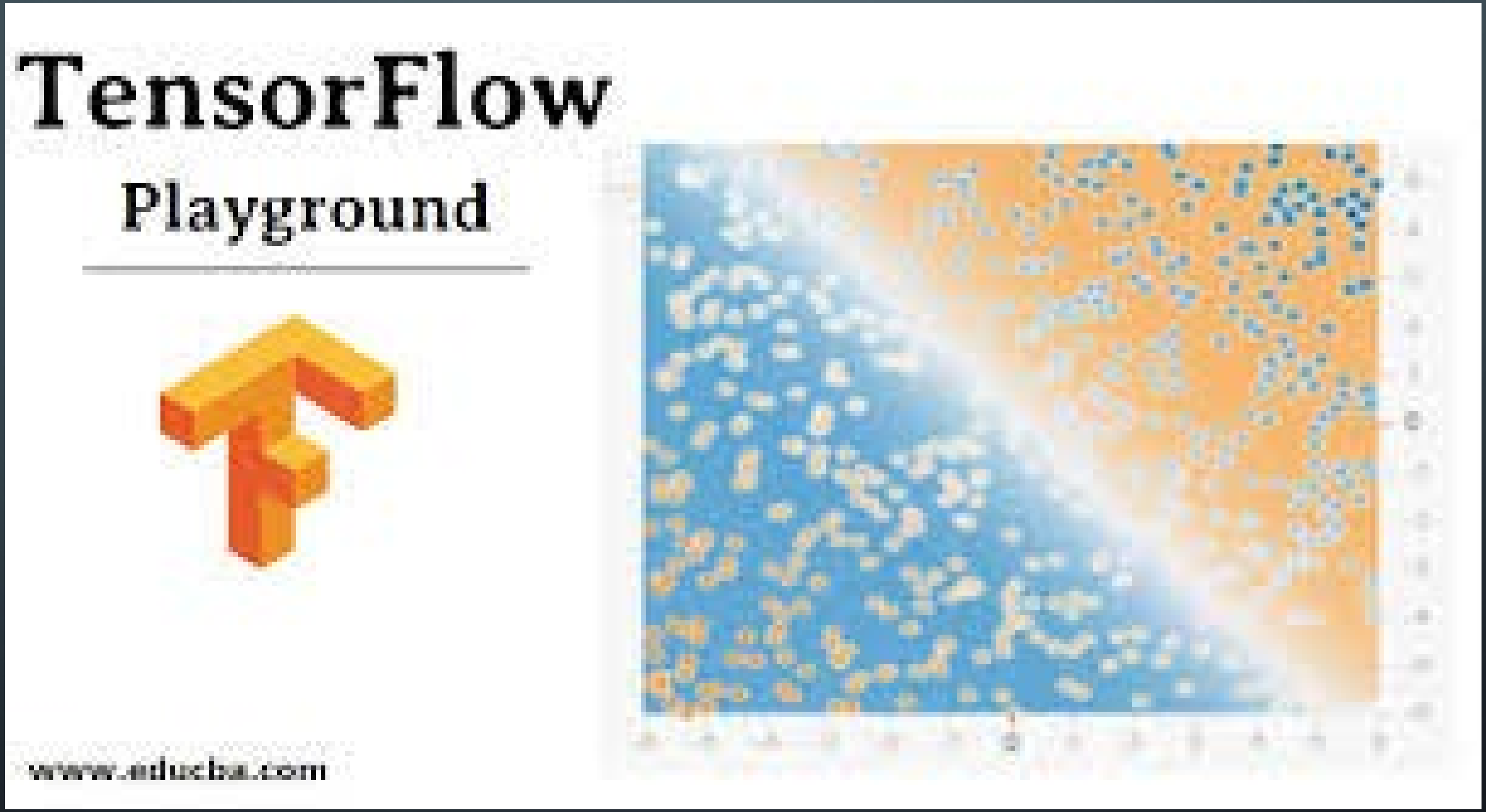
# PROBLEMS WITH PERCEPTRON



# MLP


Input Layer
Hidden Layer
Output Layer
Input #1
Input #2
Input #3
Input #4
Output


TensorFlow
Playground
www.educba.com

# MLP PROS AND CONS

- Pros
  - Flexible
  - Both regression and classification
  - Good for nonlinear data with large number of inputs
- Cons
  - black box
  - computationally very expensive and time consuming to train
  - depend a lot on training data
  - overfitting

# MLP ARCHITECTURE

- Single hidden layer is enough.
- If we have enough hidden units, we can solve every problem.
- Hidden units cant use linear (identity) activation function.

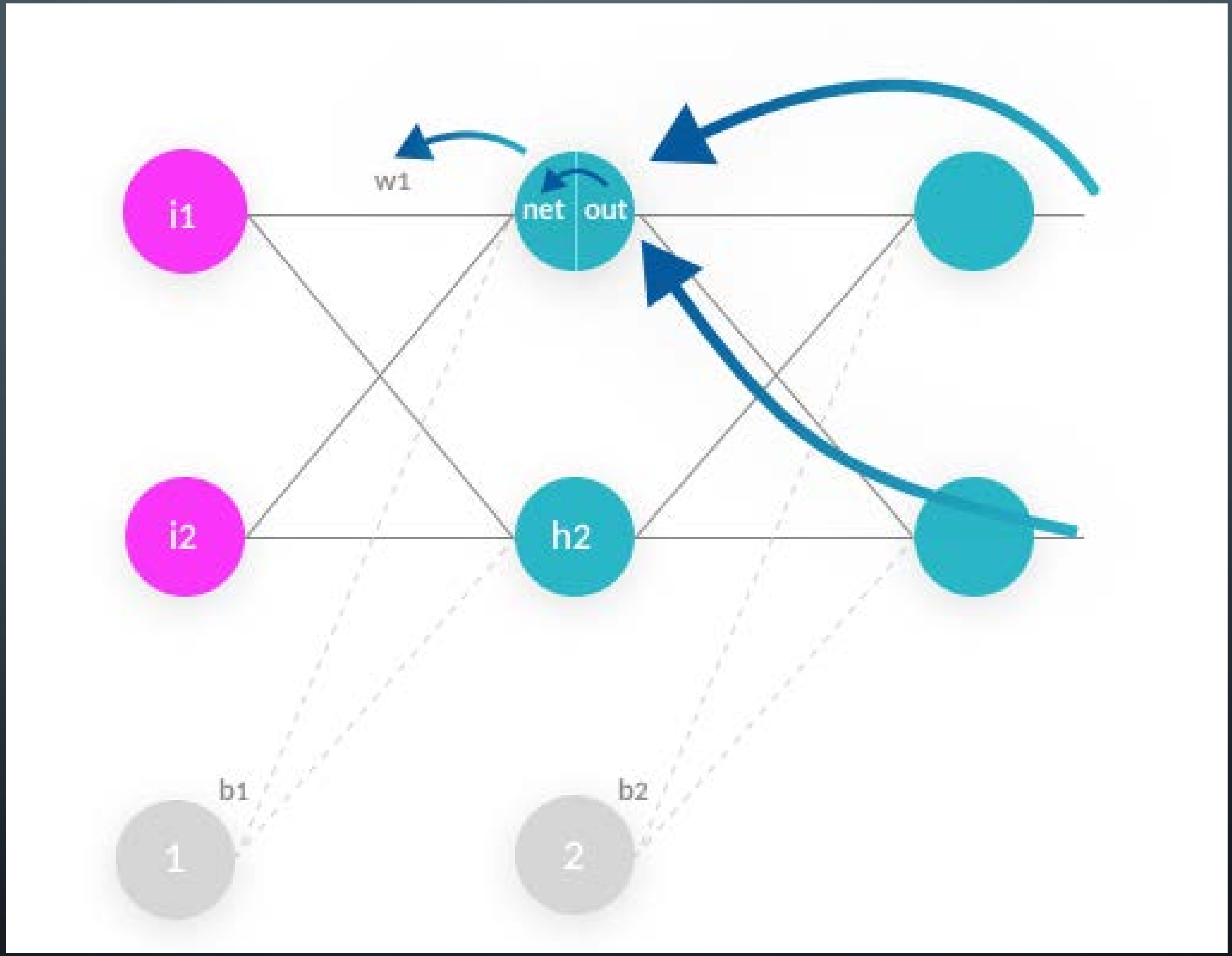
# SHALLOW VS DEEP NEURAL NETWORKS

- Shallow
  - Only one hidden layer
  - Simple neurons
- Deep
  - More than one hidden layer
  - Various types of neurons
    - Convolutional
    - Recurrent
    - …

# SHALLOW VS DEEP NEURAL NETWORKS

- deep NN with the right architectures achieve better results than shallow ones
- the deep models are able to extract/build better features than shallow models

# BACKPROPAGATION



# OPTIMIZERS

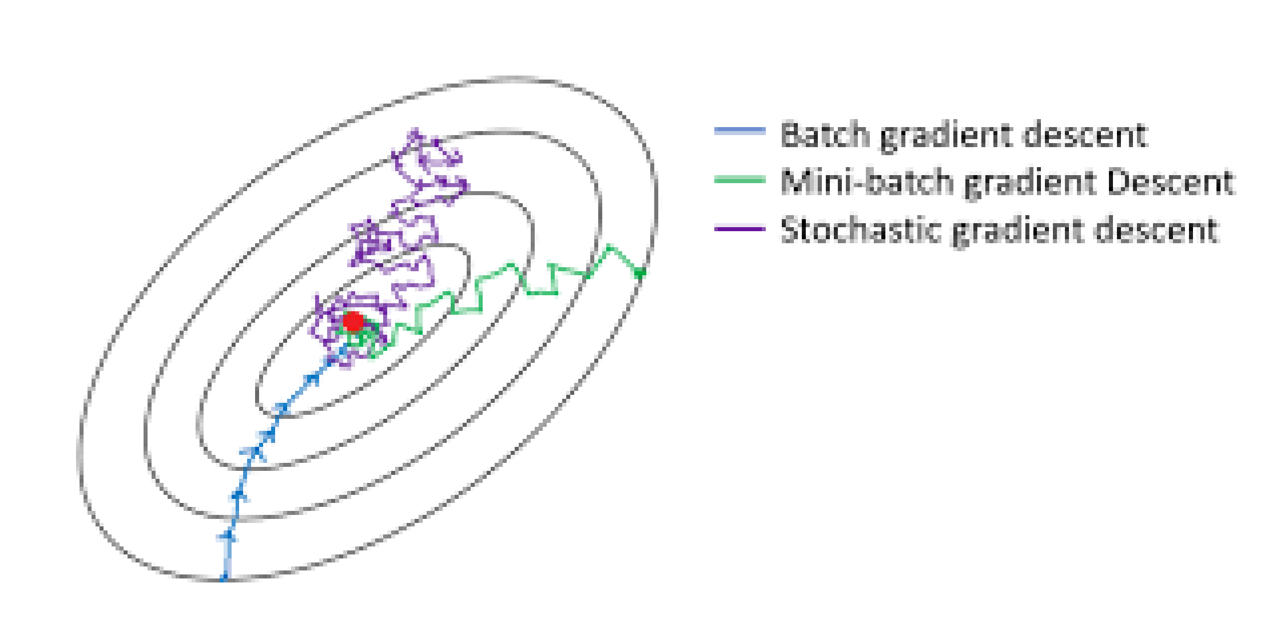
- Gradient descent
- SGD
- mini-batch GD
- Momentum
- AdaGrad
- AdaDelta
- RMSprop
- Adam

# STOCHASTIC GRADIENT DESCENT (SGD)

- Stochastic moves
- avoiding local minimum
- avoiding saddle points
- avoiding plateau
- Computational complexity

# MINI-BATCH GRADIENT DESCENT

- Mini-batch
- Not too stochastic
- Fast
- Scalable
- Batch size
- epoch



# CATEGORICAL CROSS-ENTROPY

- Binary cross-entropy:
  - Sigmoid

$$J = \frac{1}{N}\sum_{i=1}^{N} -y^{(i)} * \log\left(h^{(i)}\right) - \left(1 - y^{(i)}\right) * \log(1 - h^{(i)})$$

- Categorical cross-entropy:
  - Softmax

$$J = \frac{1}{N}\sum_{i=1}^{N} -y^{(i)} * \log\left(h^{(i)}\right)$$