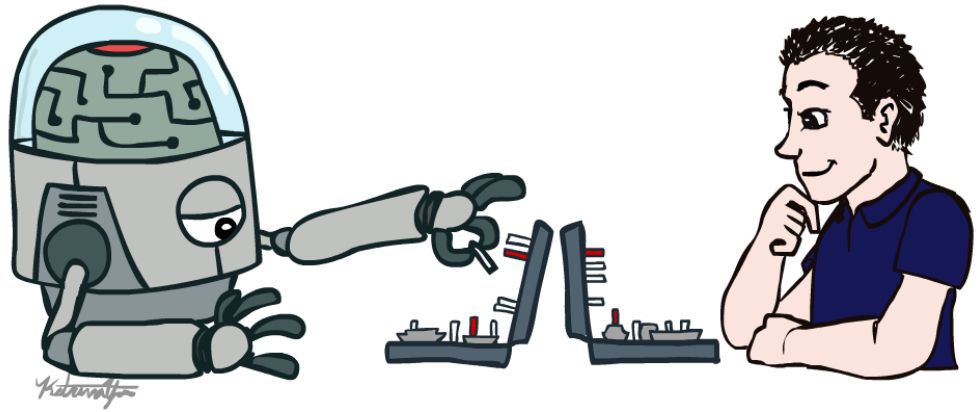
Lecture 12

Ashis Kumar Chanda
chanda@rowan.edu

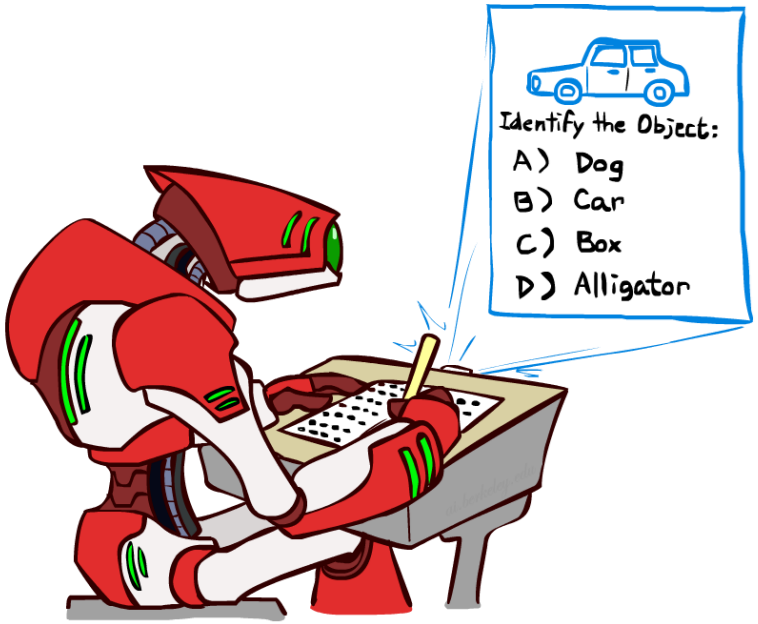


RowanUniversity

Background

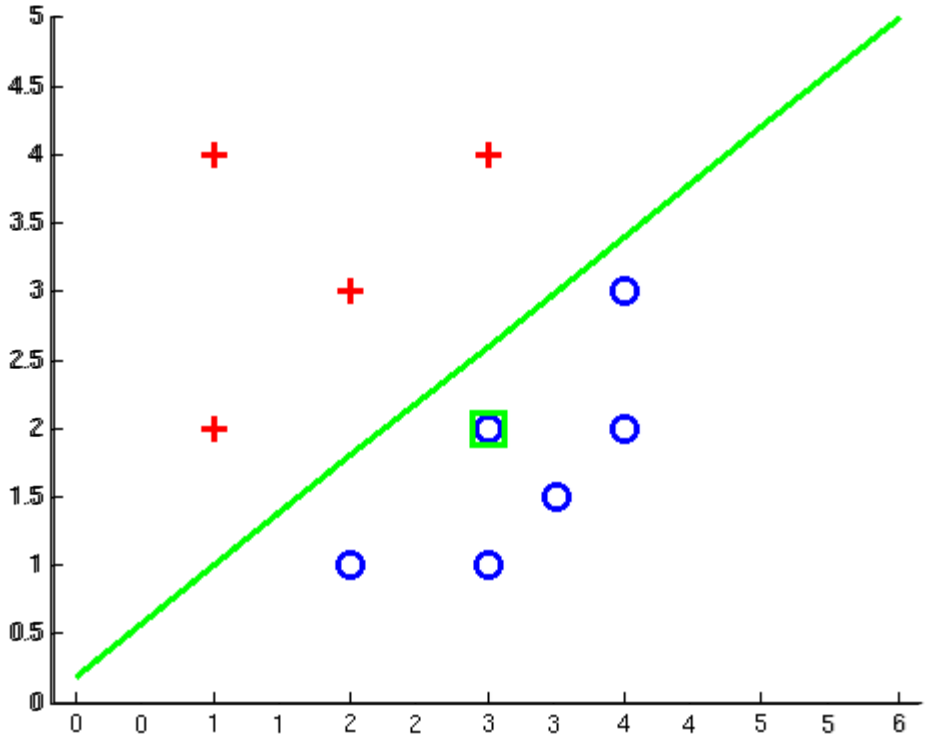
- **Last class: Decision tree for class prediction**
  - Classification: given inputs x, predict label, y (loan or not)
  - Ex: Spam email, identify object in an image.

- **What is regression problem?**
  - given inputs x, predict a value, y (loan amount)
  - Ex: Temperature, car price.



Example

- We can assume each example data as a point in a space.
- There could be a line that can separate different types of points.

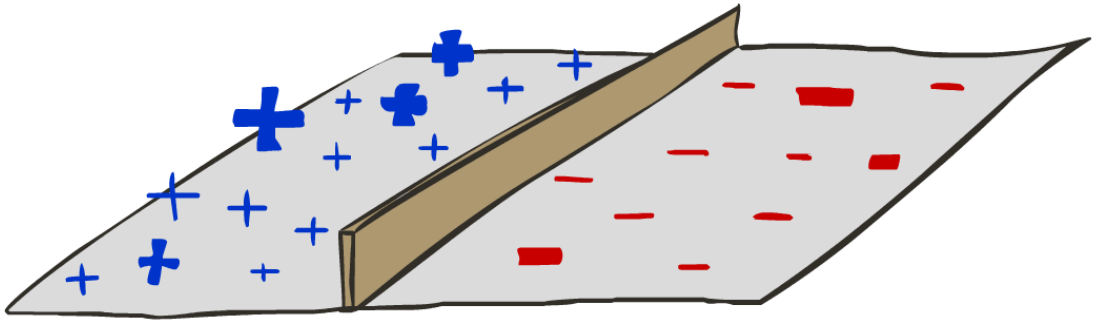


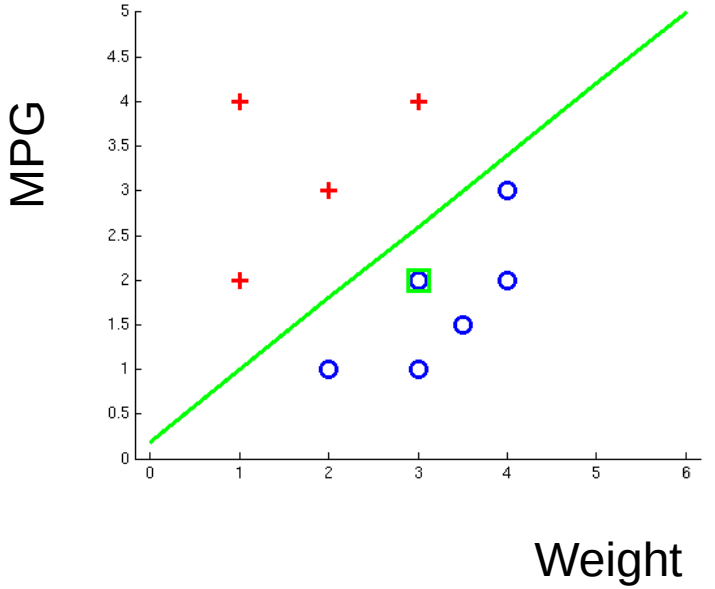
Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1

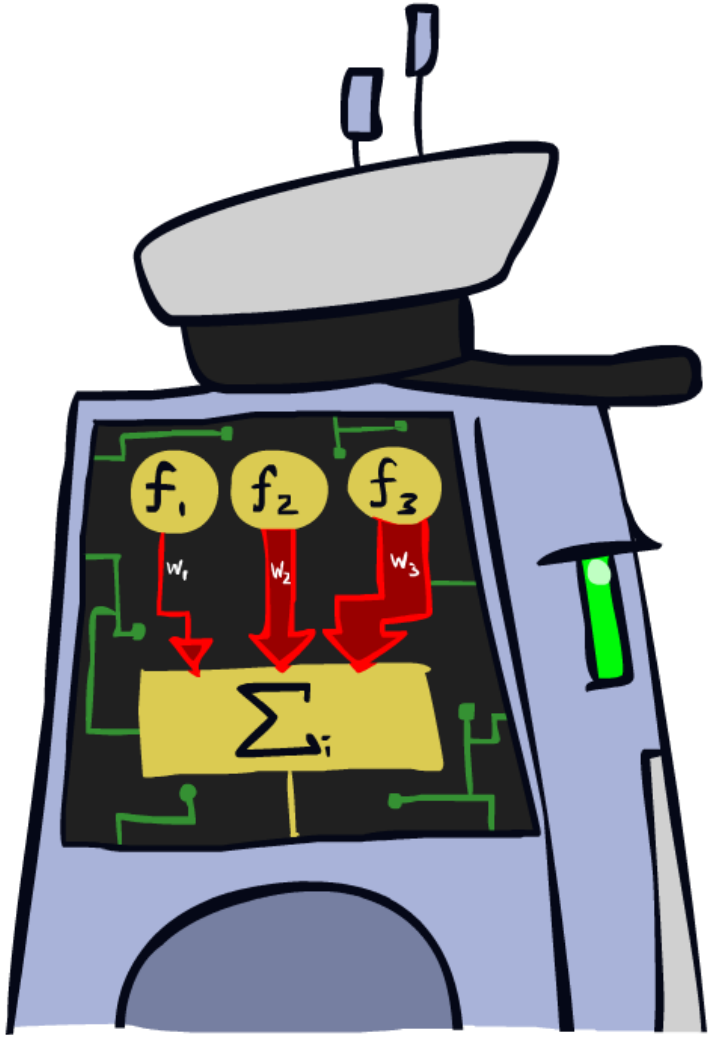
$$w$$

```
MPG   :     1*100
weight :    4K kg
...
```



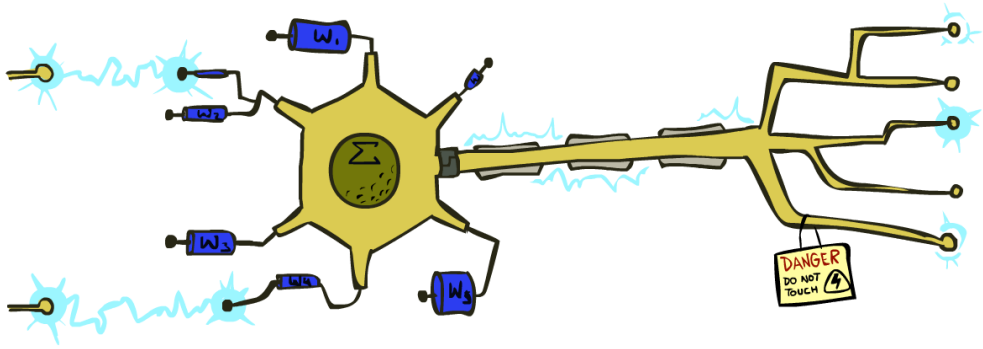


Linear Classifiers



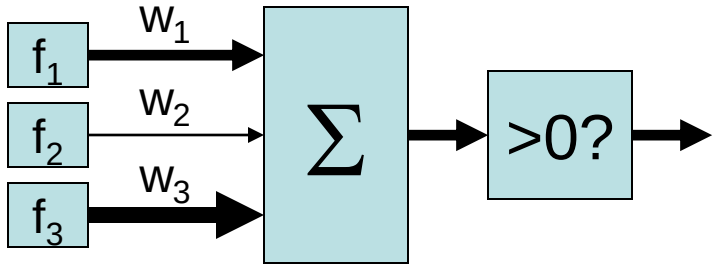
Linear Classifiers

- Inputs are feature values
- Each feature has a weight
- Sum is the activation



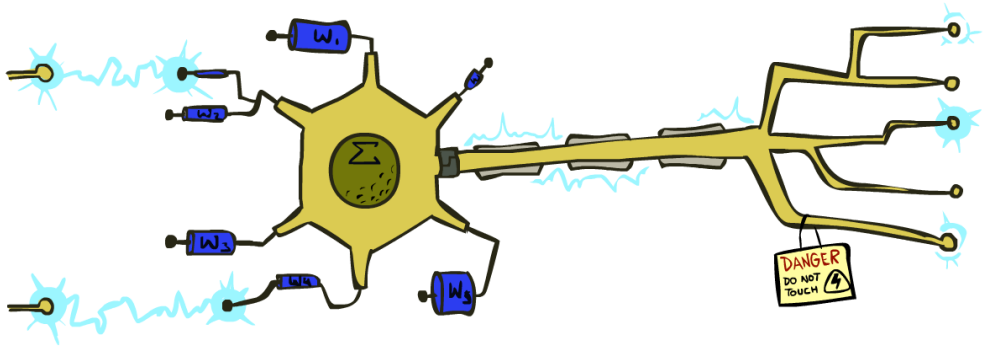
$$\text{Activation } f(x) = \sum_{i=0}^{k} Wi\ xi$$

- If the activation is:
  - Positive, output +1
  - Negative, output -1



Linear Classifiers

- Inputs are feature values
- Each feature has a weight
- Sum is the activation



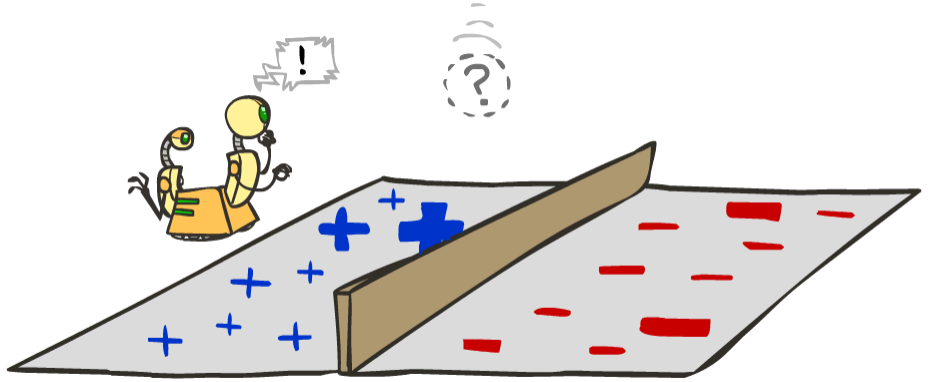
$$\text{Activation f}(x) = \sum_{i=0}^{k} Wi\ xi$$

- **What is the goal?**
- Learning: figure out the weight vector from examples

Weight Updates

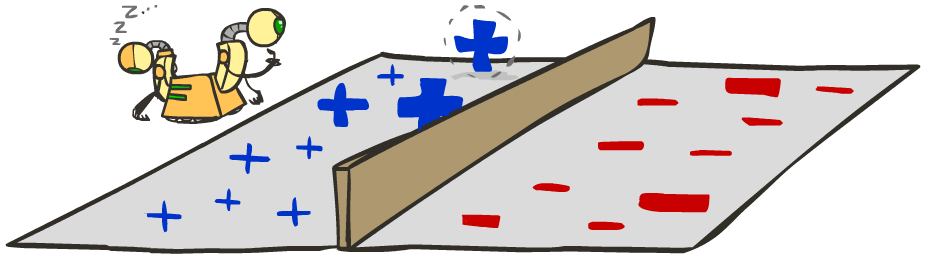
Learning: Binary Perceptron

- Start with random weights (small values)
- For each training instance:
  - Classify with current weights

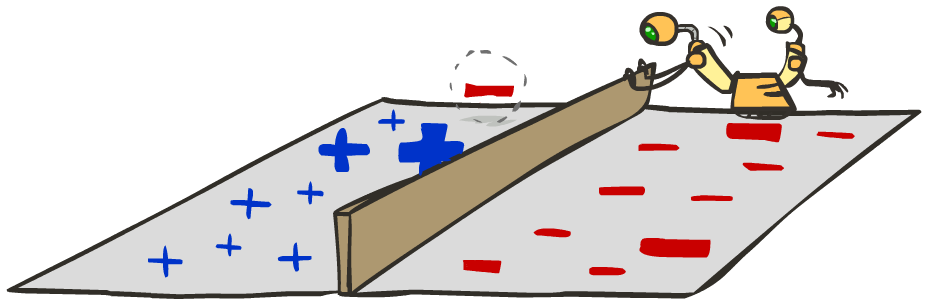


  - If correct (i.e., y=y*), no change!

y is actual value, y* is model prediction.



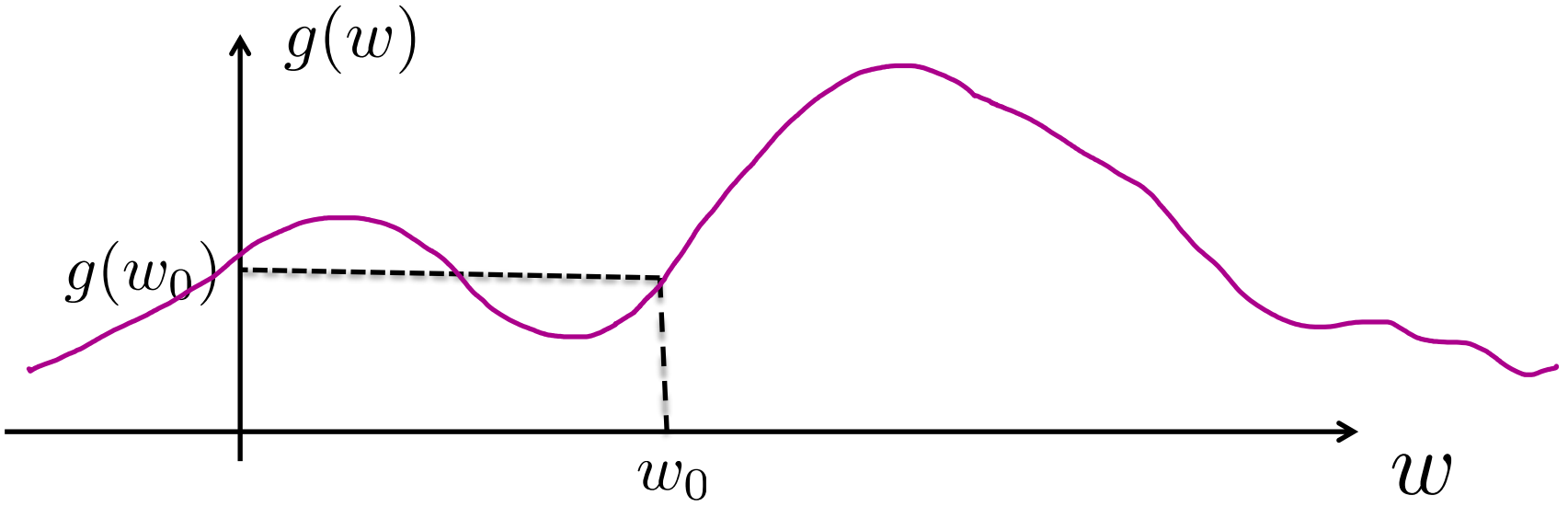
  - If wrong: adjust the weight vector



Hill Climbing

- Recall from previous lecture: simple, general idea
  - Start wherever
  - Repeat: move to the best neighboring state
  - If no neighbors better than current, quit

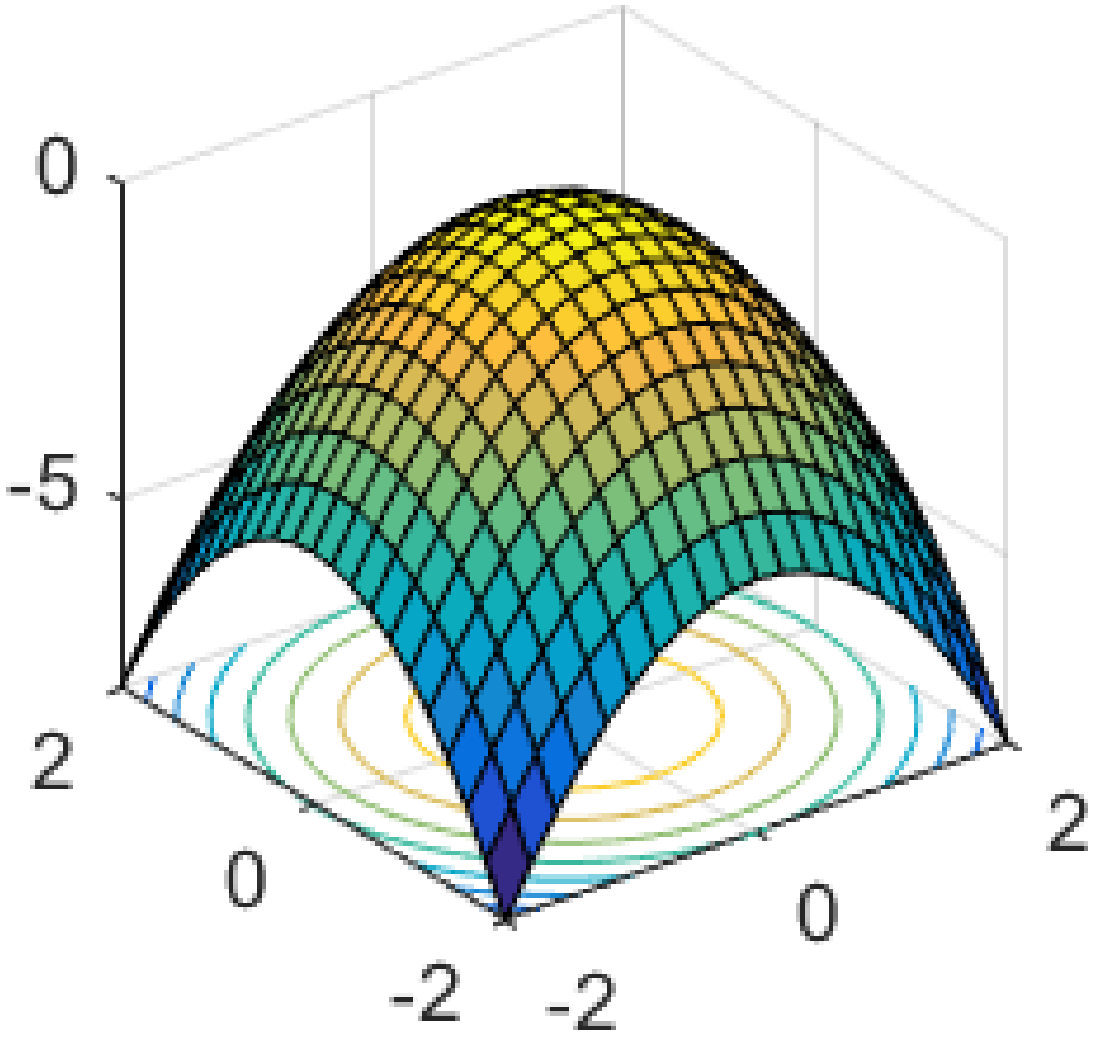
1-D Optimization



- Could evaluate $g(w_0 + h)$ and $g(w_0 - h)$
  - Then step in best direction

- Or, evaluate derivative: $\frac{\partial g(w_0)}{\partial w} = \lim_{h \to 0} \frac{g(w0 + h) - g(h)}{h}$
  - Tells which direction to step into

2-D Optimization



Source: offconvex.org

Gradient Ascent

- Perform update in uphill direction for each coordinate
- The steeper the slope (i.e. the higher the derivative) the bigger the step for that coordinate
- E.g., consider: $g(w_1, w_2)$
  - Updates:

$$w_1 \leftarrow w_1 + \alpha * \frac{\partial g}{\partial w_1}(w_1, w_2)$$

$$w_2 \leftarrow w_2 + \alpha * \frac{\partial g}{\partial w_2}(w_1, w_2)$$

α = learning rate

- Updates in vector notation:

$$w \leftarrow w + \alpha * \nabla_w g(w)$$

with: $\nabla_w g(w) = \begin{bmatrix} \frac{\partial g}{\partial w_1}(w) \\ \frac{\partial g}{\partial w_2}(w) \end{bmatrix}$ **= gradient**

Gradient in n dimensions

$$\nabla g = \begin{bmatrix} \frac{\partial g}{\partial w_1} \\ \frac{\partial g}{\partial w_2} \\ \cdots \\ \frac{\partial g}{\partial w_n} \end{bmatrix}$$

Optimization Procedure: Gradient Ascent

- init w
- for iter = 1, 2, …

$$w \leftarrow w + \alpha * \nabla g(w)$$

- α : learning rate --- tweaking parameter that needs to be chosen carefully
- How? Try multiple choices
  - Crude rule of thumb: update changes w about 0.1 – 1 %

Learning Steps

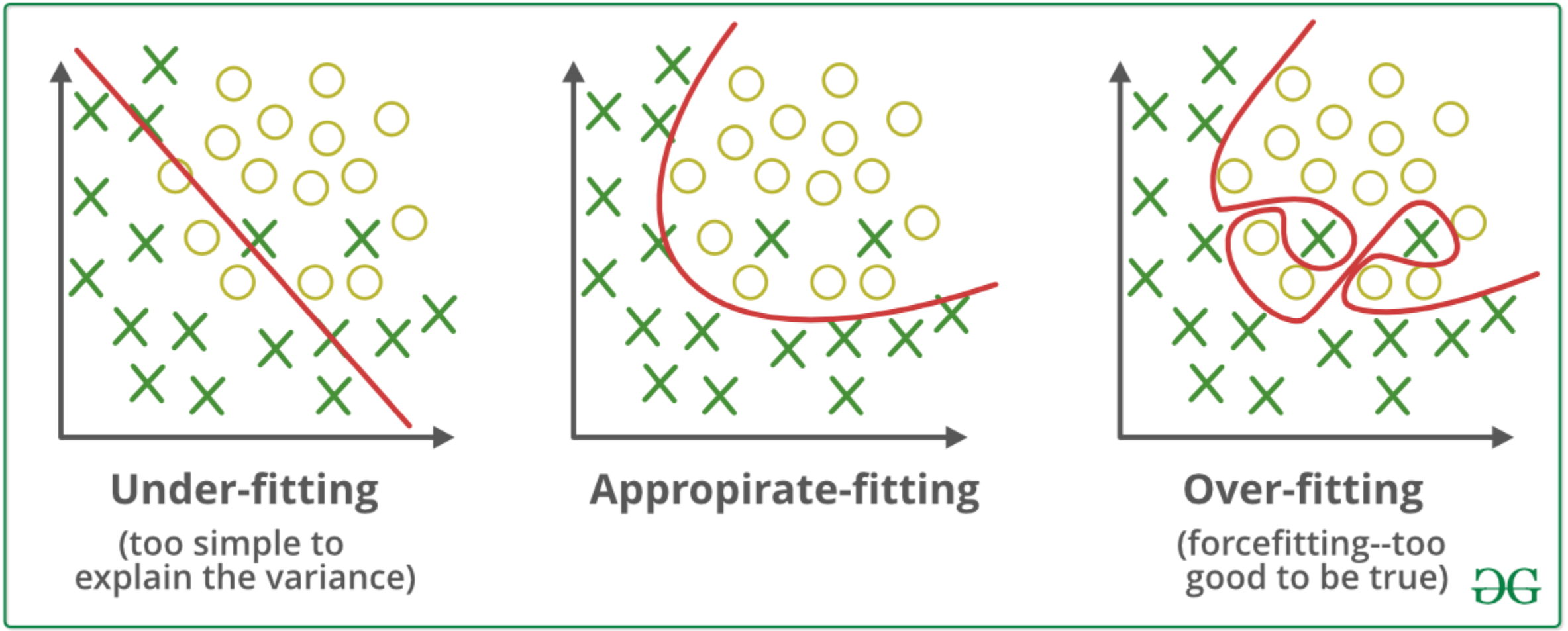
Learn values of weights from I/O pair

1. Start with random weight
2. Load training example input
3. Observe computed output
4. Modify weights to reduce difference
5. Iterate over all training example
6. Terminate when weights stops changing OR when error is very small (approx zero)

Problem

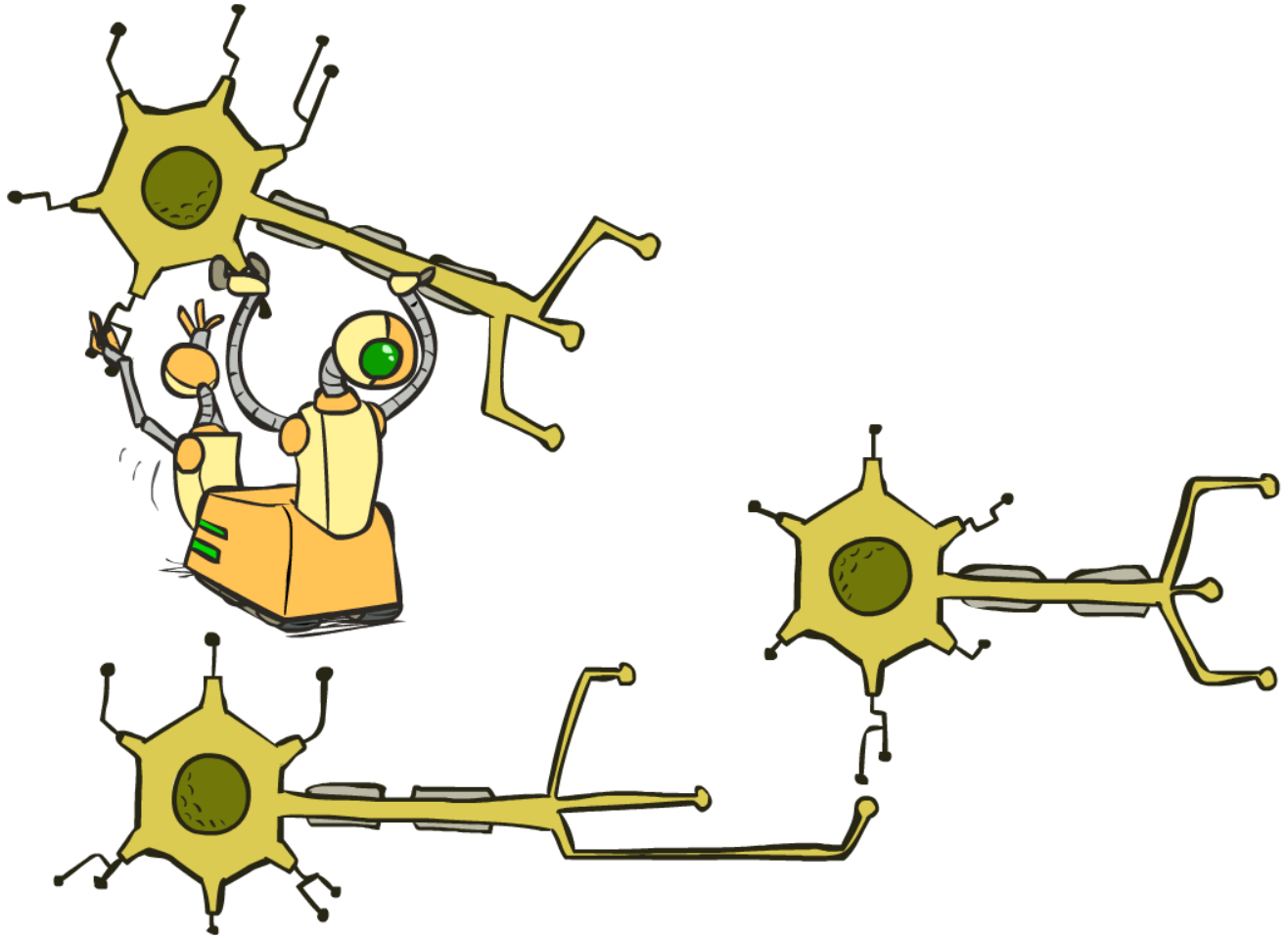
- In real life problems, data are not linearly separable.
- We need non-linear functions.
- Or a mixture of multiple linear or non-linear functions.

Problem



Source: geeksforgeeks

Neural Networks



Background

Where can neural network systems help?

- where we can't formulate an algorithmic solution.
- where we can get lots of examples of the behavior we require.
- where we need to pick out the structure from existing data.

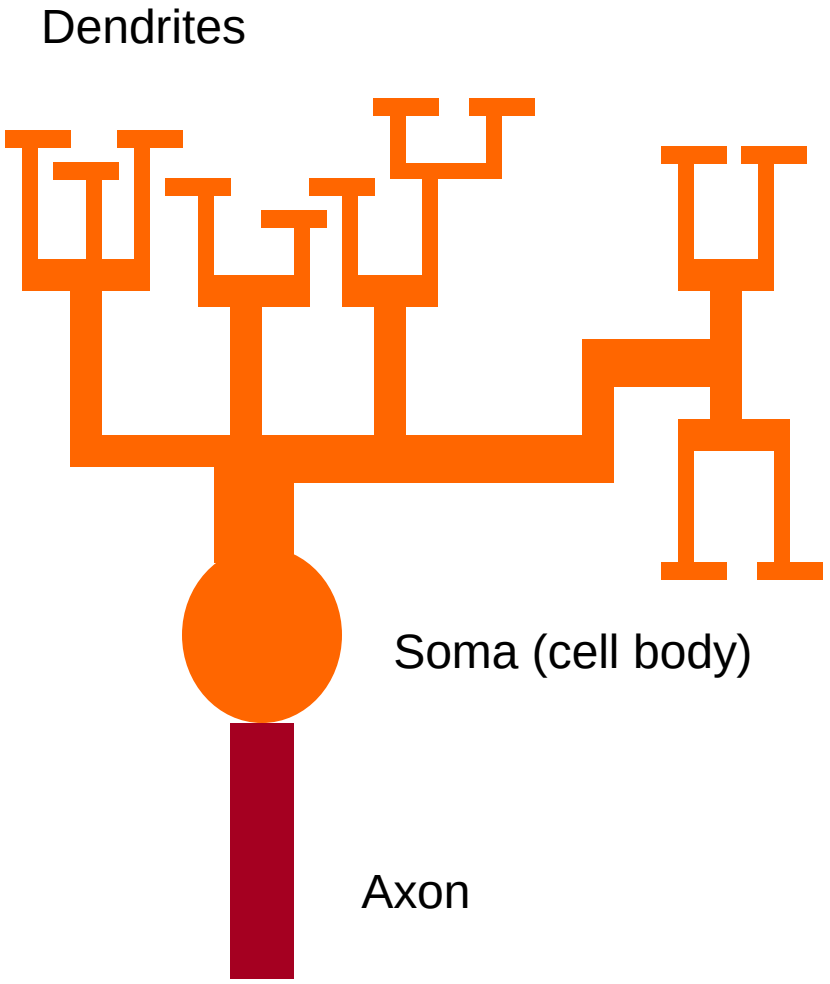
Biological inspiration

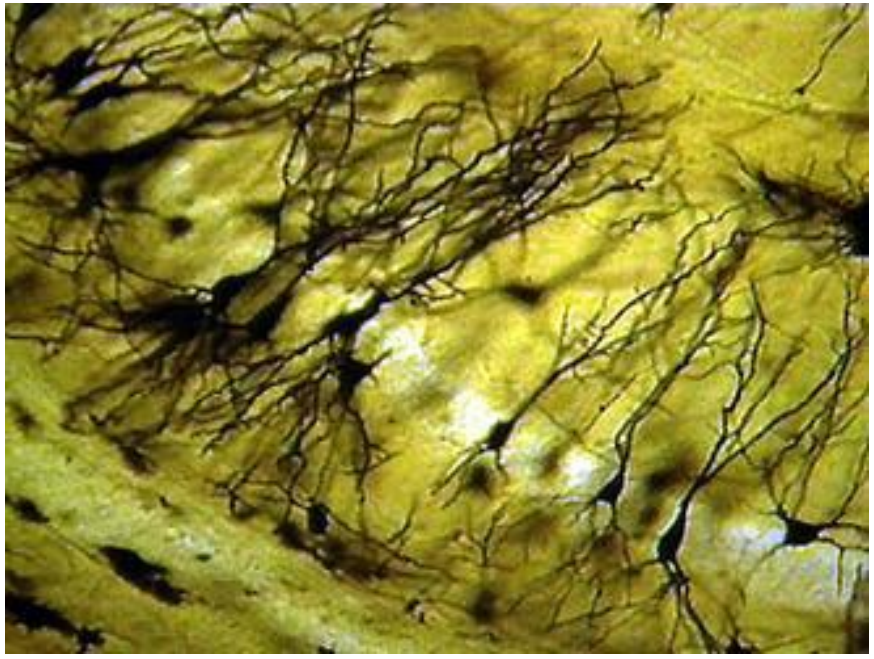
- Animals can react adaptively to changes in their external and internal environment, and they use their nervous system to perform these behaviours.

- An appropriate model/simulation of the nervous system should be able to produce similar responses and behaviours in artificial systems.

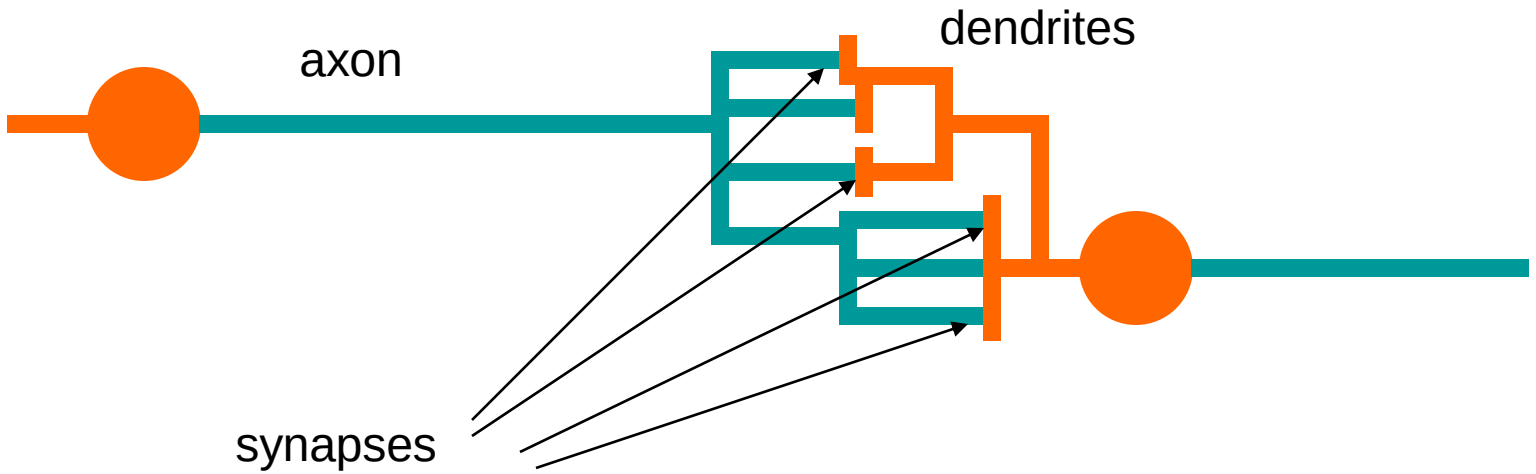
- The nervous system is built by relatively simple units, the neurons, so copying their behaviour and functionality should be the solution.

Biological inspiration





Biological inspiration



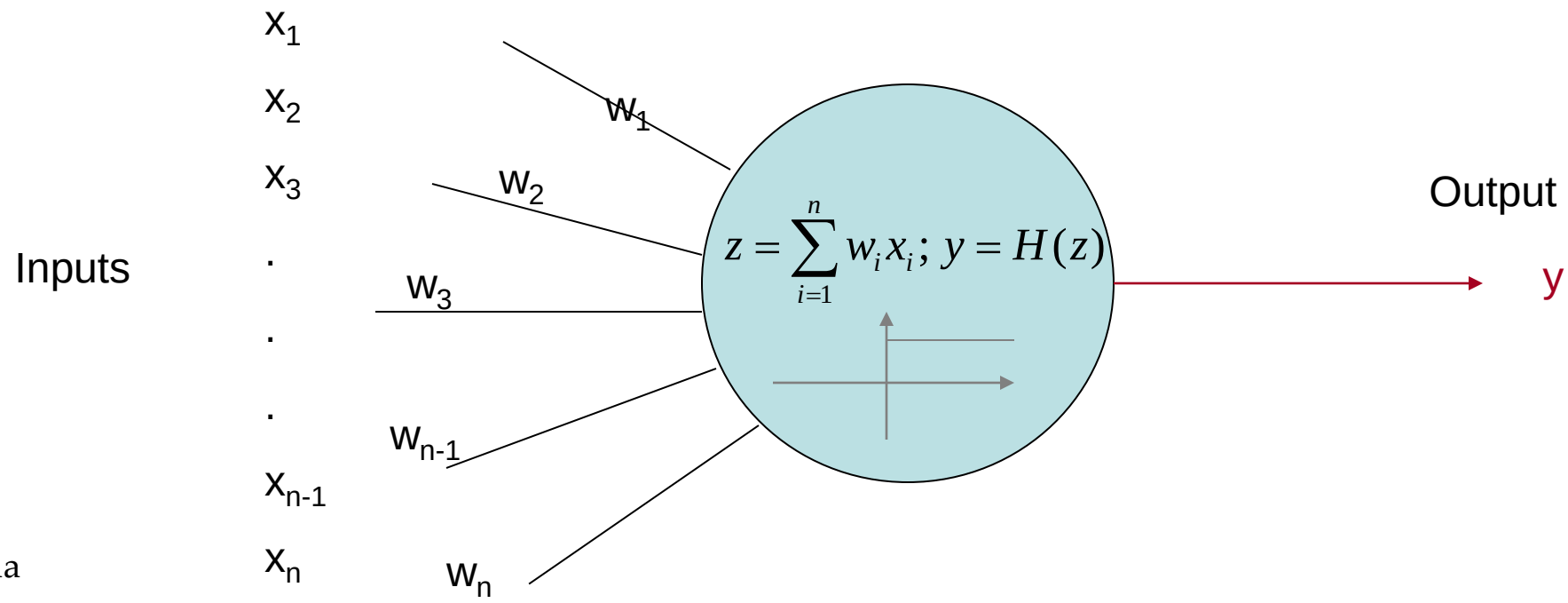
The information transmission happens at the synapses.

Biological inspiration

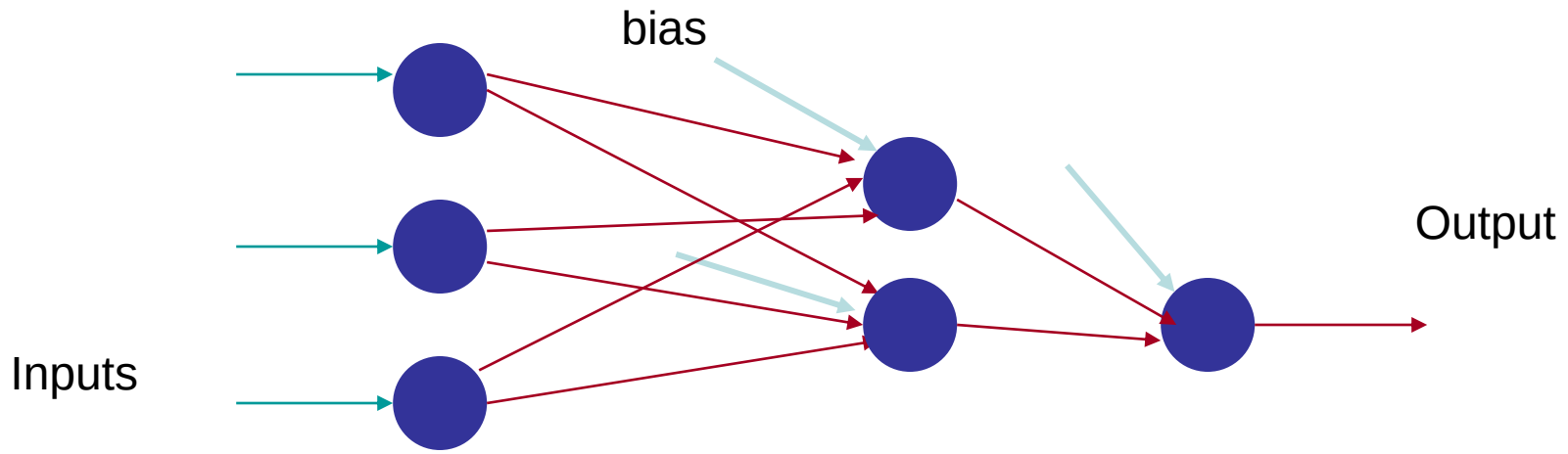
- The spikes travelling along the axon of the pre-synaptic neuron trigger the release of neurotransmitter substances at the synapse.
- The neurotransmitters cause excitation or inhibition in the dendrite of the post-synaptic neuron.
- The integration of the excitatory and inhibitory signals may produce spikes in the post-synaptic neuron.
- The contribution of the signals depends on the strength of the synaptic connection.

Artificial neurons

- Neurons work by processing information. They receive and provide information in form of spikes.

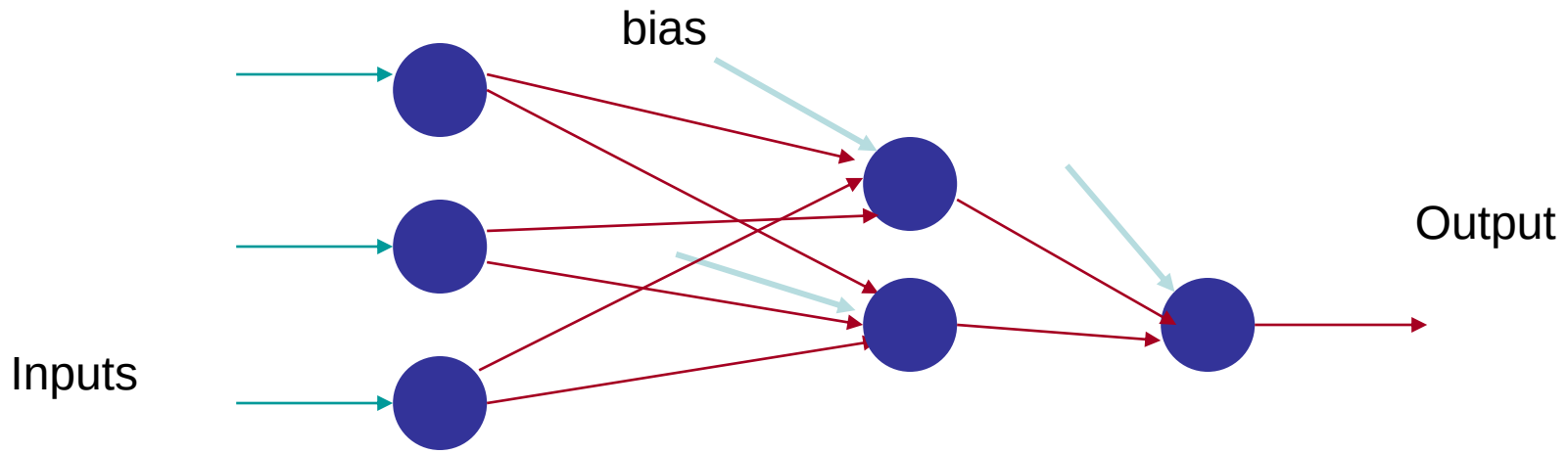


Artificial neural networks (ANN)



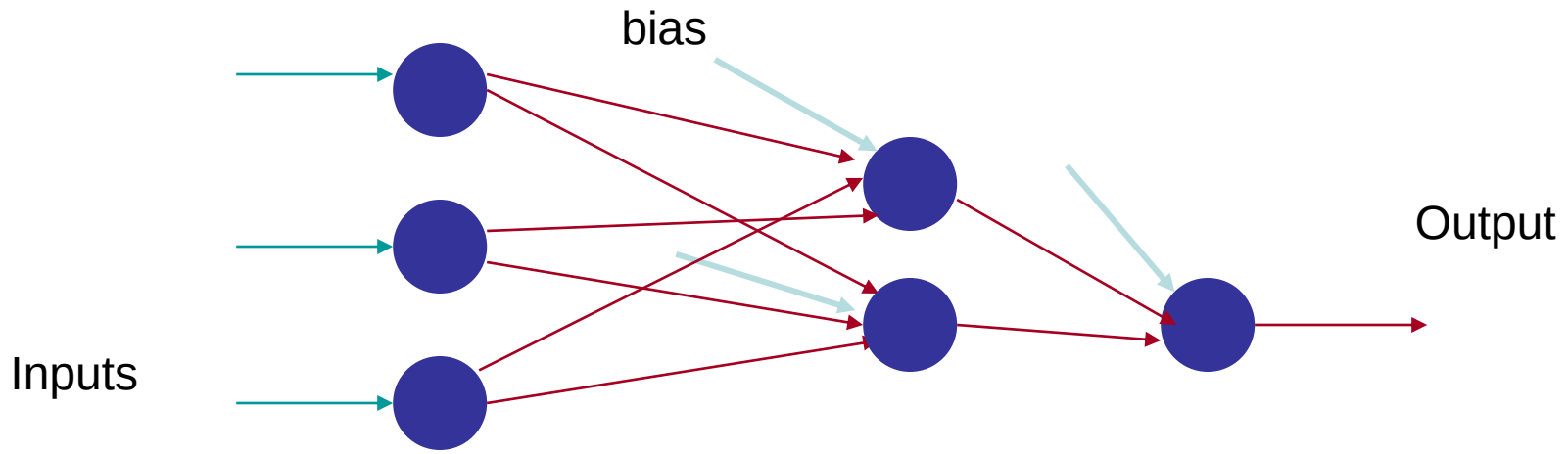
- An artificial neural network is composed of many artificial neurons that are linked together according to a specific network architecture.
- The objective of the neural network is to transform the inputs into meaningful outputs.

Artificial neural networks (ANN)



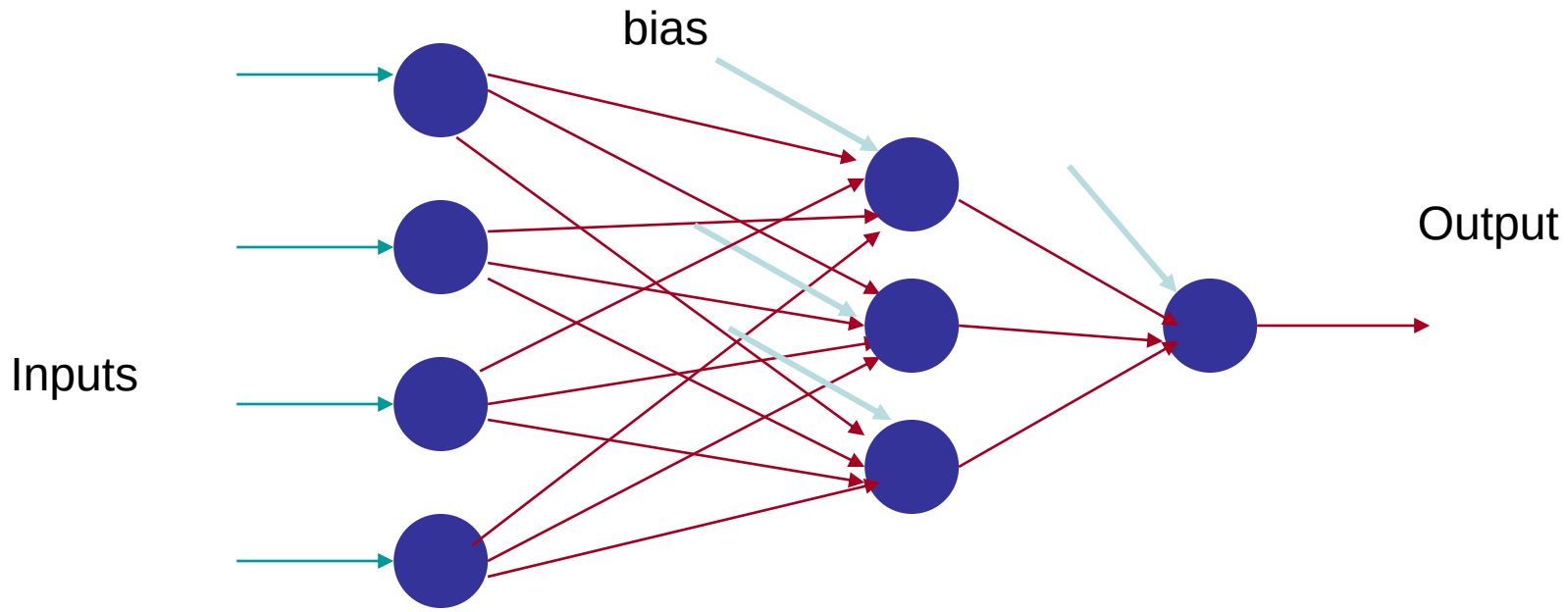
- It has one hidden layer, one output layer.
- This hidden layer has two hidden nodes.
- **Feedforward network:** information always moves one direction.
- **Fully-connected** means that each neuron of a layer has an incoming connection from all neurons of the previous layer.

Counting weights in (ANN)



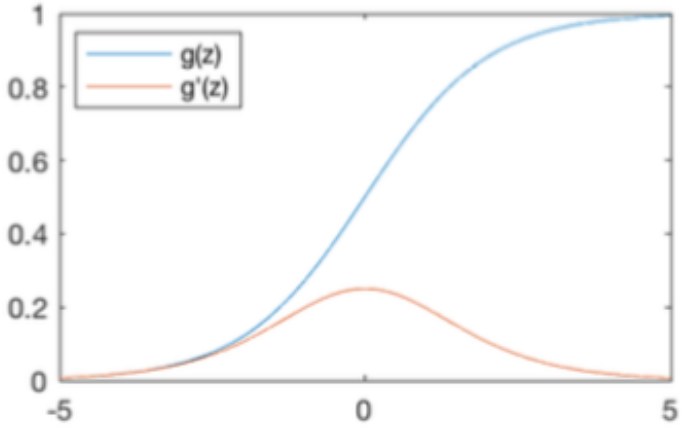
- **Fully-connected NN**:
- For hidden each layer: # hidden node * (# inputs +1)
- For output layer: (# inputs from previous layer +1)

Quiz: Counting weights in (ANN)



Common Activation Functions

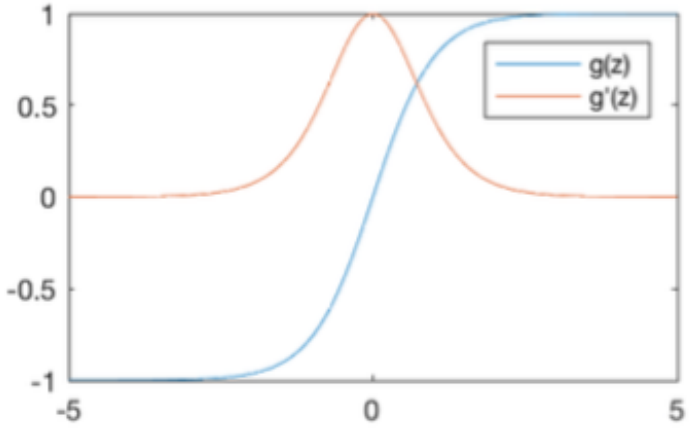
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

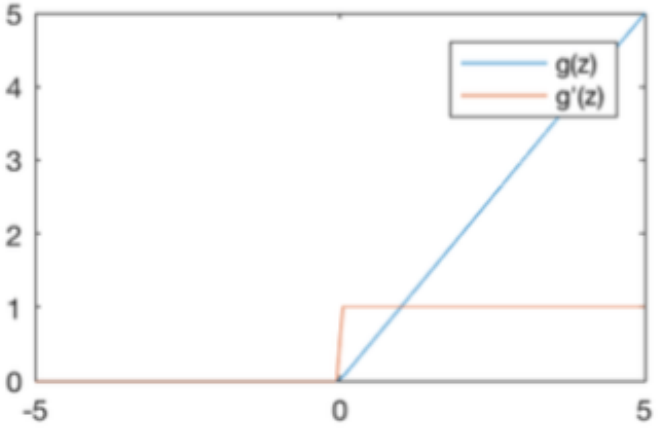
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)



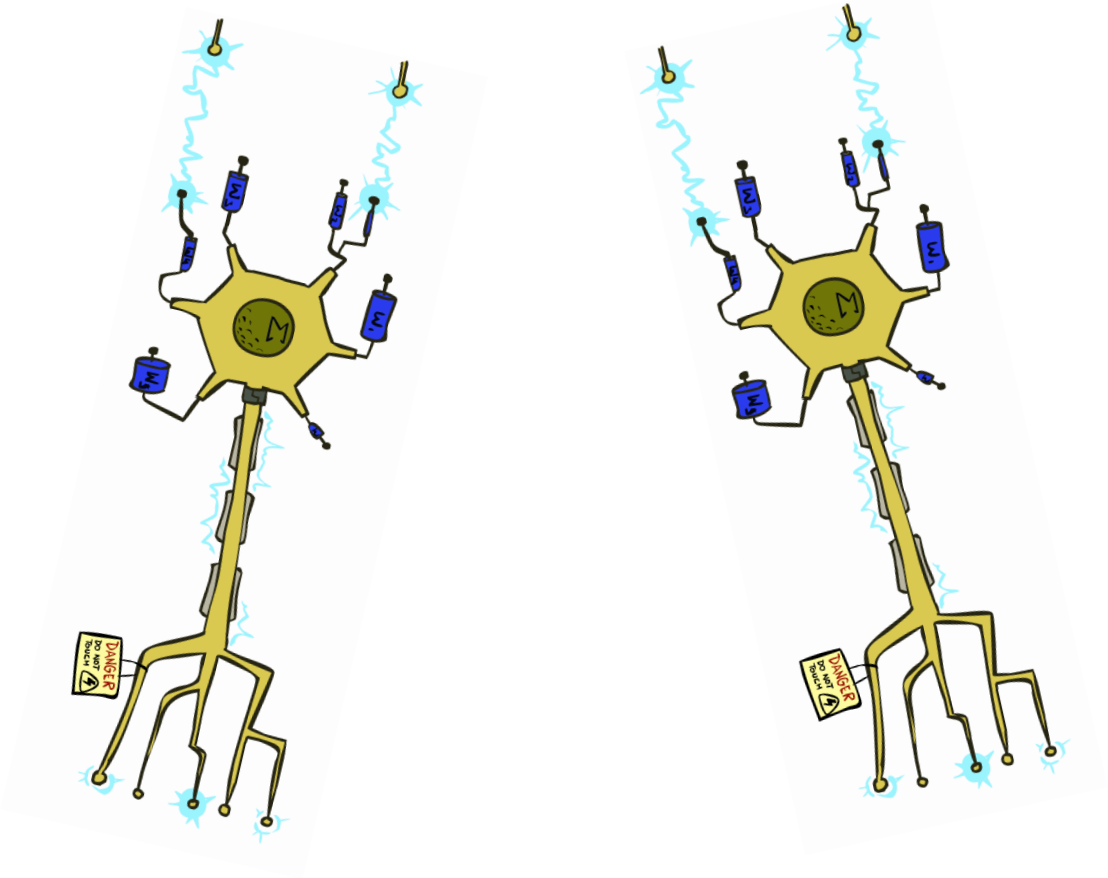
$$g(z) = \max(0, z)$$

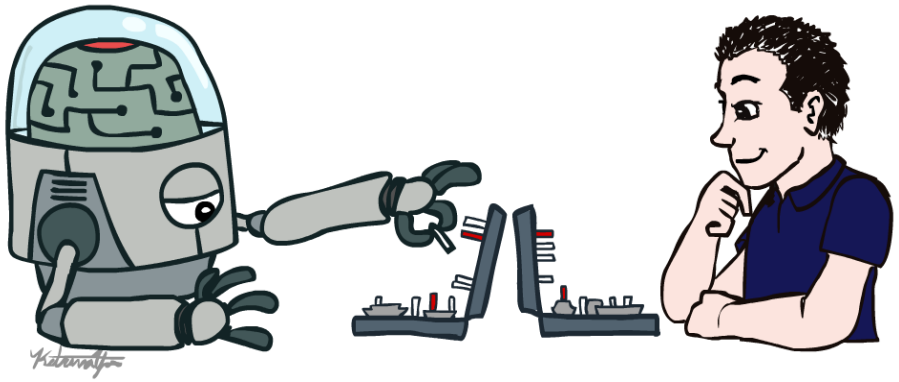
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

[source: MIT 6.S191 introtodeeplearning.com]

Today

- Linear regression
- Neural network





Thanks!