CS167: Machine Learning

Neural Network/Artificial Neural Network
Connections with Biology
Multilayer Perceptron (MLP)

Monday, April 6th, 2026

Drake
UNIVERSITY

Announcements

- **Project#1**
  - due on today (Monday) **04/06 by 11:59pm**

Today's Agenda

- Connections with biology: natural neurons vs. artificial neurons
- Multilayer Perceptrons (MLP)
- MLP Network Structure
- Learning MLP Weight Parameters
  - Recap from last week's offline lecture
  - Trainable parameters and their learnable weights

Today's Agenda

- Connections with biology: natural neurons vs. artificial neurons

Connections with biology

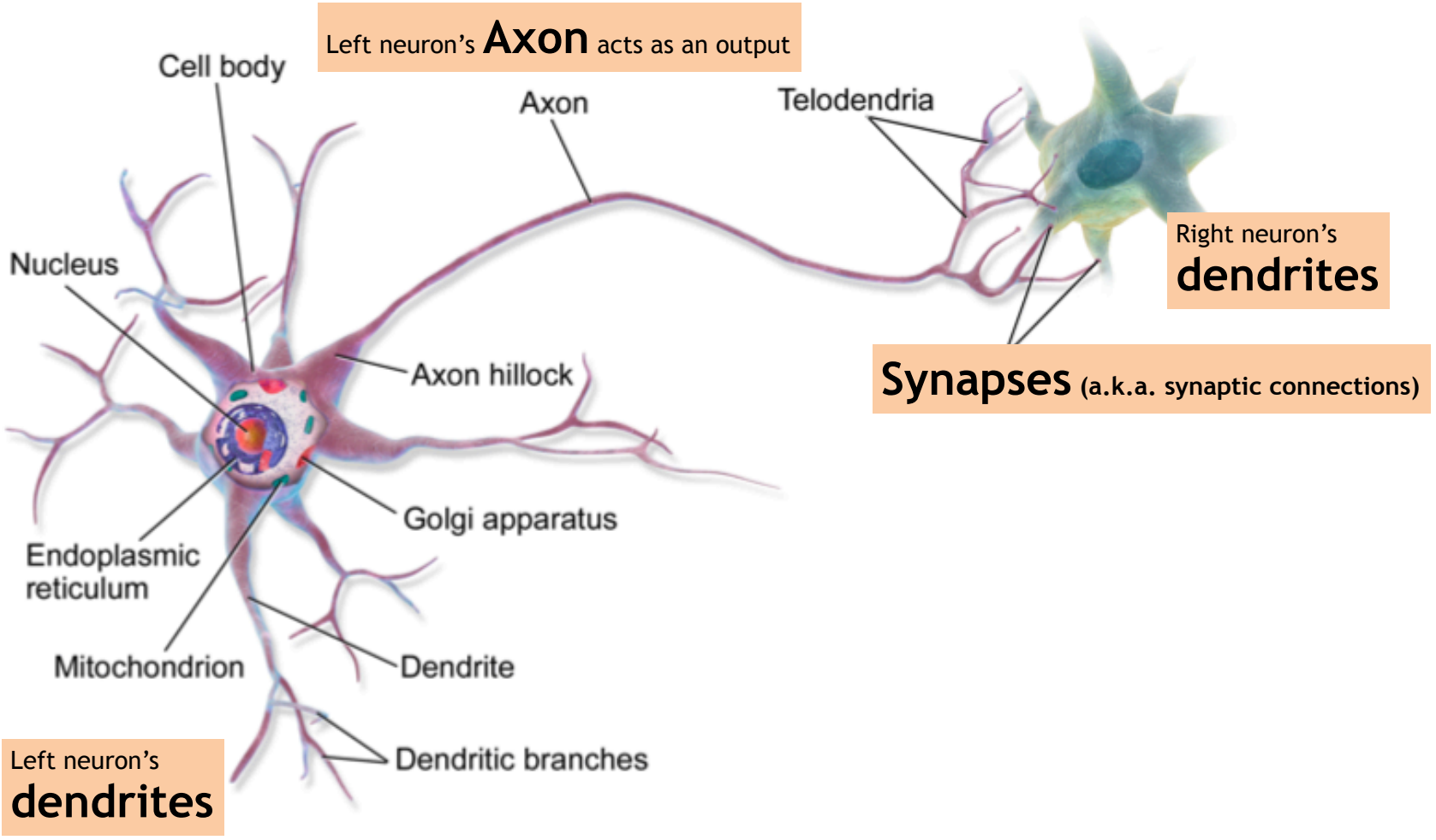


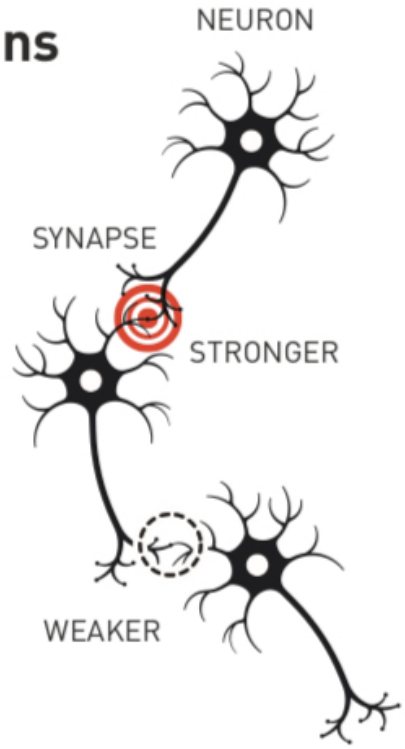
Figure: Illustration of two natural neurons connected together in a "circuit". The **output axon** of the left neuron makes a **synaptic connection** with the **dendrites** of the cell on the right. Electrical charges, in the form of ion flows, allow the cells to communicate.

Reference: Probabilistic Machine Learning - Kevin Murphy

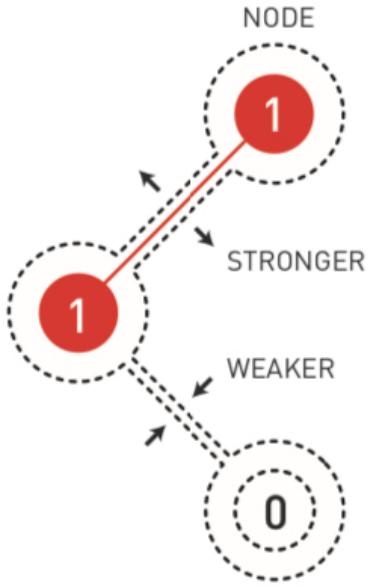
Intuition: Natural neurons vs. artificial neurons

Natural and artificial neurons

The brain's neural network is built from living cells, neurons, with advanced internal machinery. They can send signals to each other through the synapses. When we learn things, the connections between some neurons gets stronger, while others get weaker.



Artificial neural networks are built from nodes that are coded with a value. The nodes are connected to each other and, when the network is trained, the connections between nodes that are active at the same time get stronger, otherwise they get weaker.



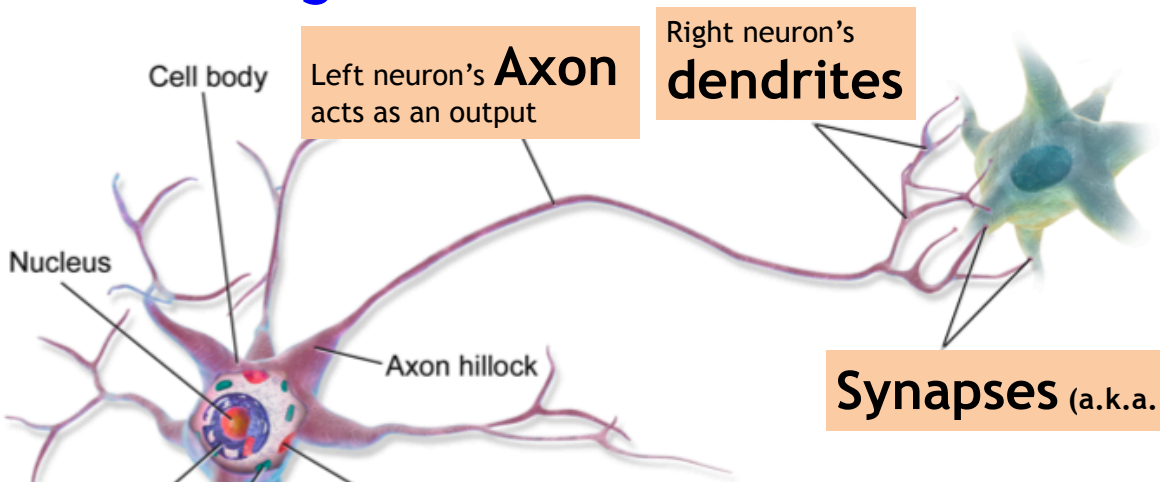
©Johan Jarnestad/The Royal Swedish Academy of Sciences

Natural neuron cells interacting

Artificial neurons interacting

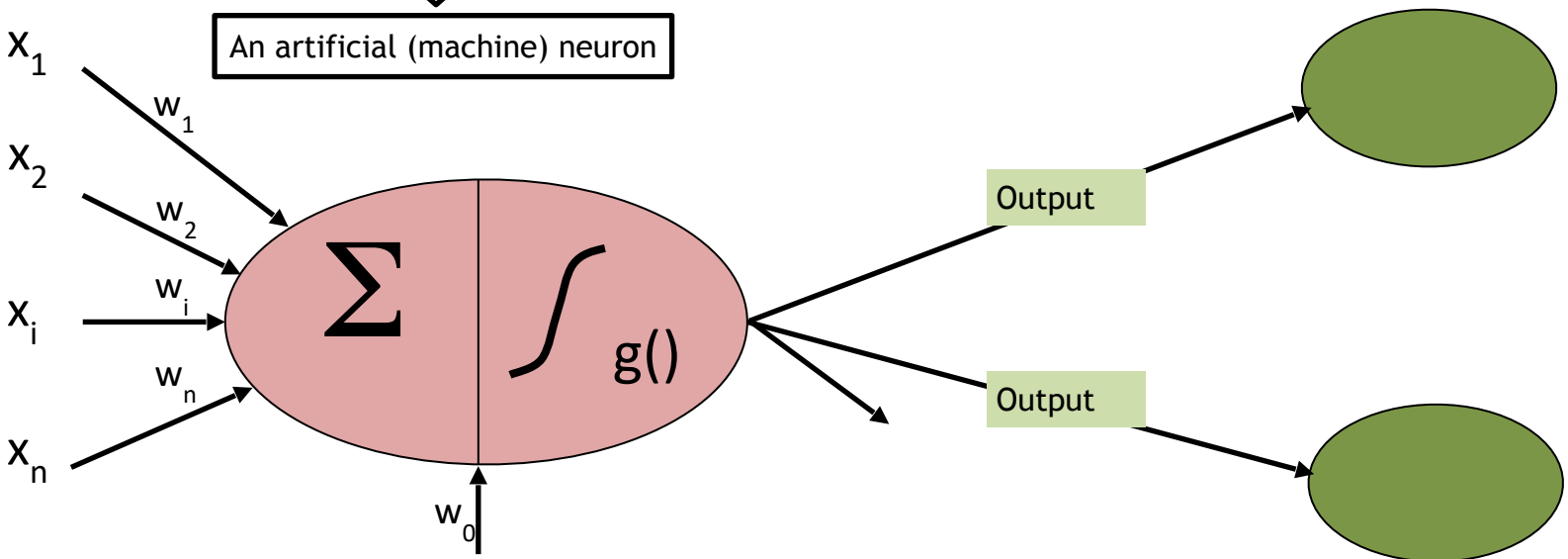
Photo Credit: Royal Swedish Academy awarding the 2024 Nobel Prize in Physics to John Hopfield and Geoffrey Hinton

Mathematical Encoding: Natural neurons vs. artificial neurons

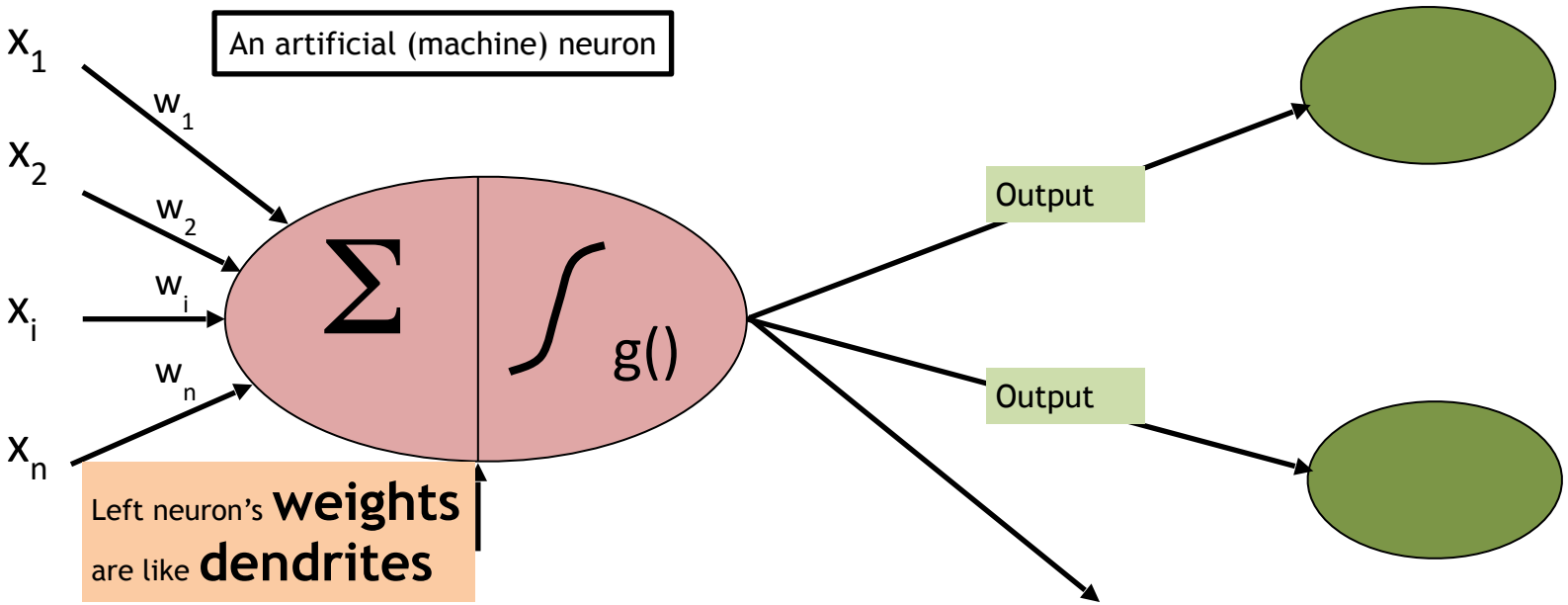


Natural neuron

An artificial (machine) neuron



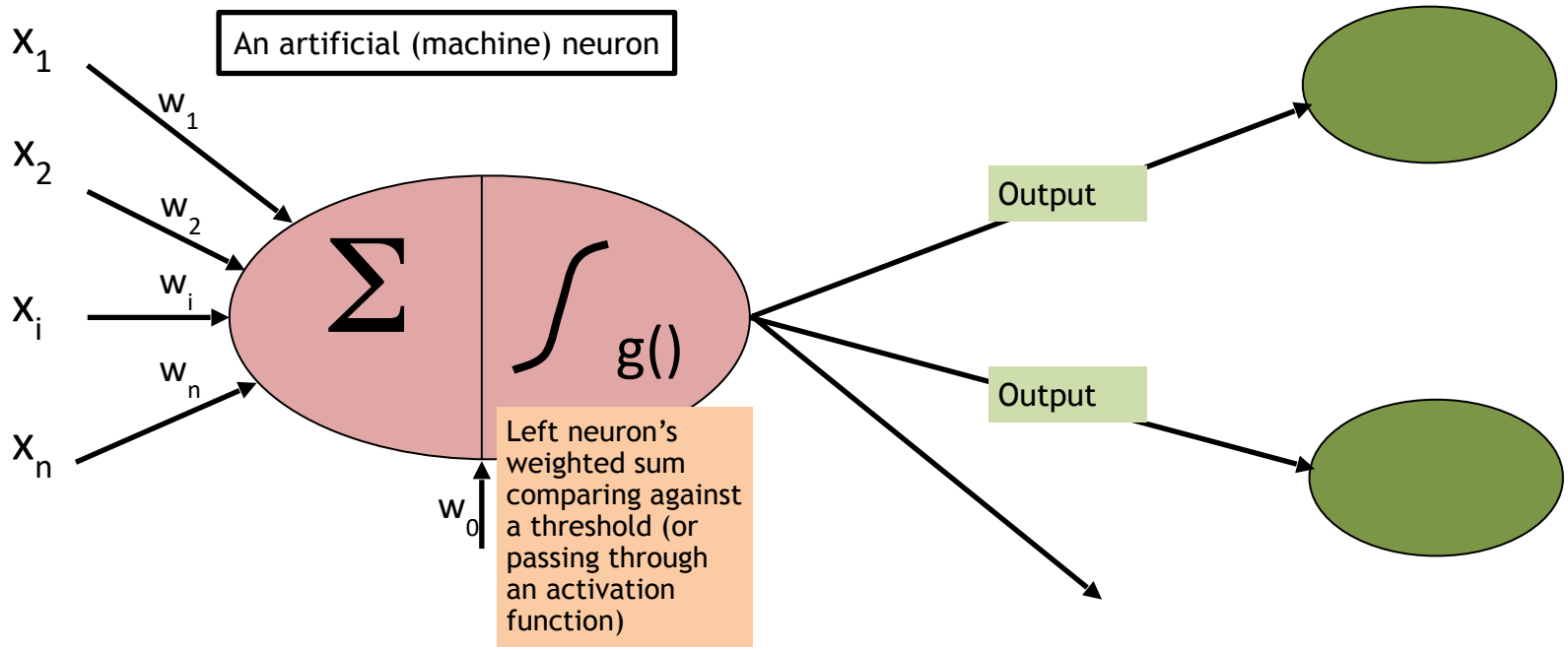
Mathematical Encoding: Natural neurons vs. artificial neurons



- we can say that whether neuron fires (its output) depends on the activity of its inputs, denoted by **x**, as well as the strength of the incoming connections, which we denote weights **w**

- We can compute a weighted sum of the inputs using $a_k = w^T_k x$. These **weights** can be viewed as "wires" connecting the inputs **x** to neuron (red one above); these are analogous to **dendrites** in a natural neuron

Mathematical Encoding: Natural neurons vs. artificial neurons

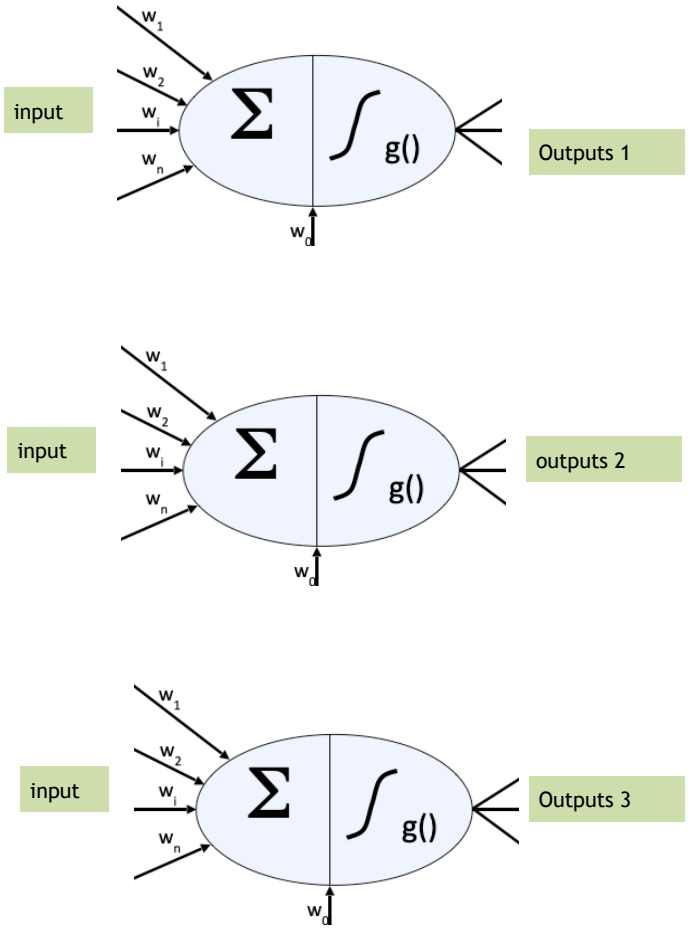


- weighted sum of the inputs ($a_k = w^T_k x$) is then compared to a threshold and if the activation exceeds the threshold, the neuron fires; this is analogous to the neuron emitting **an electrical output**. The threshold-based model is called the **McCulloch-Pitts model of the neuron**[1]

- We can combine multiple such neurons together to make an **Artificial Neural Network (ANN).** This result has been sometimes viewed as a model of the brain. However, ANNs differs from biological brains in many ways. Now, let's see such a combination of neurons in an ANN.

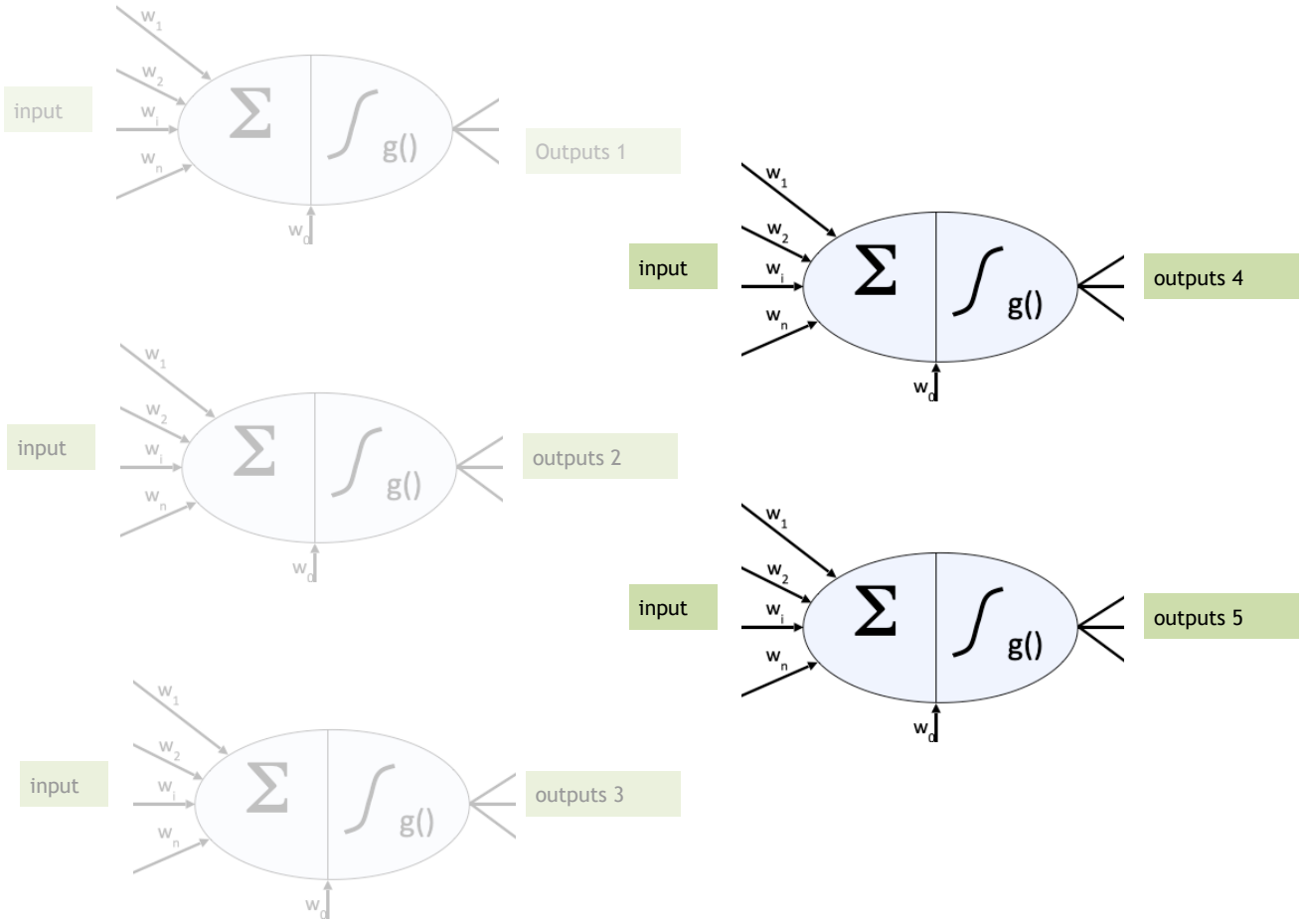
1. W. McCulloch and W. Pitts. ""A logical calculus of the ideas immanent in nervous activity". In: Bulletin of Mathematical Biophysics (1943)

Add three neurons in the first Layer

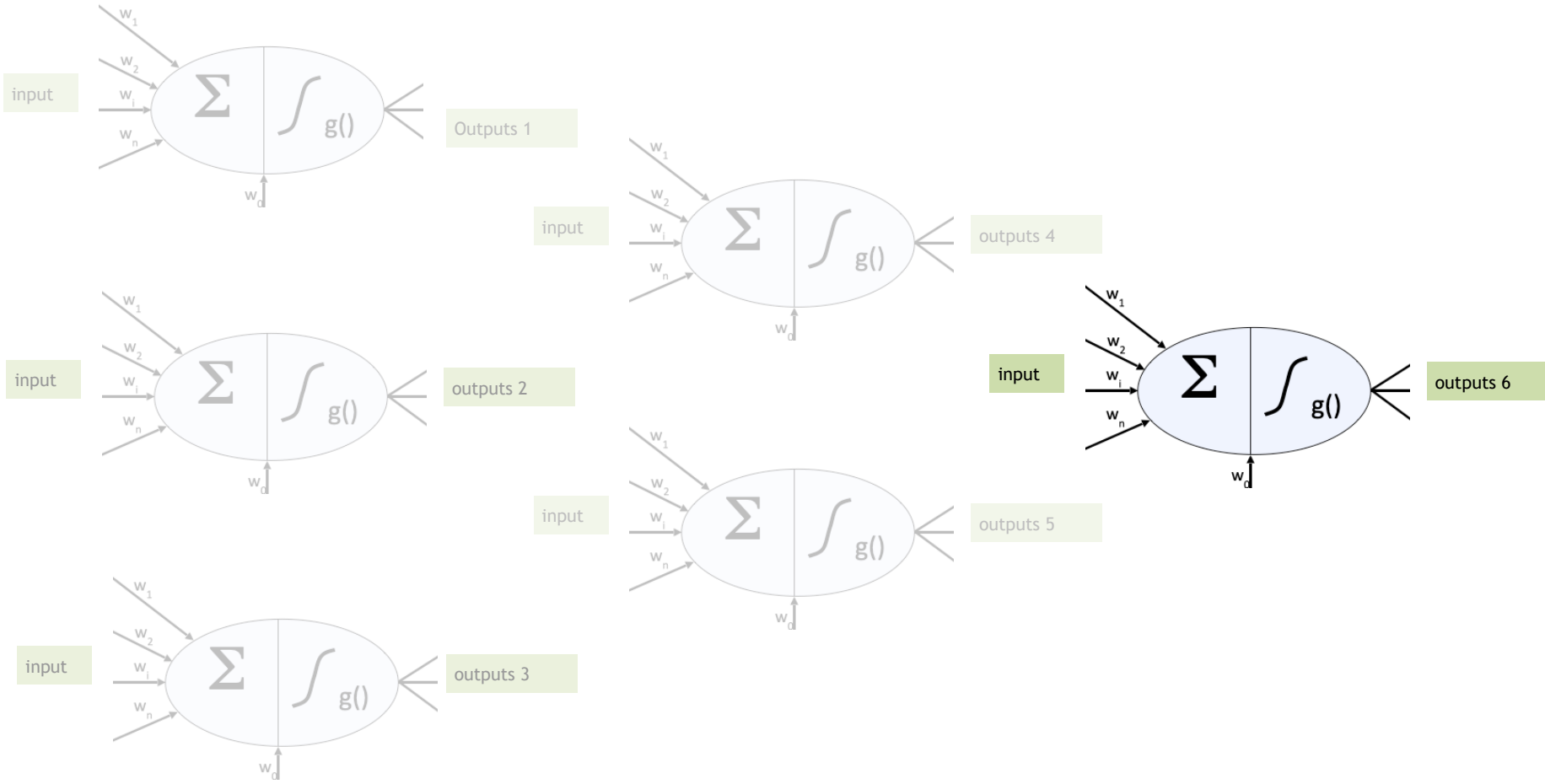


Layer 1

Add a two more neuron in the second layer

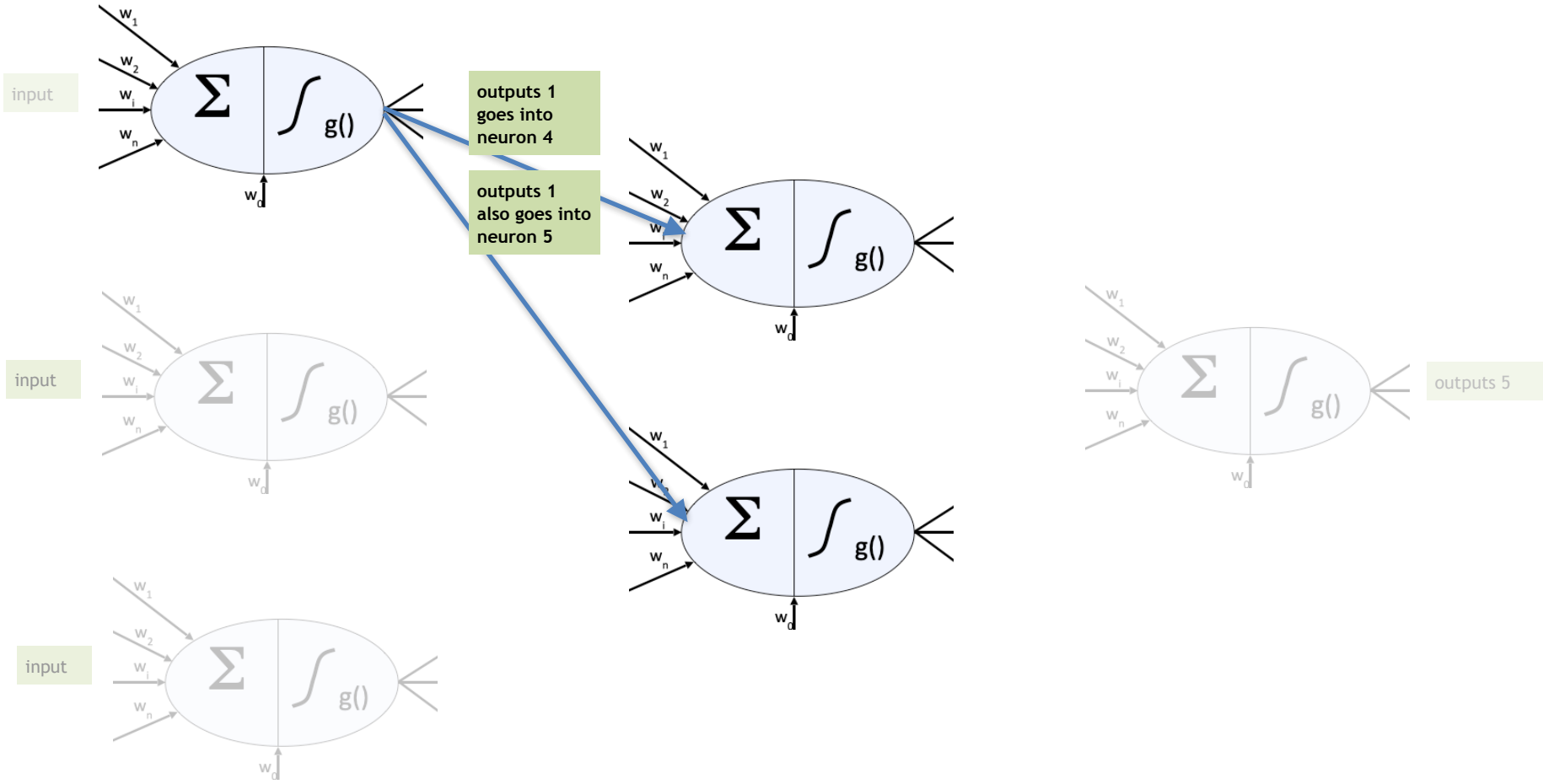


Add a two more neuron in the third layer



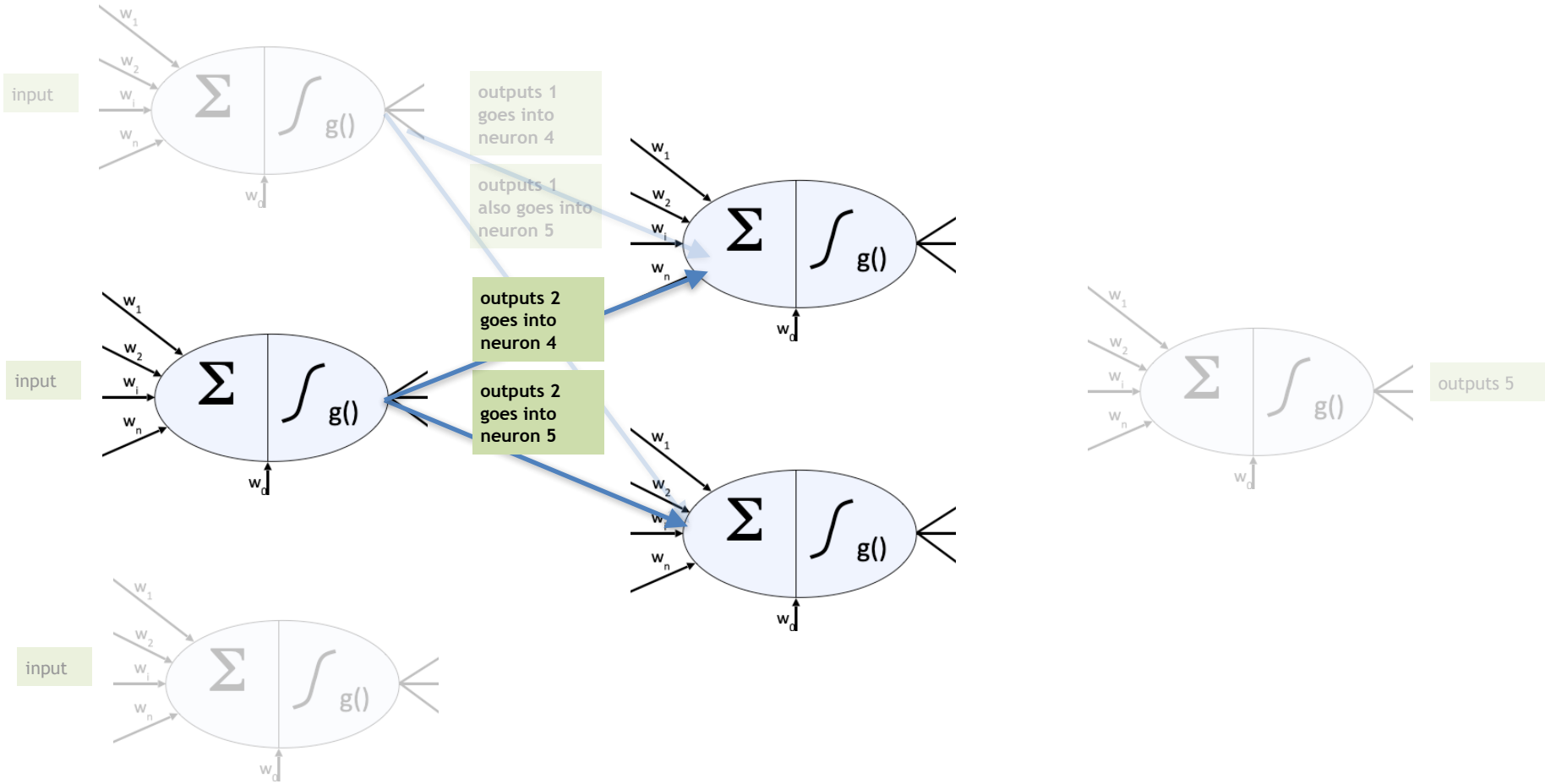
How can we Connect the Layers?

- Dense connection: connect neurons in between **Layer 1** and **Layer 2**



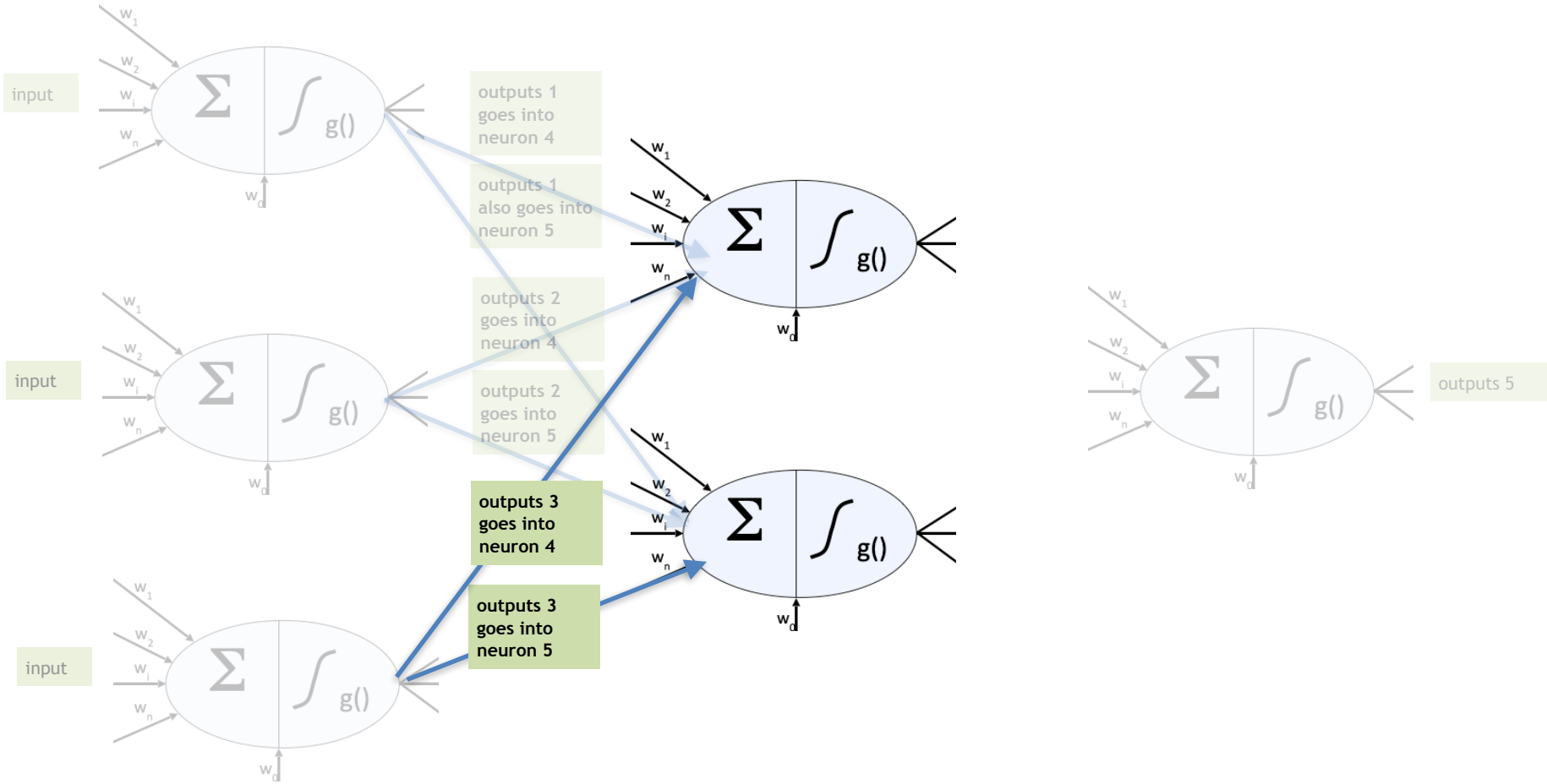
How can we Connect the Layers?

- Dense connection: connect neurons in between **Layer 1** and **Layer 2**



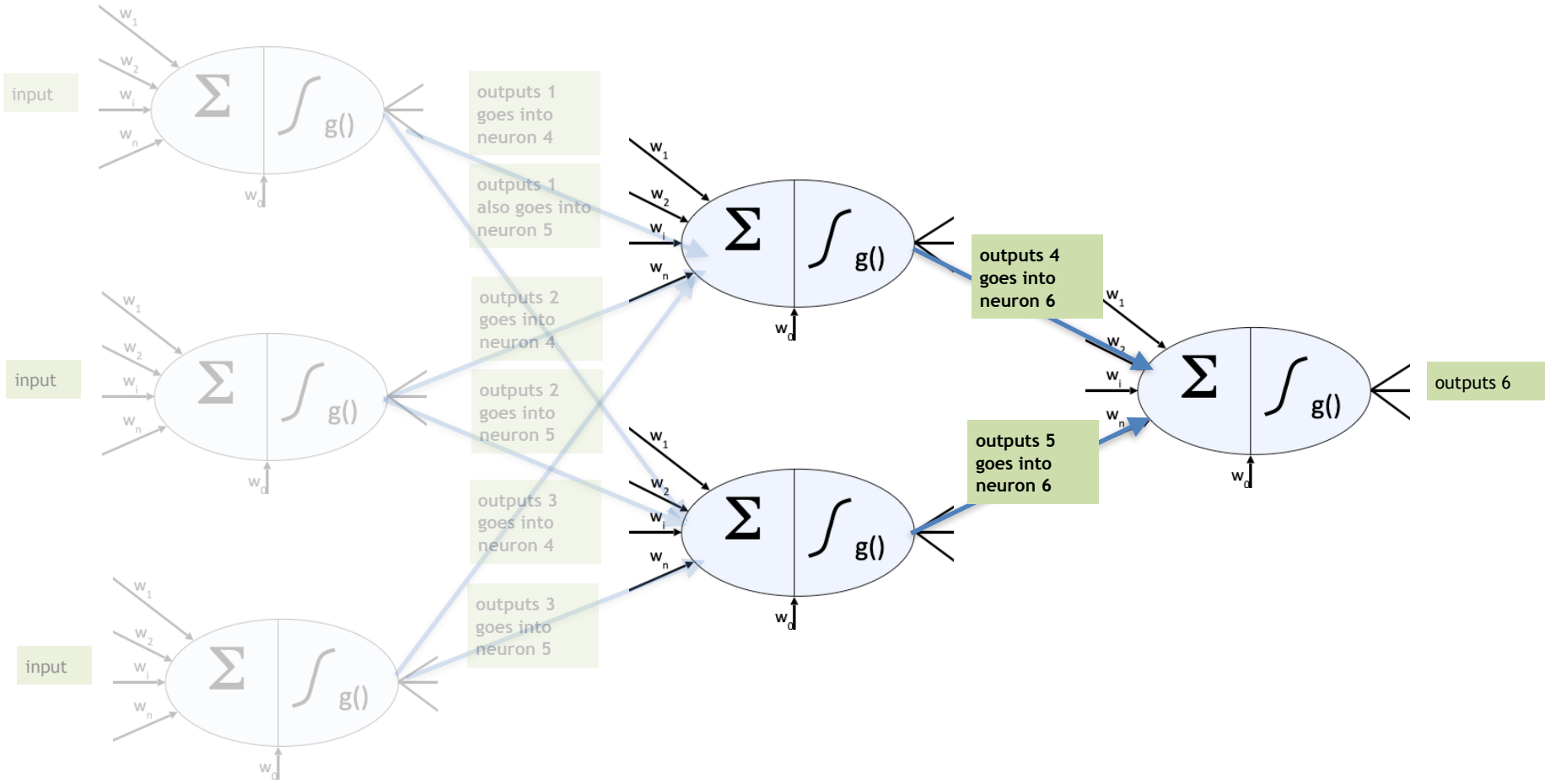
How can we Connect the Layers?

- Dense connection: connect neurons in between **Layer 1** and **Layer 2**



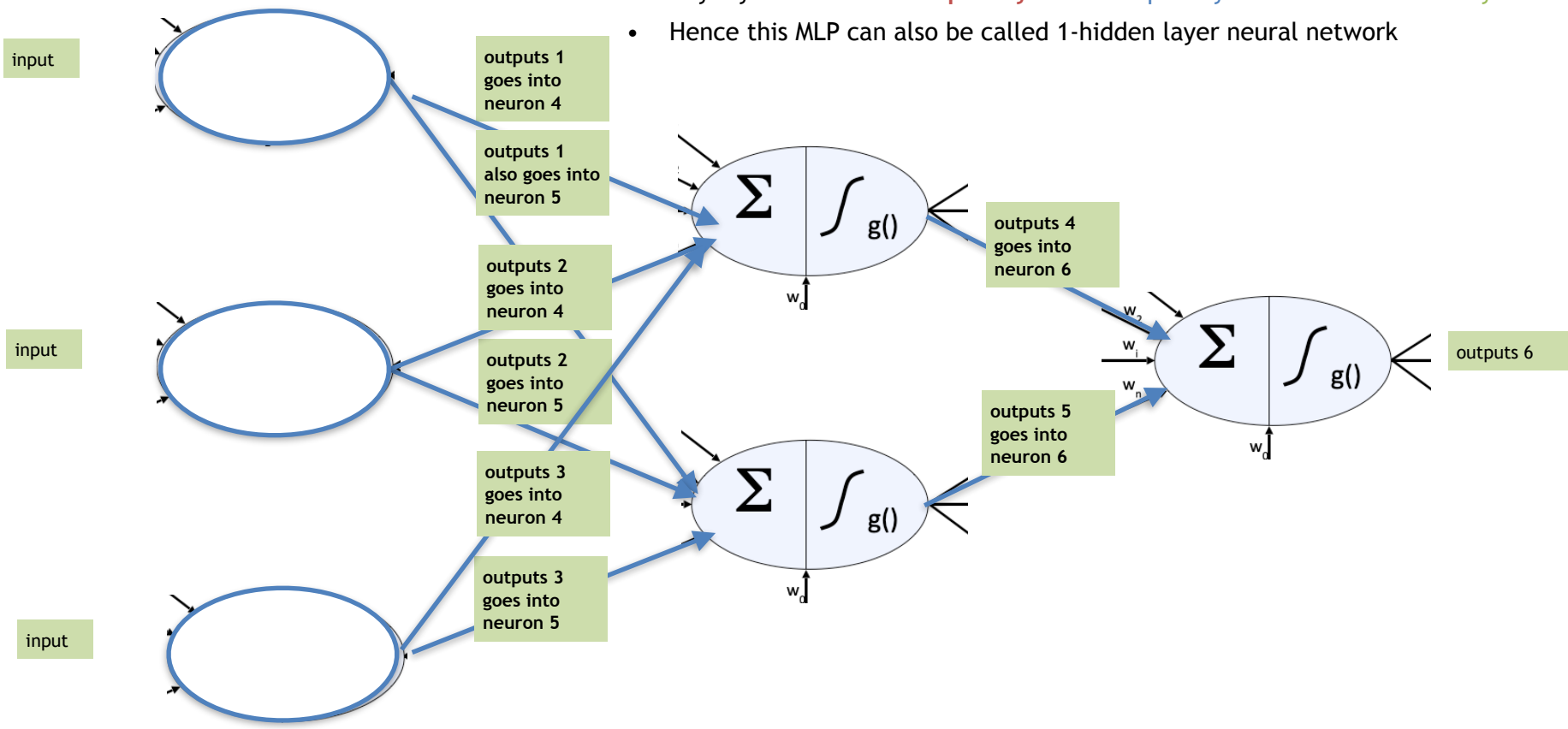
How can we Connect the Layers?

- Dense connection: connect neurons in between **Layer 2** and **Layer 3**



1-Hidden Layer Neural Network

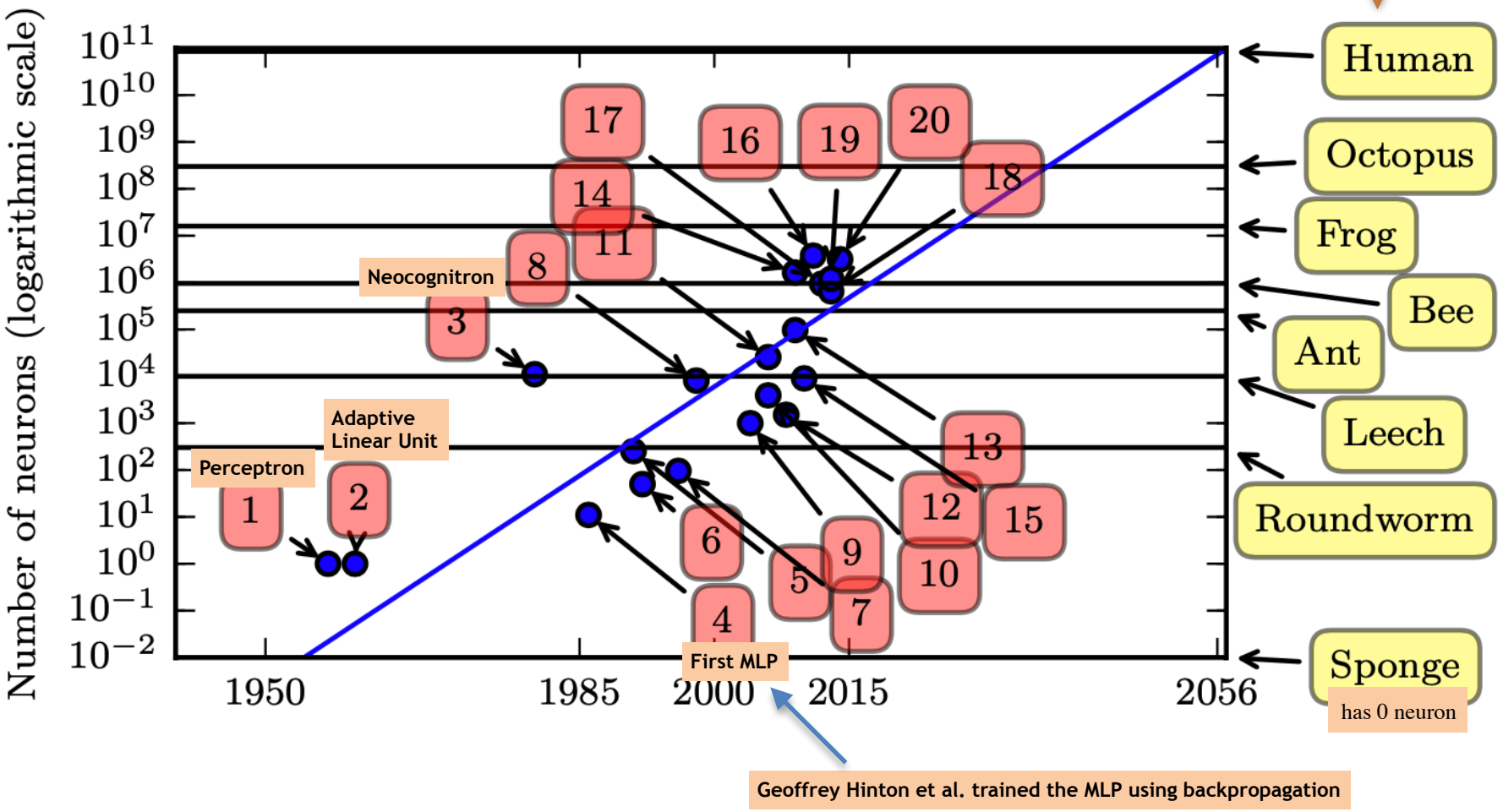
- We created our first artificial neural network (ANN)
- This type of connectivity structure is popularly known as multilayer perceptron (MLP)
- Any layers in between **input layer** and output layer are called hidden layers
- Hence this MLP can also be called 1-hidden layer neural network



Neurons in the input layer do not have activation functions

Input Layer | **Hidden Layer** | **Output Layer**

Neural Network sizes over time



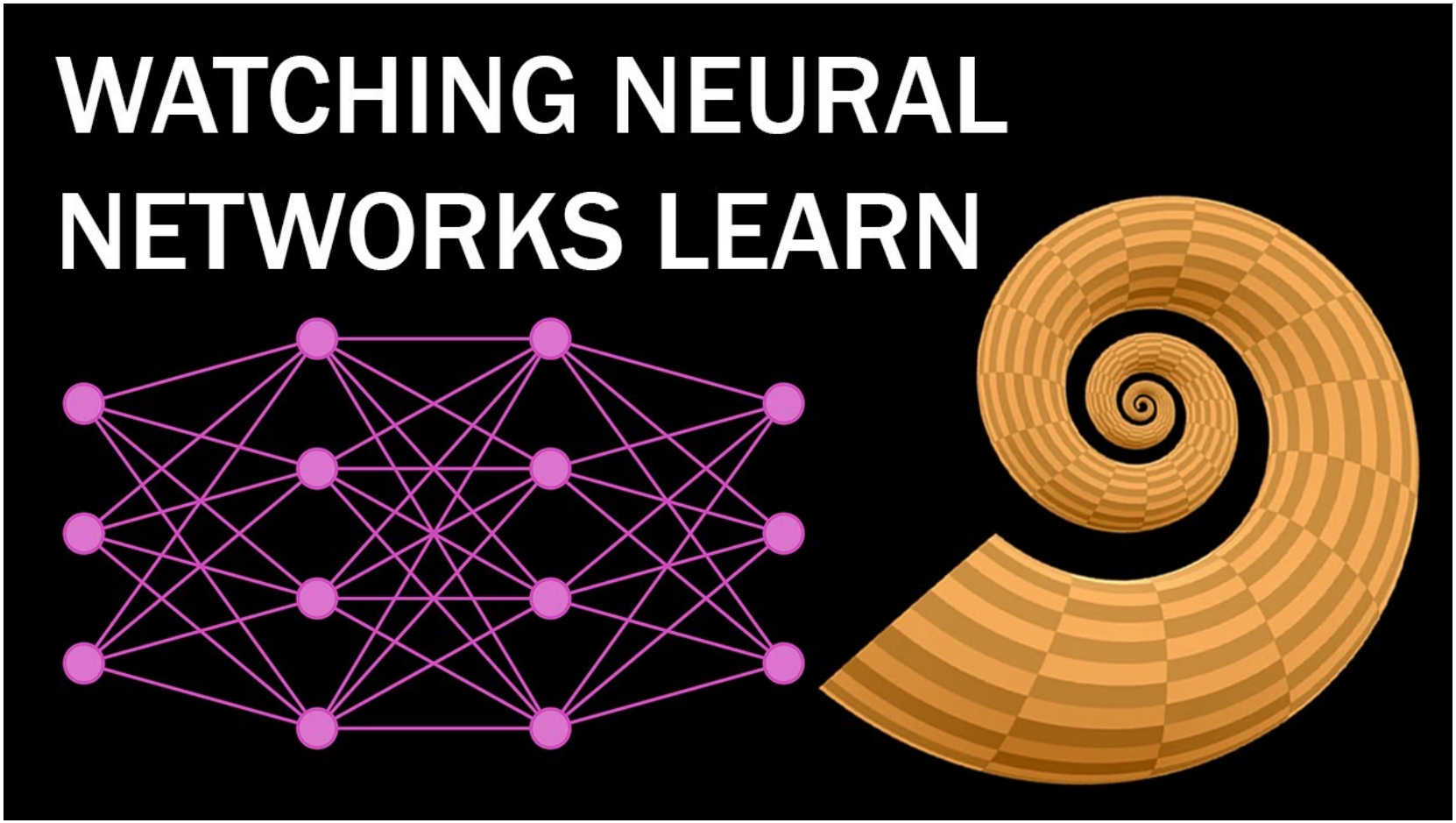
- ANNs differs from biological brains in many ways

The importance of depth in neural network

- One can show that neural network with 1-hidden layer **is a universal function approximators**[1], meaning it can model any suitably smooth function, given enough hidden units, to any desired level of accuracy
  - Intuitive explanation:
    - each hidden unit can specify a half plane, and a sufficiently large combination of these can "carve up" any region of space, to which we can associate any response

1. K. Hornik. "**Approximation Capabilities of Multilayer Feedforward Networks**". In: Neural Networks 4.2 (1991), pp. 251–257

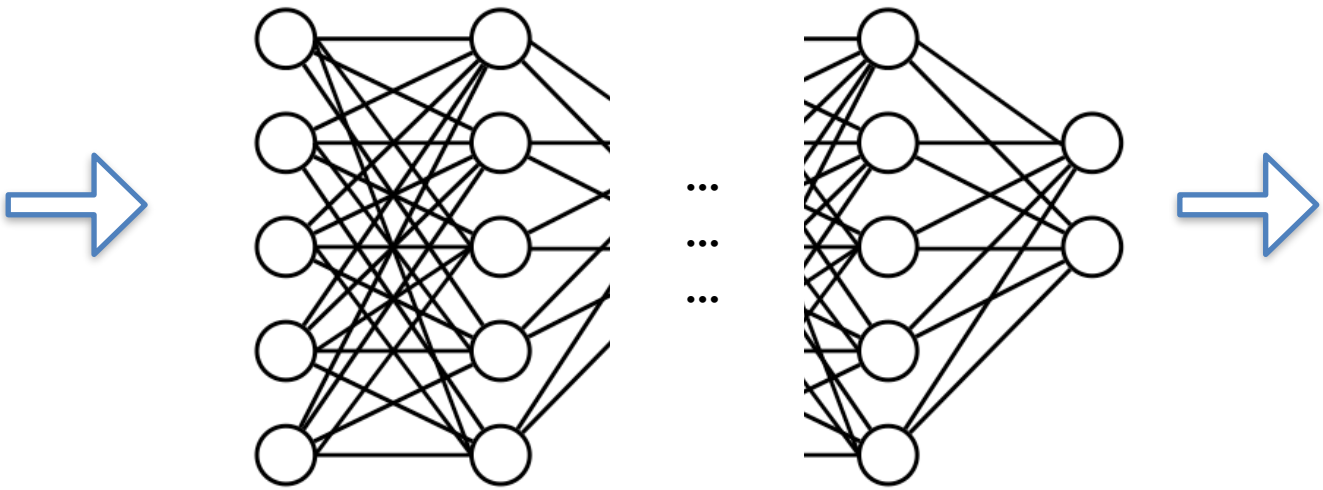
Neural Networks as Universal Function Approximators



Reference: https://www.youtube.com/watch?v=TkwXa7Cvfr8&source_ve_path=MjM4NTE&feature=emb_title

Neural Networks as Universal Function Approximators

- MLPs, neural networks in general, are *universal function approximators*
  - given any function, and a complicated enough network, they can accurately model that function



Group Activity

- Challenge: Can you devise an algorithmic means for classifying whether a photo contains a Dog using MLP?

Hint: an image is a collection of pixels arranged in a grid-like structure. Let's say there are 50x50 pixels in the left image

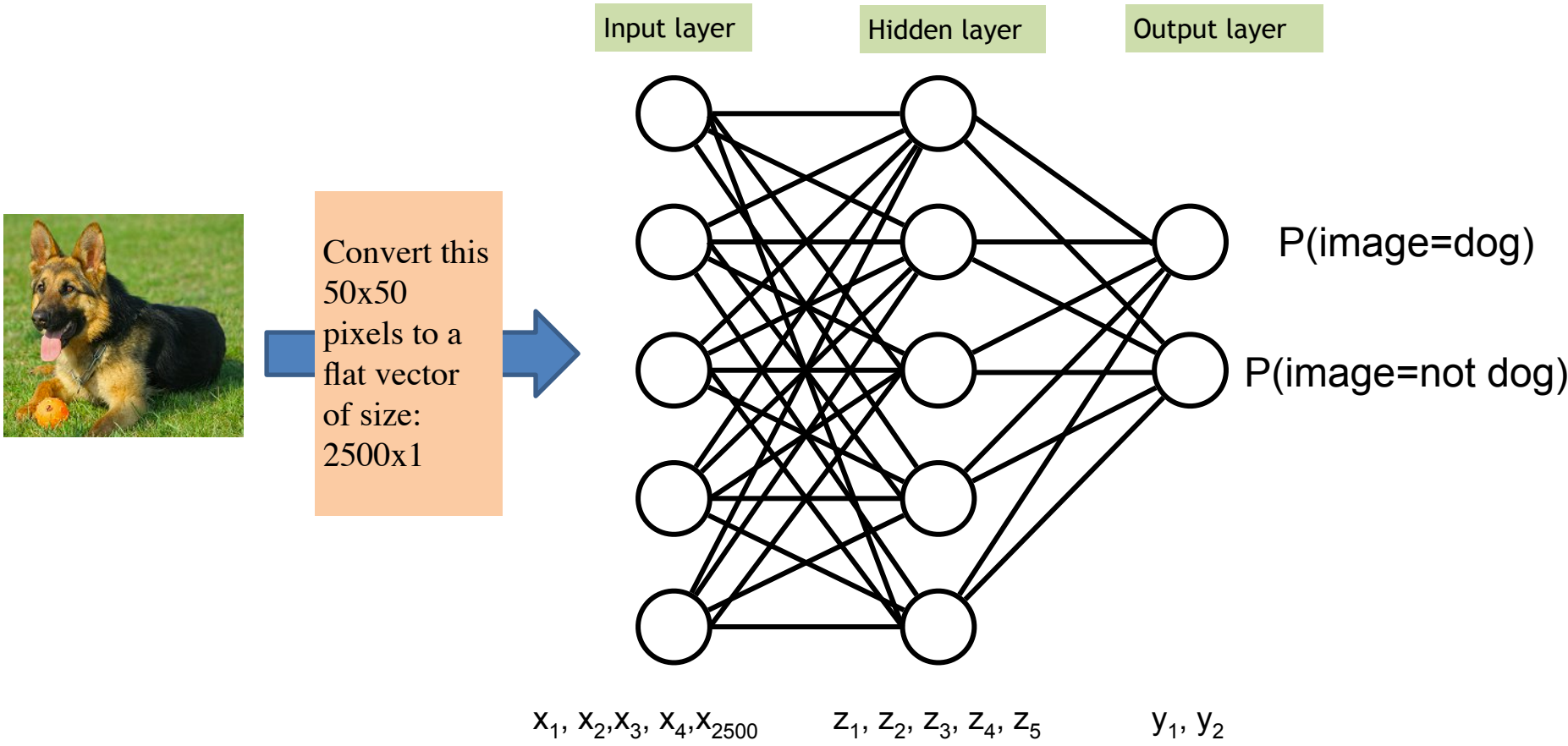
Today's Agenda

- Connections with biology: natural neurons vs. artificial neurons

- Multilayer Perceptrons (MLP)

Formal Name: Multilayer Perceptron (MLP)

- A **multilayer perceptron** is the simplest type of neural network. It consists of perceptrons (aka nodes, neurons) arranged in layers

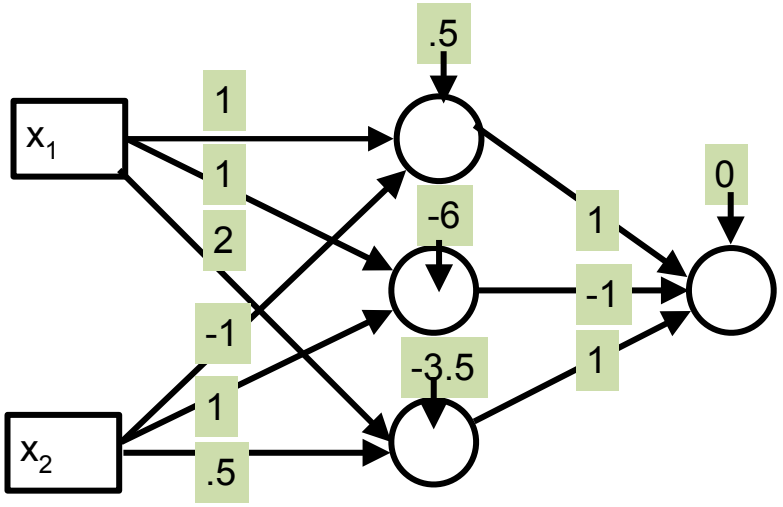


Forward Pass in Multilayer Perceptron

- A **multilayer perceptron** is the simplest type of neural network. It consists of perceptrons (aka nodes, neurons) arranged in layers

- Each neuron contains two operations:
  - a dot product between *a weight vector (edges in the graph)* and *an input vector*, which produces a number
  - Then, that number through an activation function, which produces a number as an output

Sample#	x_1	x_2
1	3	5
2	2	7
3	1	1
4	2	3



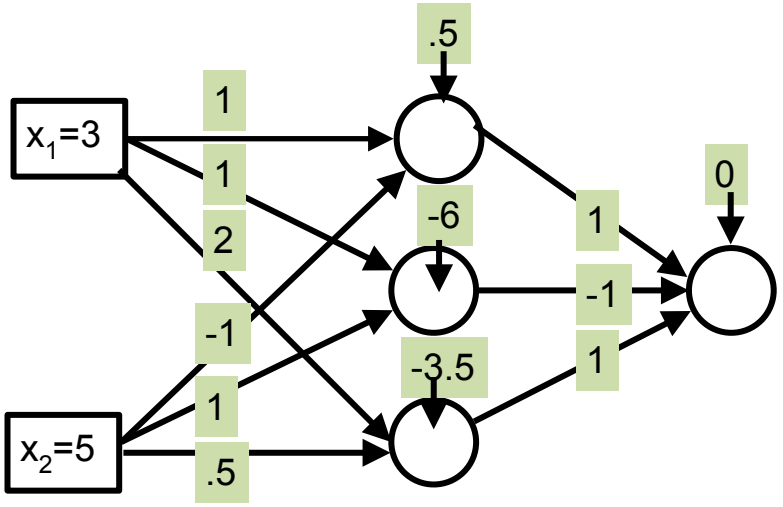
Forward Pass in Multilayer Perceptron

- A **multilayer perceptron** is the simplest type of artificial neural network (ANN). It consists of perceptrons (aka nodes, neurons) arranged in layers

- Each neuron contains two operations:
  - a dot product between *a weight vector (edges in the graph)* and *an input vector*, which produces a number
  - Then, that number through an activation function, which produces a number as an output

Sample#	x_1	x_2
1	3	5
2	2	7
3	1	1
4	2	3

Consider doing forward pass for this example



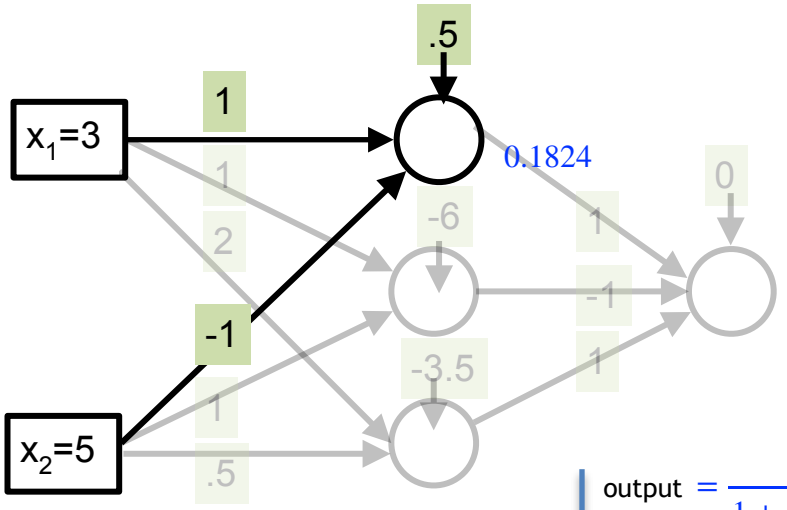
Forward Pass in Multilayer Perceptron

- A **multilayer perceptron** is the simplest type of artificial neural network (ANN). It consists of perceptrons (aka nodes, neurons) arranged in layers

- Each neuron contains two operations:
  - a dot product between *a weight vector (edges in the graph)* and *an input vector*, which produces a number
  - Then, that number through an activation function, which produces a number as an output

Sample#	x_1	x_2
1	3	5
2	2	7
3	1	1
4	2	3

Consider doing forward pass for this example



$$\mathbf{w^T x} = [w_0 \quad w_1 \quad w_2] \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} = [0.5 \quad 1 \quad -1] \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} = (1 * 0.5 + 1 * 3 + (-1) * 5)$$
$$= -1.5$$

$$\text{output} = \frac{1}{1 + exp^{-\mathbf{w^T x}}}$$
$$= \frac{1}{1 + exp^{-(-1.5)}}$$
$$= 0.1824$$

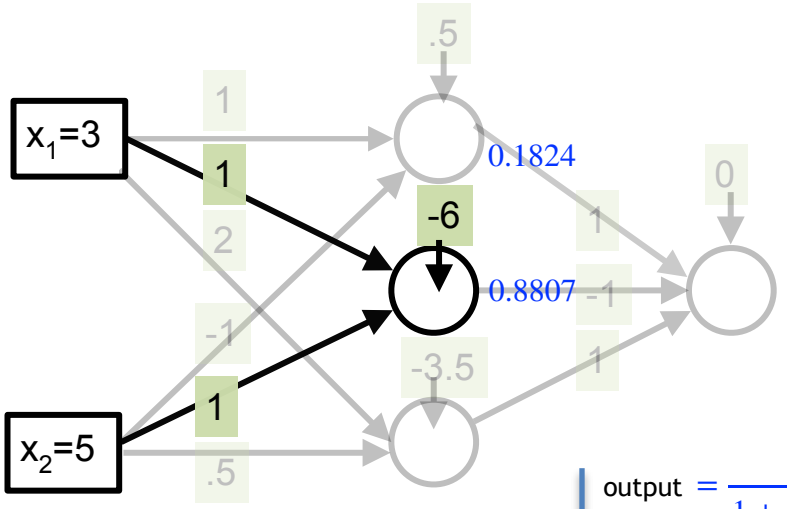
Forward Pass in Multilayer Perceptron

- A **multilayer perceptron** is the simplest type of artificial neural network (ANN). It consists of perceptrons (aka nodes, neurons) arranged in layers

- Each neuron contains two operations:
  - a dot product between *a weight vector (edges in the graph)* and *an input vector*, which produces a number
  - Then, that number through an activation function, which produces a number as an output

Sample#	x_1	x_2
1	3	5
2	2	7
3	1	1
4	2	3

Consider doing forward pass for this example



$$\mathbf{w^T x} = [w_0 \quad w_1 \quad w_2] \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} = [-6 \quad 1 \quad 1] \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} = (-6) * 1 + 1 * 3 + 1 * 5 = 2$$

$$\text{output} = \frac{1}{1 + exp^{-\mathbf{w^T x}}} = \frac{1}{1 + exp^{-2}} = 0.8807$$

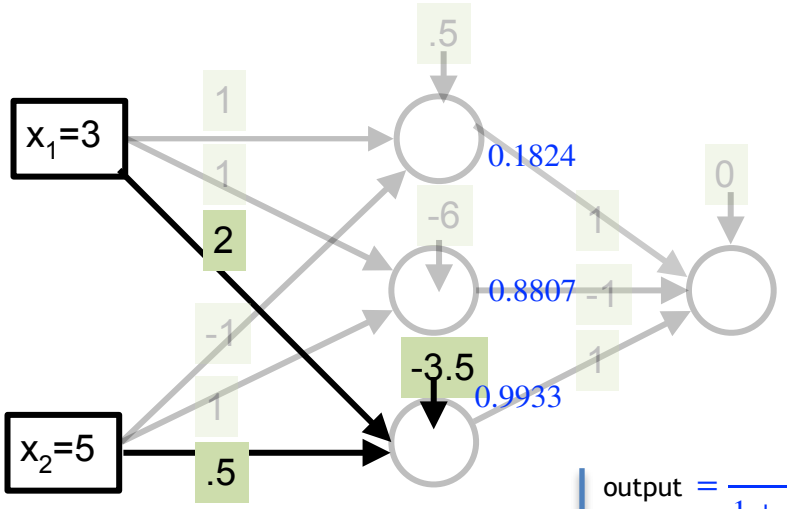
Forward Pass in Multilayer Perceptron

- A **multilayer perceptron** is the simplest type of artificial neural network (ANN). It consists of perceptrons (aka nodes, neurons) arranged in layers

- Each neuron contains two operations:
  - a dot product between *a weight vector (edges in the graph)* and *an input vector*, which produces a number
  - Then, that number through an activation function, which produces a number as an output

Sample#	x_1	x_2
1	3	5
2	2	7
3	1	1
4	2	3

Consider doing forward pass for this example



$$\mathbf{w^T x} = [w_0 \quad w_1 \quad w_2] \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} = [-3.5 \quad 2 \quad 0.5] \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} = (-3.5) * 1 + 2 * 3 + 0.5 * 5 = 5$$

$$\text{output} = \frac{1}{1 + exp^{-\mathbf{w^T x}}} = \frac{1}{1 + exp^{-5}} = 0.9933$$

MLP Forward Pass Group Exercise

- Each neuron contains two operations:
  - a dot product between *a weight vector (edges in the graph)* and *an input vector*
  - that number through an activation function, which produces a number as an output

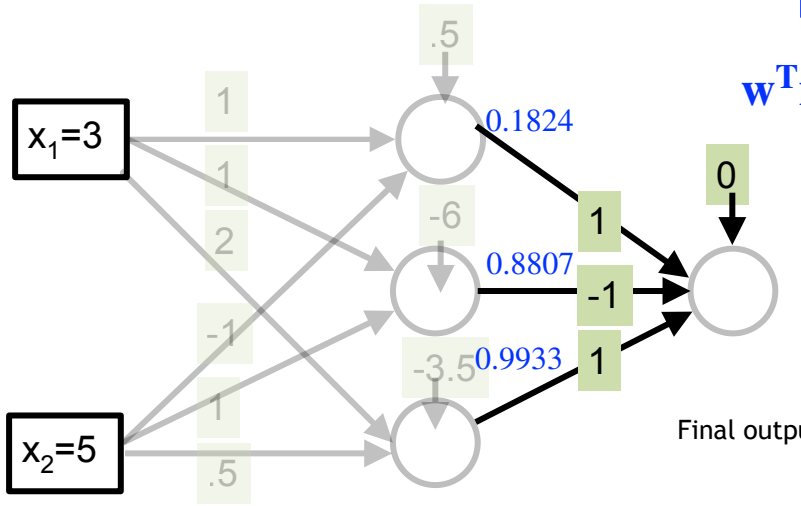
$$\mathbf{w}^T = [\,?\quad ?\quad ..\quad ..\quad ..\,] = ?$$

$$\mathbf{x} = \begin{bmatrix} 1 \\ ? \\ ? \\ \vdots \\ \vdots \end{bmatrix} = ?$$

$$\mathbf{w}^T\mathbf{x} = ?$$

Sample#	x_1	x_2
1	3	5
2	2	7
3	1	1
4	2	3

Consider doing forward pass for this example



Final output $= \dfrac{1}{1 + exp^{-\mathbf{w}^T\mathbf{x}}}$

$= \dfrac{1}{1 + exp^{-?}}$

$= ?$

MLP Forward Pass Group Exercise Answer

- Each neuron contains two operations:
  - a dot product between *a weight vector (edges in the graph)* and *an input vector*
  - that number through an activation function, which produces a number as an output

$$\mathbf{w^T} = [w_0 \quad w_1 \quad w_2 \quad w_3] = [1 \quad 0.1824 \quad 0.8807 \quad 0.9933]$$

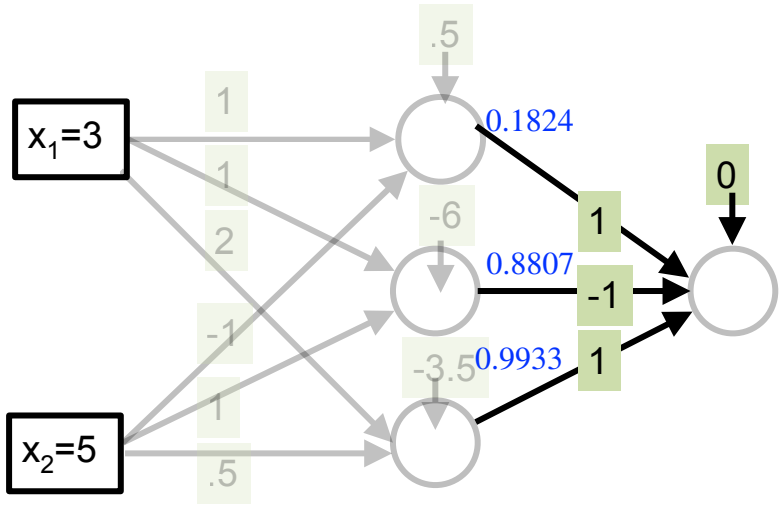
$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 1.0 \\ -1 \\ 1.0 \end{bmatrix}$$

$$\mathbf{w^T x} = 1*0 + 0.1824*1 + 0.8807*(-1) + 0.9933*1 = 0.295$$

Final output $= \frac{1}{1+exp^{-\mathbf{w^T x}}} = \frac{1}{1+exp^{-0.295}} = 0.5732$

Sample#	x_1	x_2
1	3	5
2	2	7
3	1	1
4	2	3

Consider doing forward pass for this example



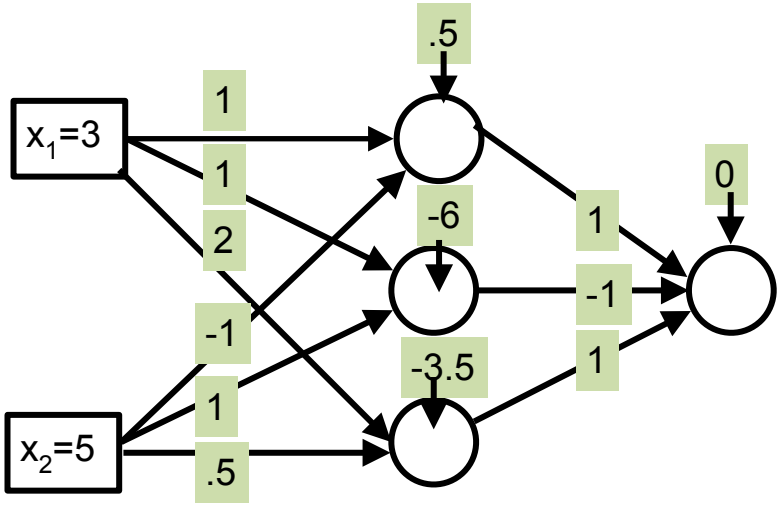
Multilayer Perceptron

- A **multilayer perceptron** is the simplest type of neural network. It consists of perceptrons (aka nodes, neurons) arranged in layers

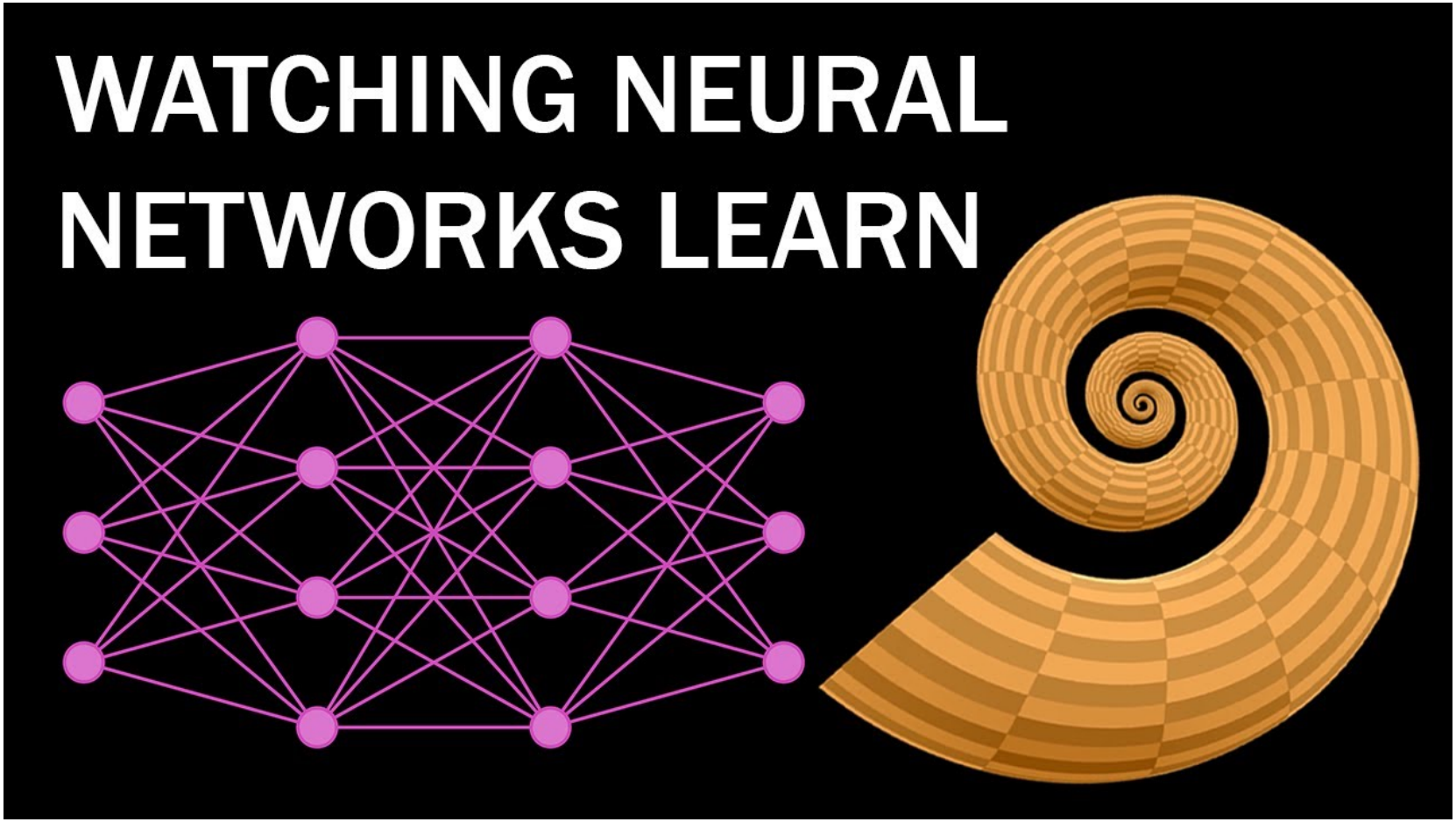
- Each neuron contains two operations:
  - a dot product between *a weight vector (edges in the graph)* and *an input vector*, which produces a number
  - Then, that number through an activation function, which produces a number as an output

Sample#	x_1	x_2
1	3	5
2	2	7
3	1	1
4	2	3

You can do the same forward pass for the next examples sequentially



Neural Network Architecture



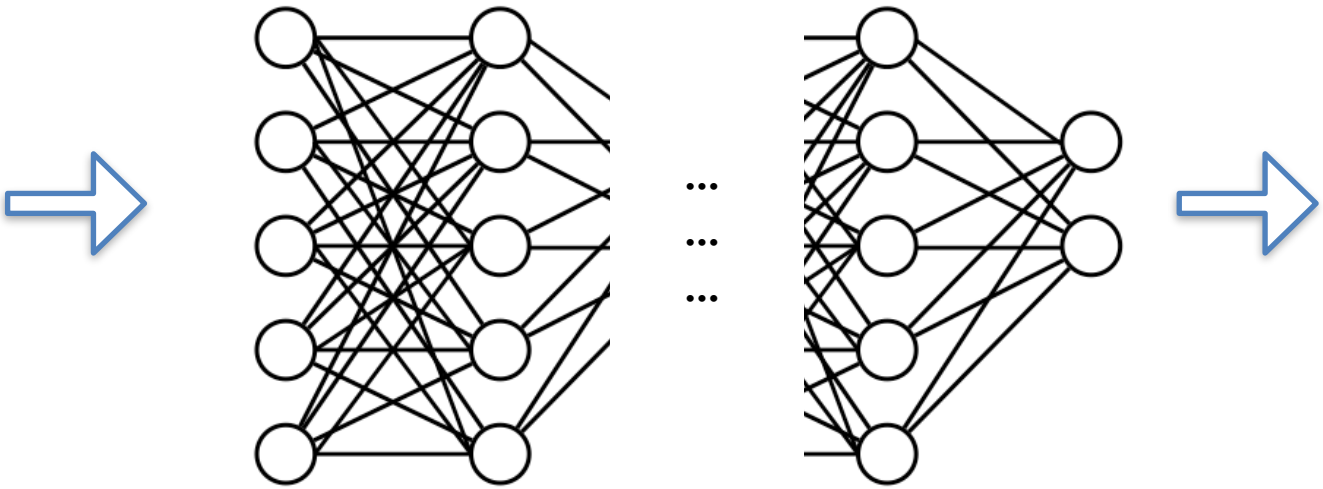
Reference: https://www.youtube.com/watch?v=TkwXa7Cvfr8&source_ve_path=MjM4NTE&feature=emb_title

Today's Agenda

- Connections with biology: natural neurons vs. artificial neurons
- Multilayer Perceptrons (MLP)
- MLP Structure

MLP (Network) Structure

- Each of these questions need to be answered before you set up your neural network:
  - how many hidden layers should I have? (depth)
  - how many neurons should be in each layer? (width)
  - what should your activation be at each of the layers?



The importance of depth in neural network

- One can show that MLPs with one hidden layer is a universal function approximators, meaning it can model any suitably smooth function, given enough hidden units, to any desired level of accuracy

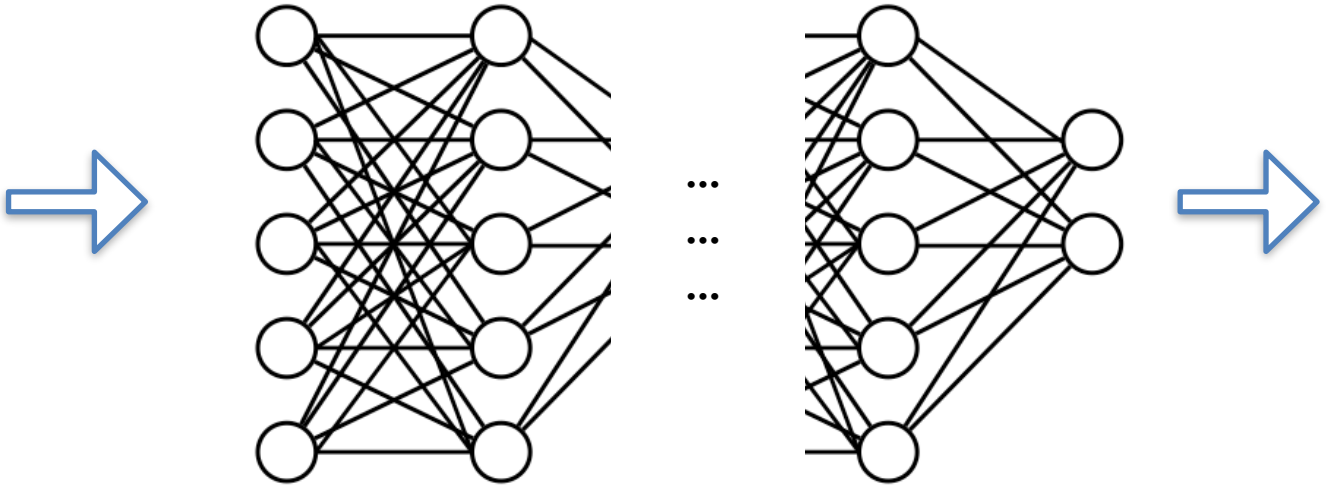
  - Theoretic and experimental evidences[1]:
    - deep networks work better than shallow ones. The reason is that later layers can leverage the features that are learned by earlier layers; that is, the function is defined in a compositional or hierarchical way.

    - Eg, suppose we want to classify DNA strings, and the positive class is associated with the string AACGCGAA. Although we could fit this with a single hidden layer model, intuitively it will be easier to learn if the model first learns to detect the AA and CG "motifs" using the hidden units in layer#1, and then uses these features to define a simple linear classifier in layer#2

1. **T. Poggio**, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. "Why and when can deep- but not shallow-networks avoid the curse of dimensionality: A Review". en. In: International Journal Automation and Computation (2017)

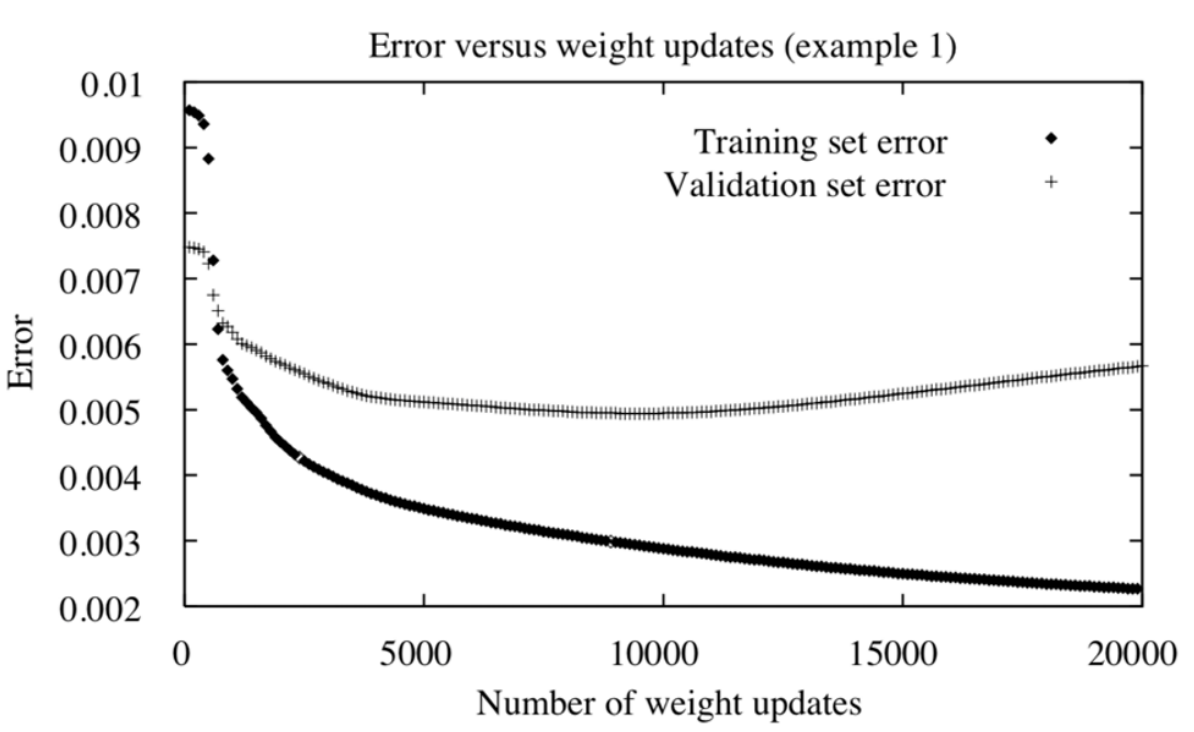
MLP (Network) Structure

- How to choose the size and structure of networks?
  - If network is too large, risk of over-fitting (data caching)
  - If network is too small, representation may not be rich enough



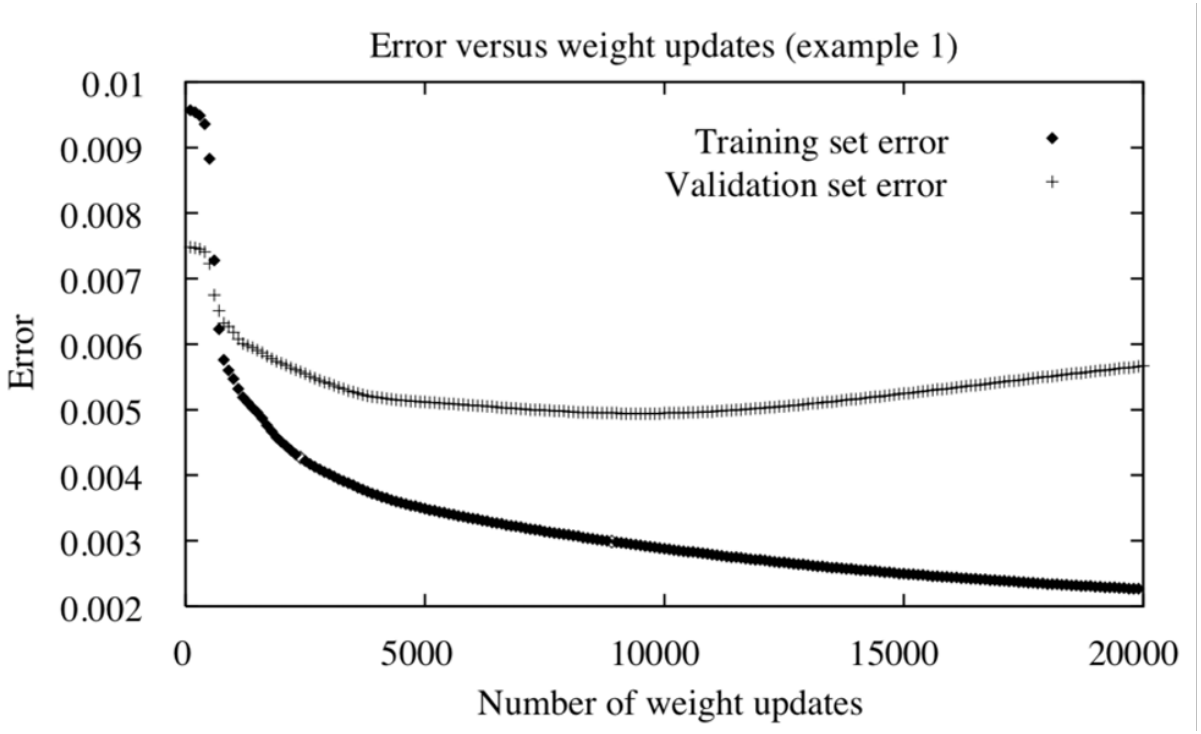
Overfitting

- MLPs, like many machine learning models, are susceptible to **overfitting**
  - How can we recognize overfitting?
  - Given the graph below, at what point do you think our model started overfitting?



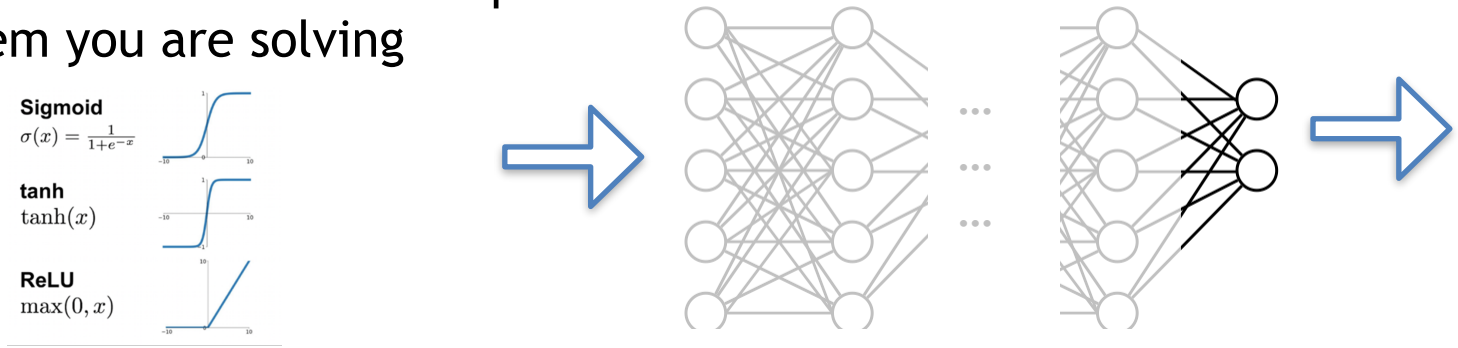
Overfitting

- **Overfitting** happens when the *training set error* continues to improve, but the *validation (testing) set error* starts to worsen (increase).
  - So... how do we know when to stop training our model to avoid overfitting?



Final Output Nodes

- In general, the complexity of your network should match the complexity of your problem. The final output nodes should be related to what kind of problem you are solving



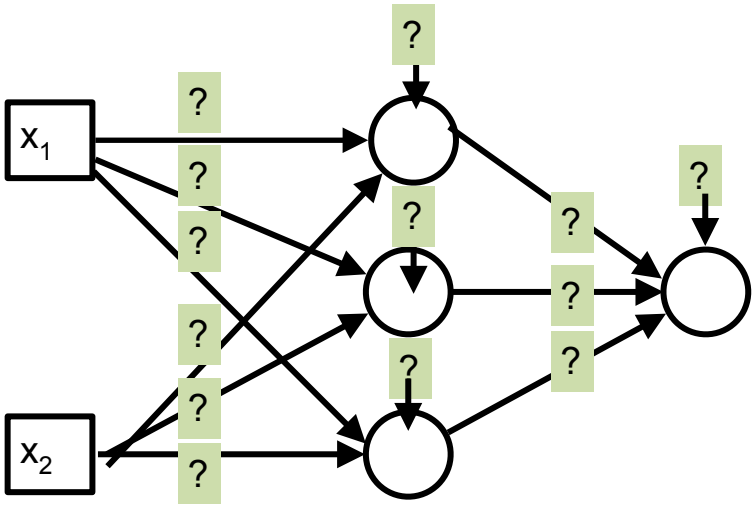
Activation Function	Function	Lower bound	Upper bound	Type of Machine Learning
Linear	$f(z) = az$	$-\infty$	∞	regression where results can be negative
Rectified Linear Unit (ReLU)	$relu(z) = max(0, z)$	0	∞	regression where results can't be negative
Sigmoid	$sigmoid(z) = \frac{1}{1+e^{-z}}$	0	1	binary classification
Softmax	$softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)}$	0	1	multiclass classification

Today's Agenda

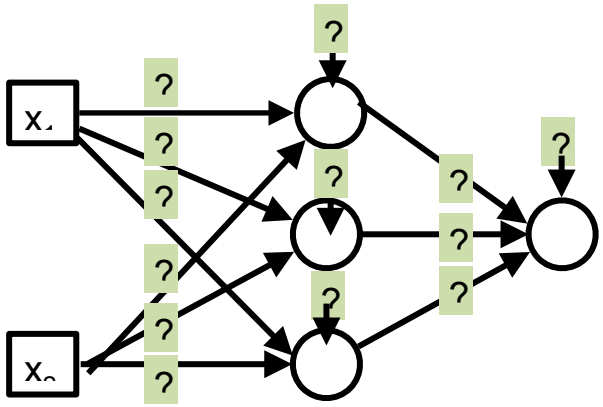
- Connections with biology: natural neurons vs. artificial neurons
- Multilayer Perceptrons (MLP)
- MLP Structure
- Learning MLP Weight Parameters
  - Recap from last week's offline lecture
  - Trainable parameters and their learnable weights

Training to Learn MLP (Network) Structure Parameters

- The trainable parameters are the *weights (w's)* which are learned from the training data



Training to Learn MLP (Network) Structure Parameters



- The goal is to **minimize the error** predicted by the network (from last lecture) from the training data
  - Gradient Descent
  - Stochastic Gradient descent

- Gradient Descent
  - calculate the gradient vector based on that batch $\nabla E(\mathbf{w})$
  - adjust (or update) the values of the weights based on the gradient vector to that batch

$$\mathbf{w}^{\mathbf{new}} = \mathbf{w}^{\mathbf{old}} - \eta \nabla \mathbf{E}(\mathbf{w})$$

Training to Learn MLP (Network) Structure Parameters

- The specific name for the weight learning algorithm is Backpropagation. It is glorified name but it is gradient descent under the hood.
- It tunes **the weights** over a neural network using **gradient descent** to iteratively reduce the error in the network.

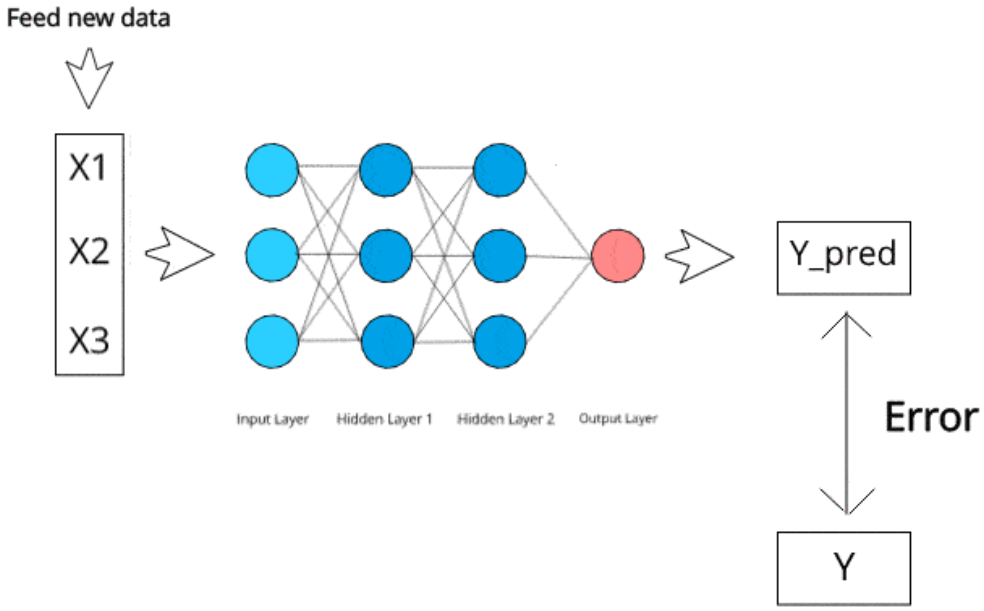
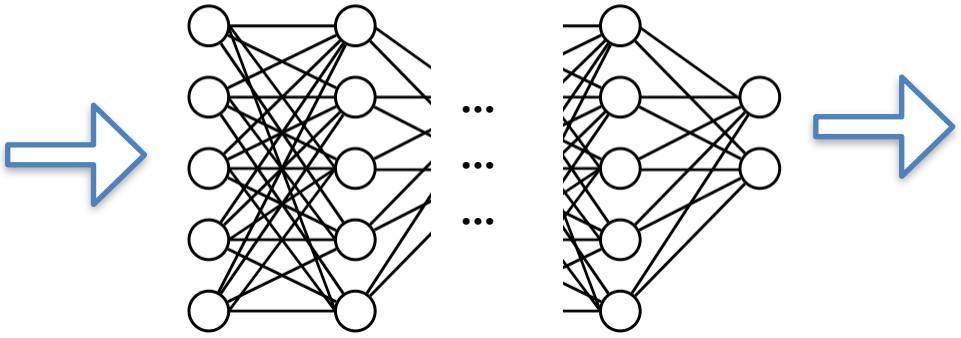


Image reference

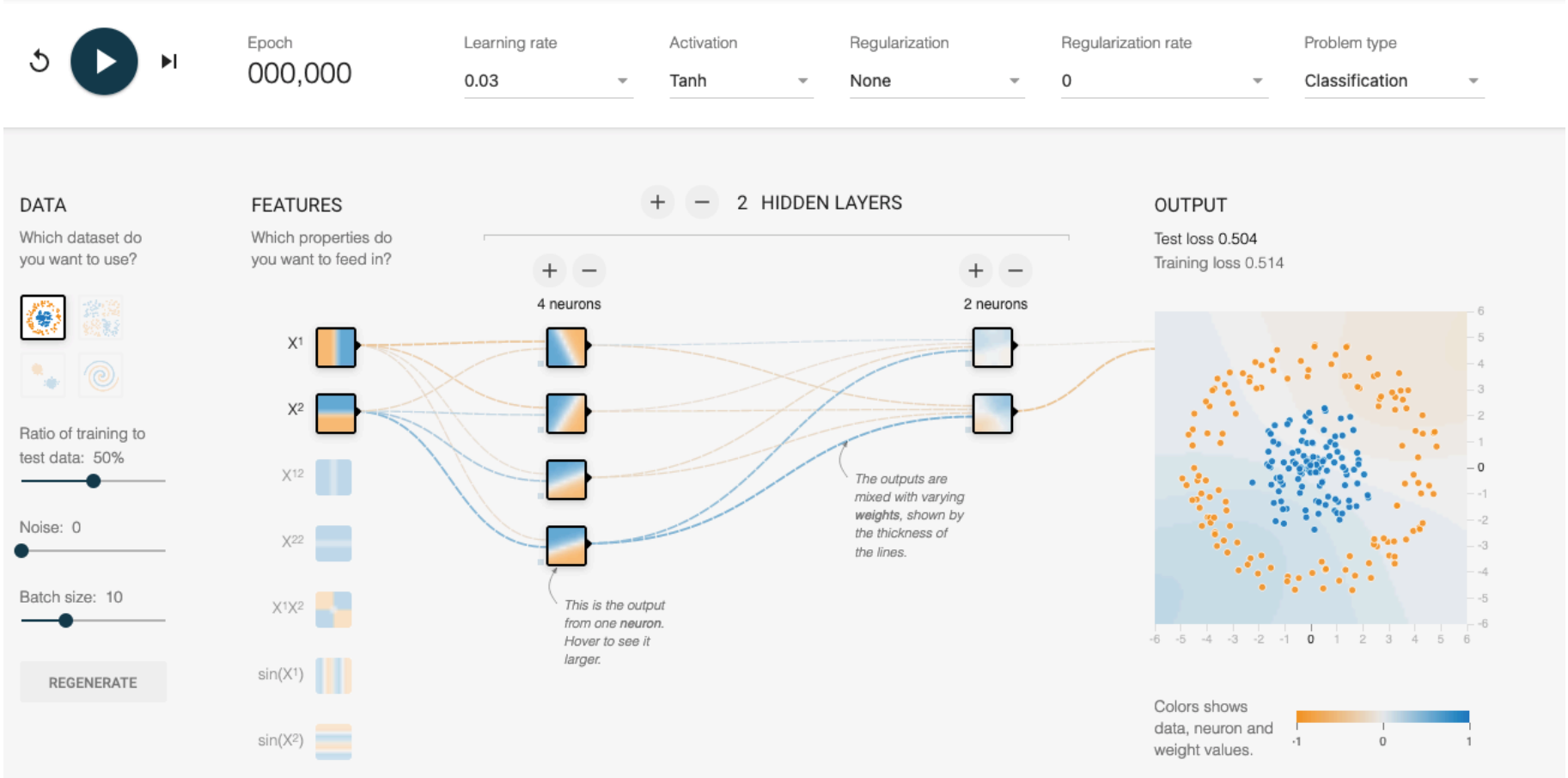
MLP Summary

- MLPs are effective in finding non-linear patterns in the training data
  - can be applied to **regression** or **classification**.
  - **backpropagation** tunes the weights over a neural network using **gradient descent** to iteratively reduce the error in the network
  - **overfitting** the training data is common and is important to avoid
  - the following parameters should be tuned when using MLPs:
    - number of epochs
    - structure of the network (depth, width)
    - activation function
    - eta (learning rate)



Tinker with the Following to See MLP in Action

- MLPs are effective in finding non-linear patterns in the training data



https://playground.tensorflow.org