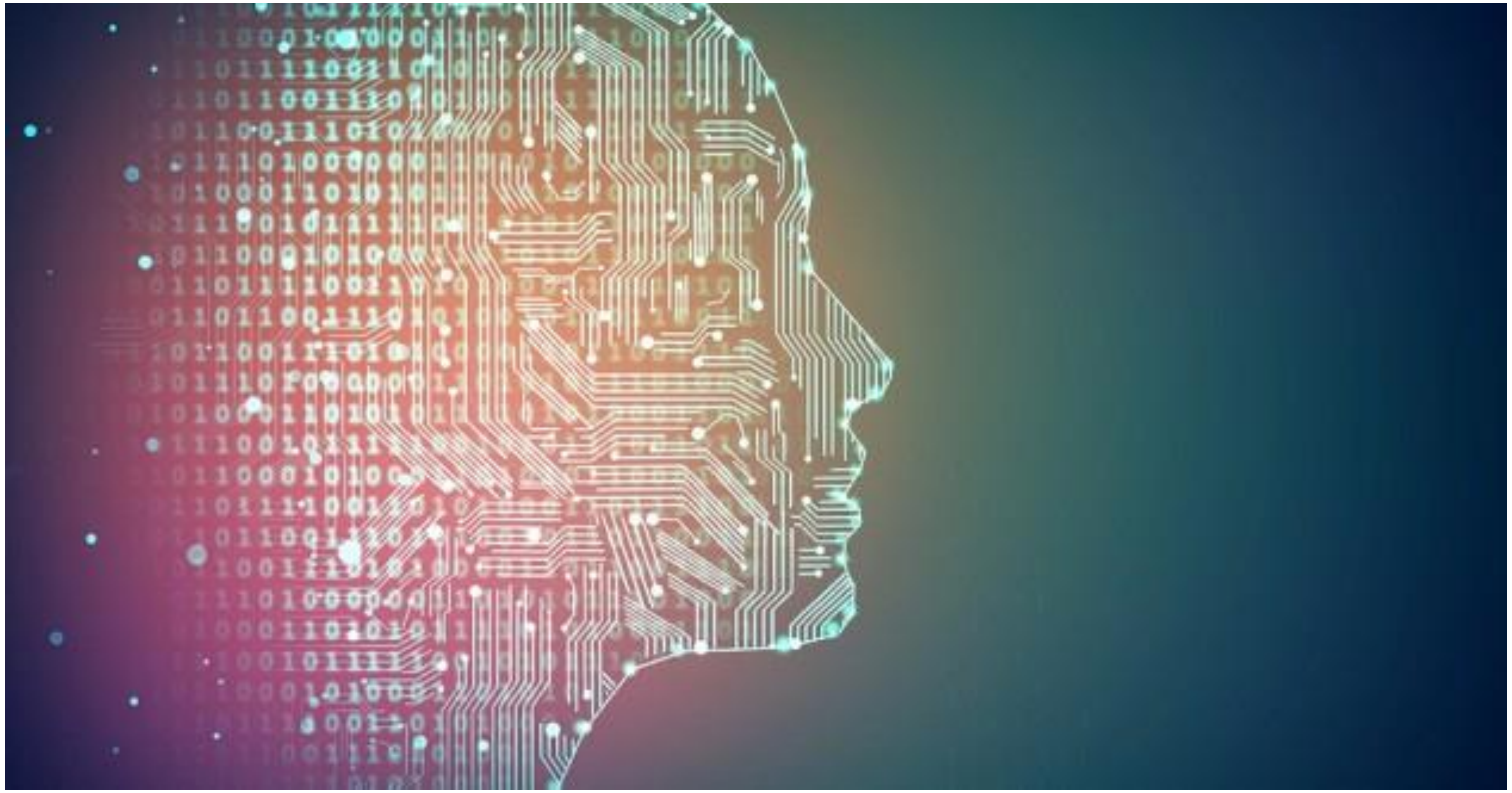
AI Deep Learning: Feedforward Neural Networks

Thuan L Nguyen, PhD

Slide 2: AI Deep Learning: Feedforward Neural Networks (FFNN)



AI Deep learning (Source: mindovermachines.com)

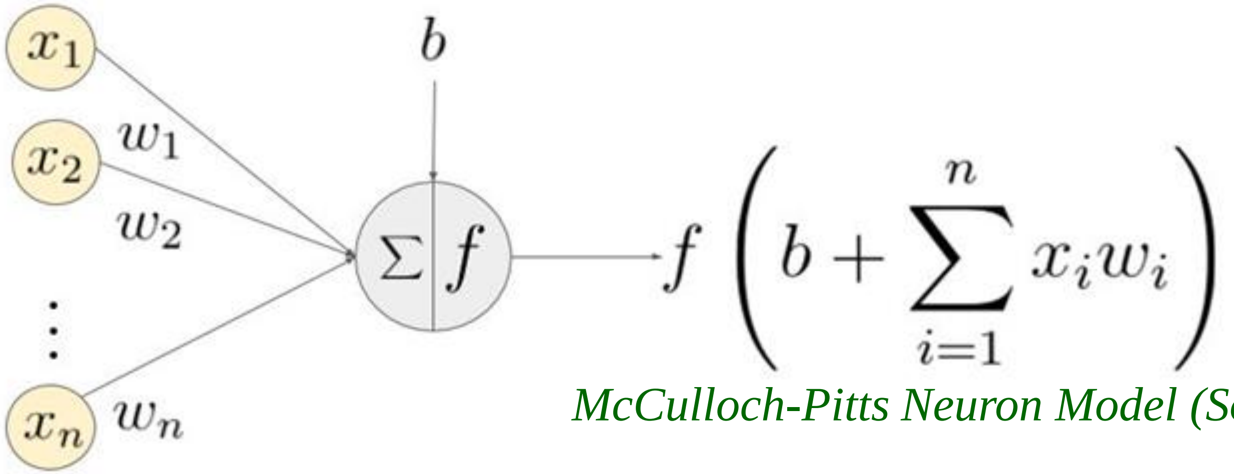
Slide 3: AI Deep Learning: Feedforward Neural Networks (FFNN)

1. Feedforward Neural Networks: Neurons and Perceptron
2. Feedforward Neural Networks: Overview
3. Feedforward Neural Networks: Architecture
4. Feedforward Neural Networks: Gradient-Based Learning
5. Feedforward Neural Networks: Optimization & Cost Function
6. Feedforward Neural Networks: Backpropagation Algorithm

Slide 4: AI Deep Learning: Feedforward Neural Networks (FFNN)

Artificial Neurons: The McCulloch-Pitts Neuron

A **simple rate coding model** of real neurons is also known as a **Threshold Logic Unit** :

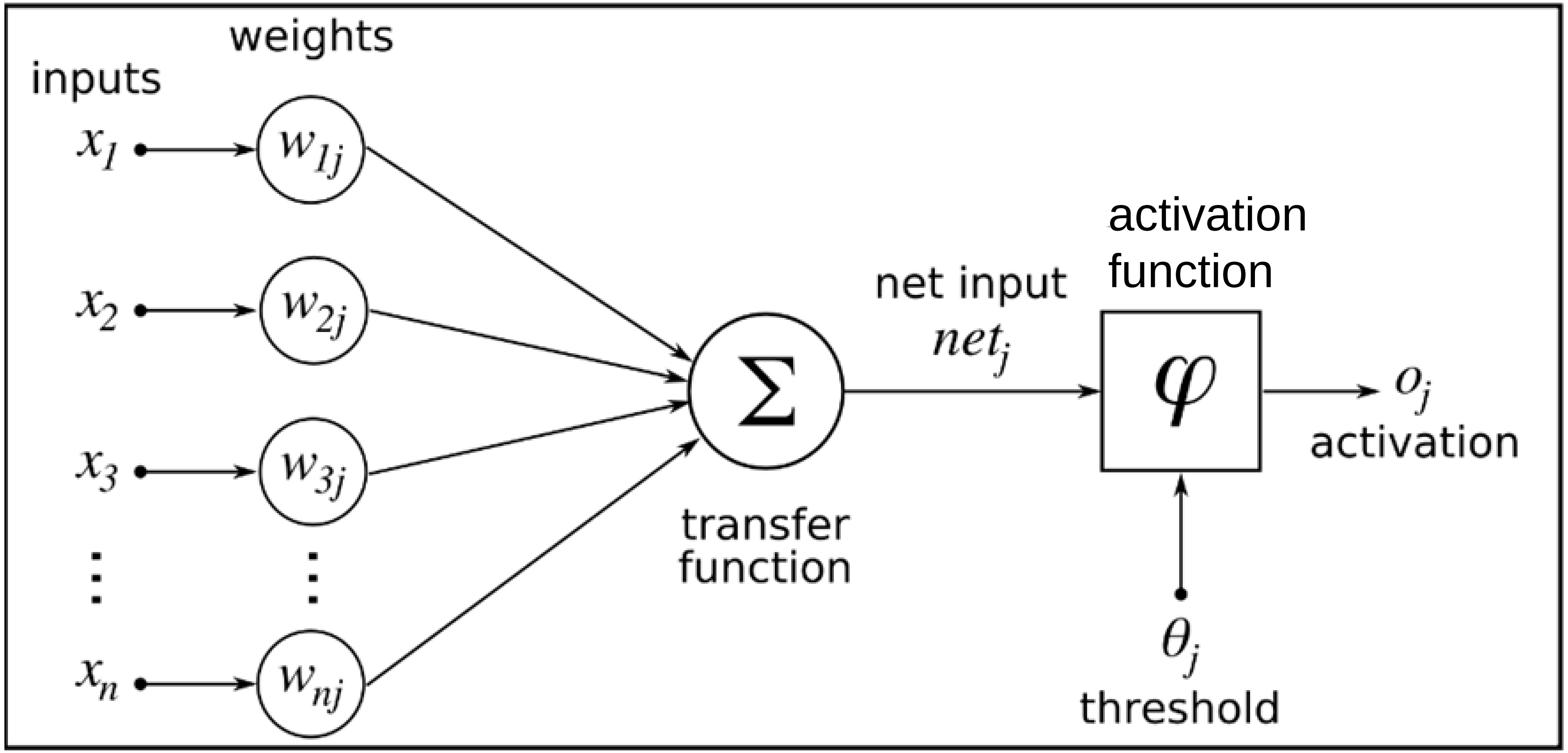


McCulloch-Pitts Neuron Model (Source:Wikipedia.com)

- A set of synapses (i.e. connections) brings in activations, i.e., **inputs**, from other neurons.
- A **processing unit** sums the inputs, and then applies a **non-linear activation function**
  - Is also often called a threshold or transfer or squashing function
- An **output** line transmits the result to other neurons.

Slide 5: AI Deep Learning: Feedforward Neural Networks (FFNN)

Perceptron: A Network of The McCulloch-Pitts Neurons



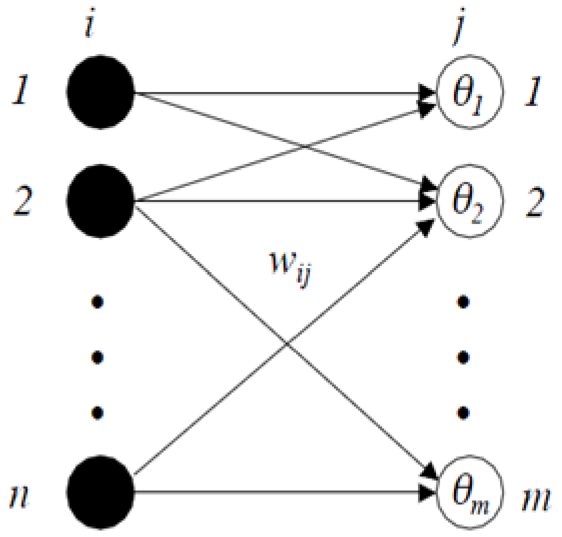
McCulloch-Pitts Neuron model (Sources: wikimedia.org)

Slide 6: AI Deep Learning: Feedforward Neural Networks (FFNN)

Deep Learning: Simple Single-Layer Neural Networks

Perceptron:

- The fundamental unit of an artificial neural network
- A simple – single-layer – artificial neural network:
  - A simple neural network that has one layer of input neurons feeding forward to one output layer of McCulloch-Pitts neurons, with full connectivity.



Perceptron: Single-Layer Neural Network (Sources: Wikipedia)

Slide 7: AI Deep Learning: Feedforward Neural Networks (FFNN)

Perceptron: A Network of The McCulloch-Pitts Neurons

- Frank Rosenblatt introduced the concept of perceptron (1958)
  - Each input I_i is multiplied by a weight w_{ji} (synaptic strength)
  - These weighted inputs are summed to give the activation level, A_j
  - The activation level is then transformed by an activation function to produce the neuron's output, Y_i
  - W_{ji} is known as the weight from unit i to unit j
    - $W_{ji} > 0$, synapse is excitatory
    - $W_{ji} < 0$, synapse is inhibitory
  - Note that I_i may be
    - External input
    - The output of some other neuron

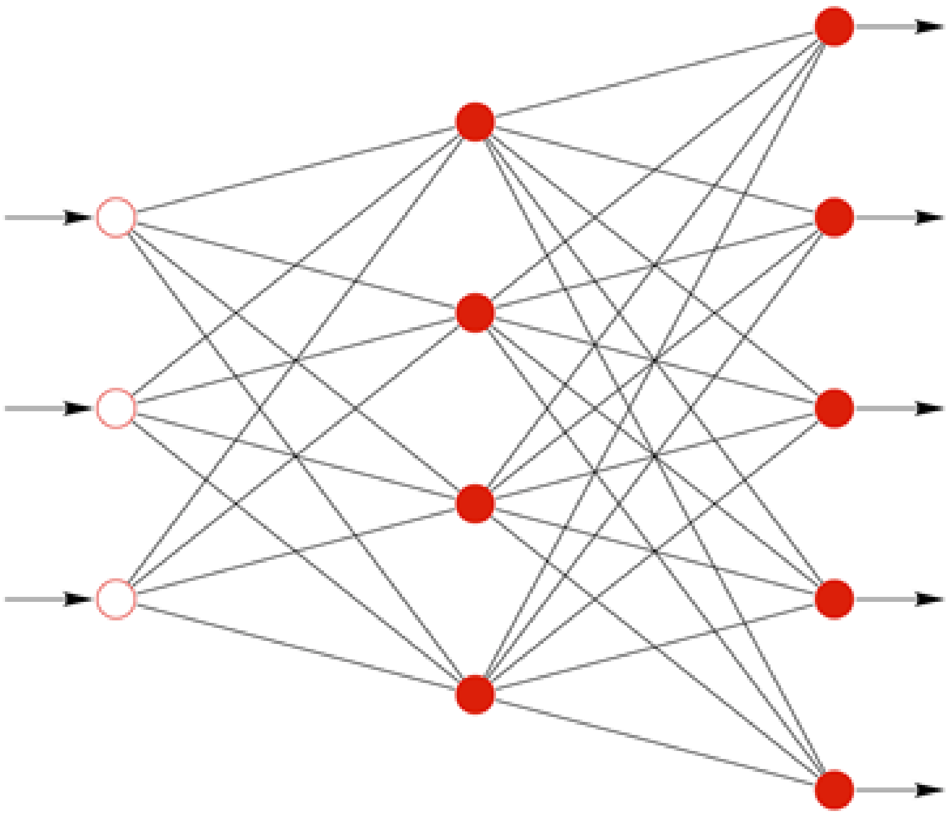
Slide 8: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks (FFNN): Overview

A **feedforward neural network** is a biologically inspired artificial neural network (ANN).

It consist of a number of simple **artificial neurons** processing units, arranged in **layers**.

Every unit in a layer is **connected** with all the units in the previous layer.

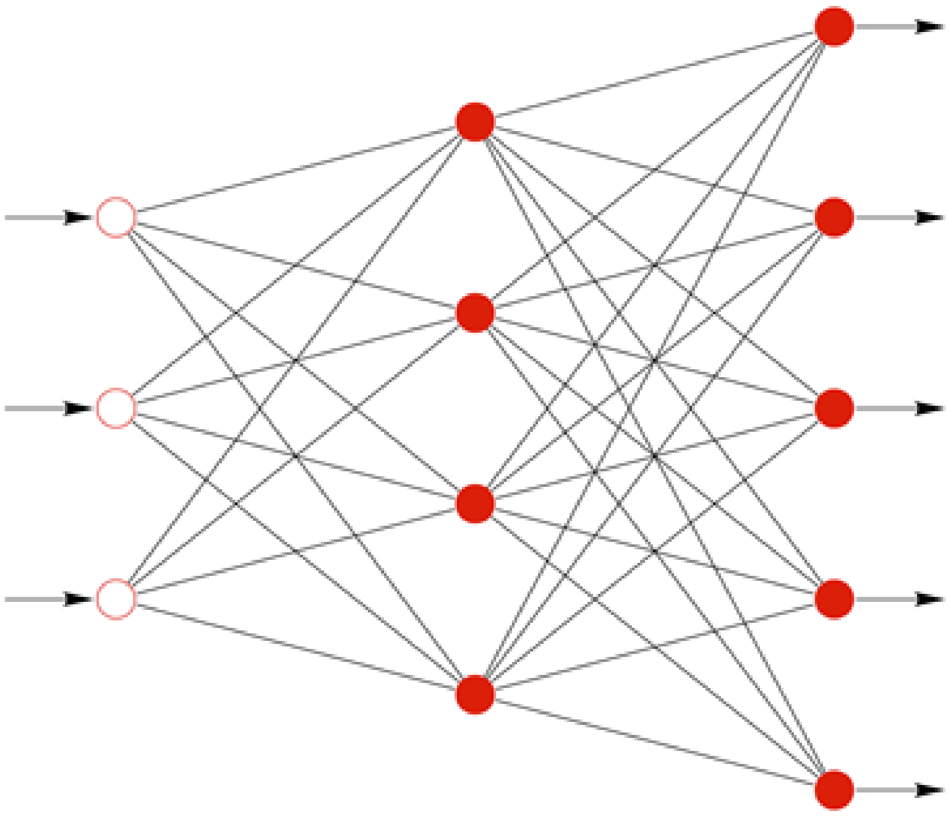


Feedforward Neural Network

Slide 9: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks (FFNN): Overview

- The artificial neuron units in a neural network are called nodes.
- The links between neuron nodes are not all equal: **Each link** may have a different strength or **weight**.
- These **weights** on these links **represent** the **knowledge** of a network.
- Data enters at the inputs and passes through the layers of the network until it arrives at the outputs.
- This network is called **feedforward neural networks** because there is no feedback between layers.



Feedforward Neural Network

Slide 10: AI Deep Learning: Feedforward Neural Networks (FFNN)

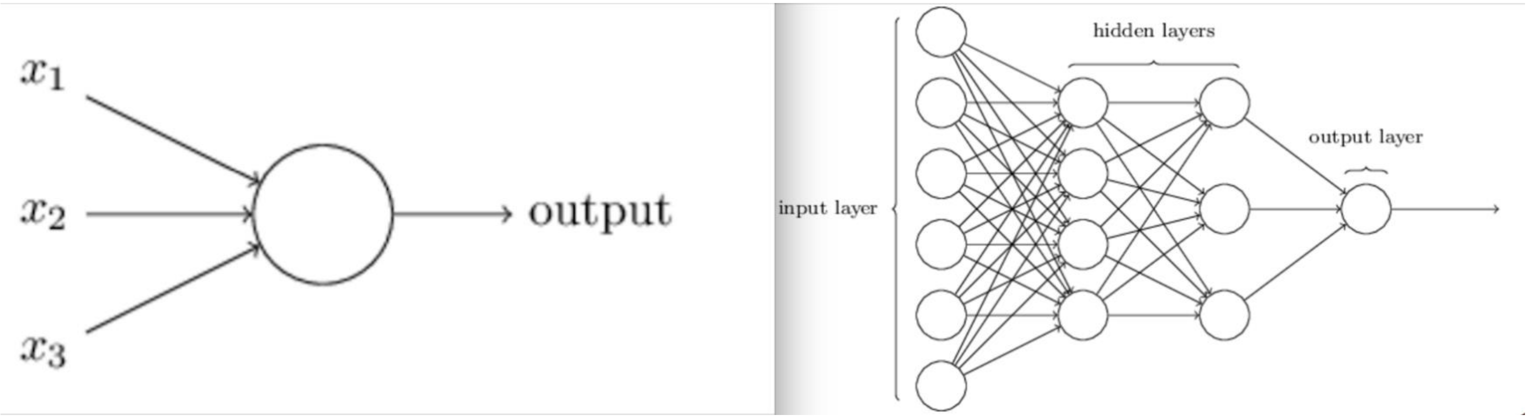
Feedforward Neural Network (FFNN) = Multi-Layer Perceptron (MLP)

Feedforward neural networks are also called

- Deep Feedforward Network

OR

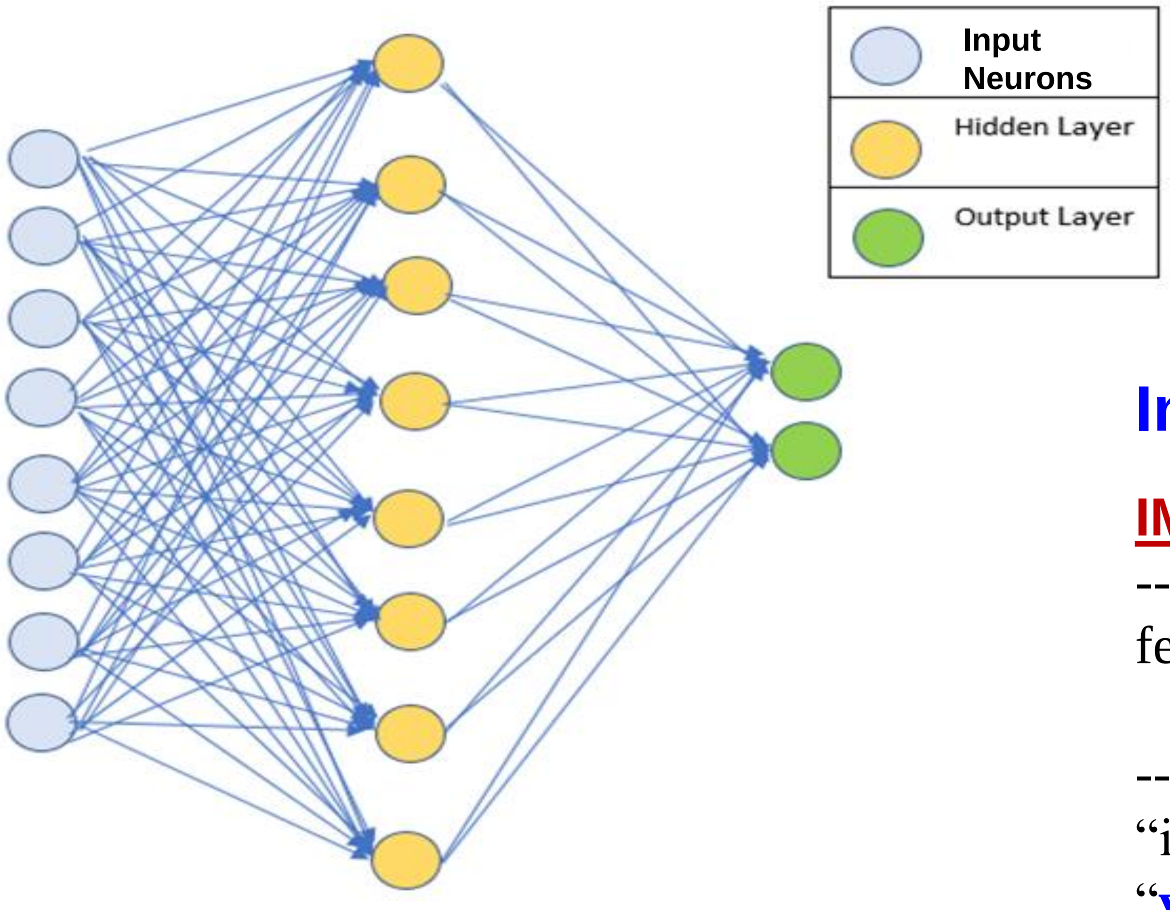
- Multi-Layer Perceptron (MLP)



McCulloch-Pitts neuron model and Feedforward Neural Network (Sources: towardsdatascience.com)

Slide 11: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks (FFNN): Overview



Feedforward Neural Network

Input Neurons:

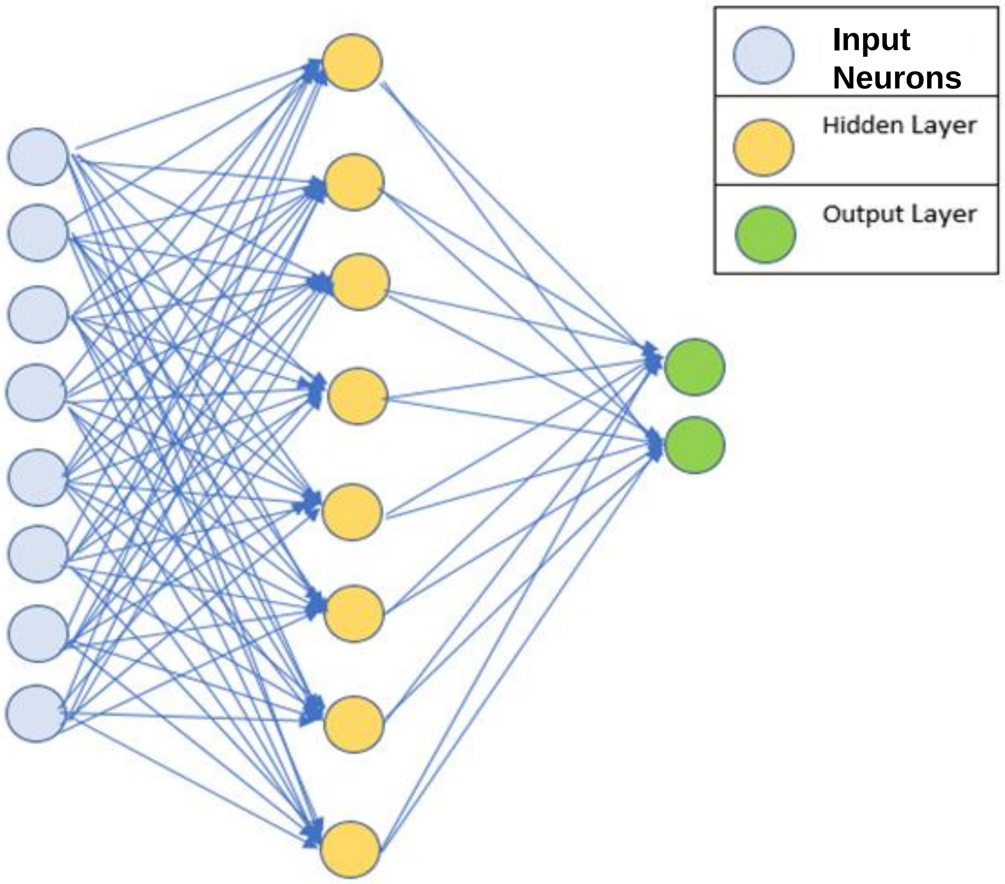
IMPORTANT NOTES:

--) Input neurons **do not** belong to any layer of the feedforward neural network.

--) The set of input neurons is sometimes referred to as the "input layer". However, the input layer is viewed as a "**virtual layer**" and **not counted** as a layer of the network.

Slide 12: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks (FFNN): Overview



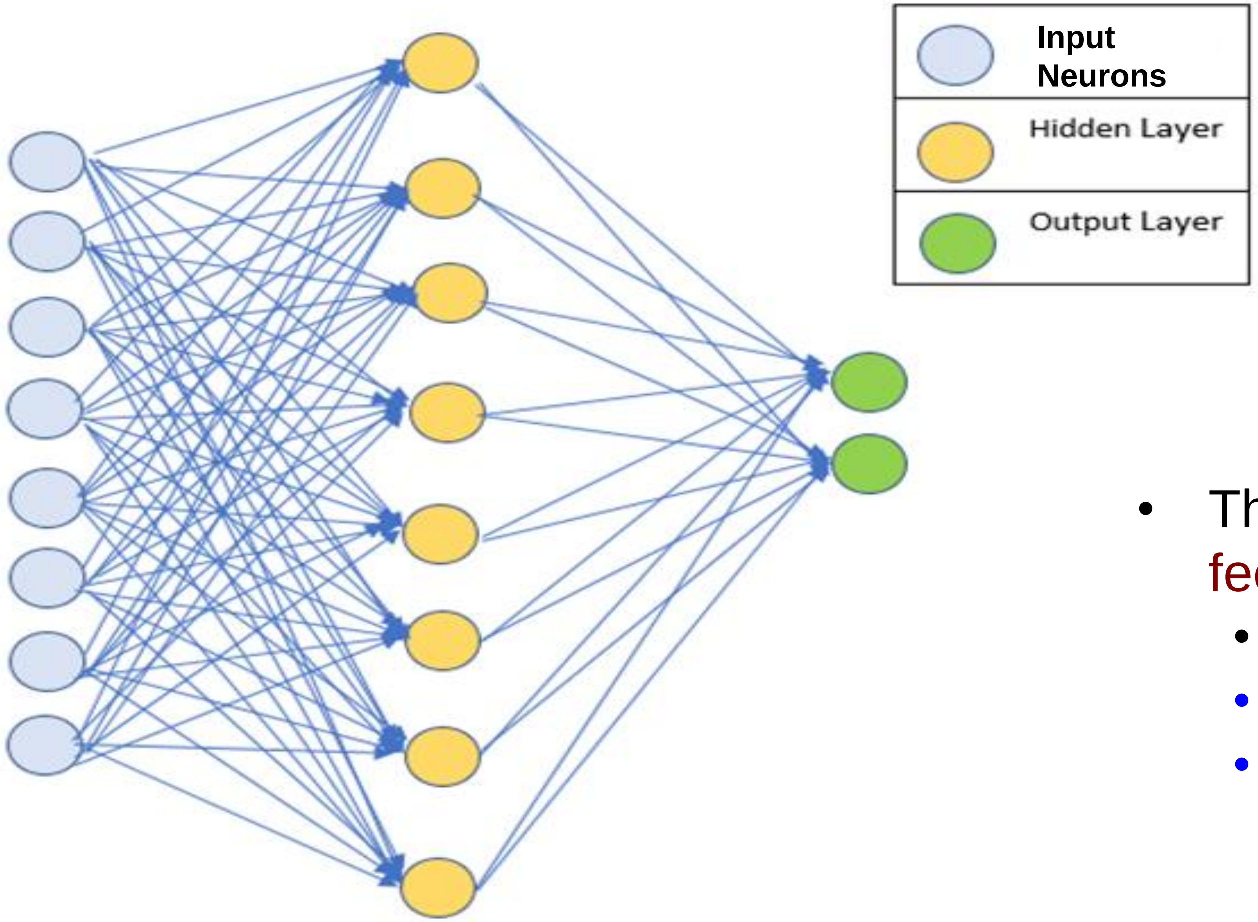
Feedforward Neural Network

Hidden Layers:

- Any layer **between** the input neurons and the output layer is a **hidden layer**.
- During the training process, the training data does not show desired outputs of hidden layers.
- A network can consist of **any number** of hidden layers.
- Each hidden layer can contain **any number** of artificial neuron units/nodes.

Slide 13: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks (FFNN): Overview



Feedforward Neural Network

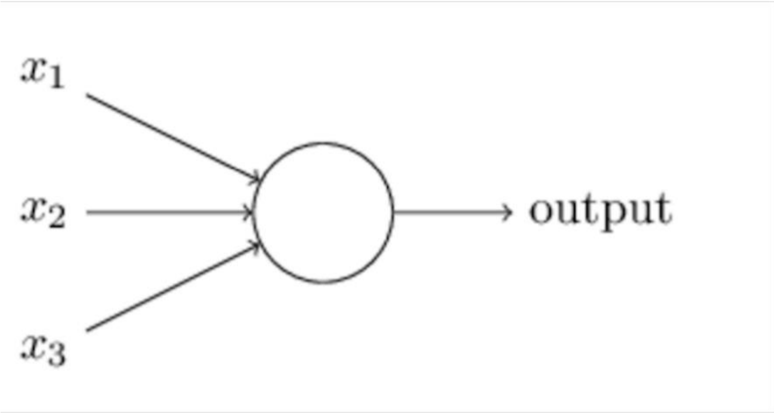
- The figure to the left shows an example of a feedforward neural network of two layers:
  - 8 input neurons
  - One **hidden** layer of 8 neurons
  - One **output** layer of 2 neurons

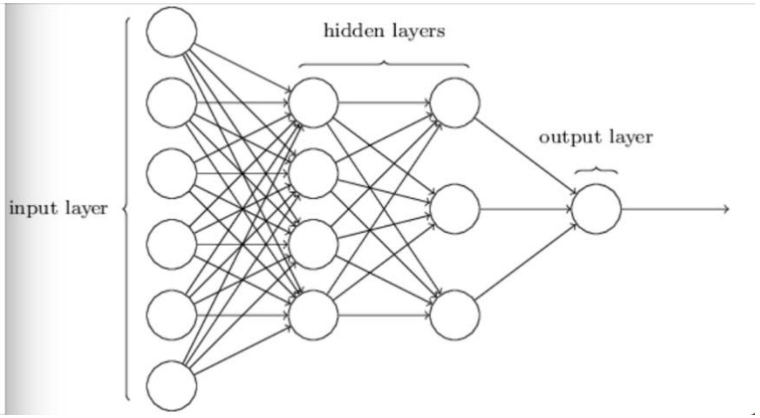
Slide 14: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network (FFNN) = Multi-Layer Perceptron (MLP)

In a feedforward neural network:

- Data enters at the input neurons and passes through the layers of the network until it arrives at the output layer, i.e., only one direction, the **forward**.
- There are **no feedback connections** in which **outputs** of the network or **outputs** of each layer are **fed back** into itself.





McCulloch-Pitts neuron model and Feedforward Neural Network (Sources: towardsdatascience.com)

Slide 15: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks:

Feedforward neural networks are the quintessential deep learning models.

- The **goal** of a feedforward network is to **approximate** some function $f^*(x)$.

- For example:
  - For a classifier, $y = f^*(\mathbf{x})$ maps an input **x** to a category **y**.
  - A feedforward network defines **a mapping** $\mathbf{y = f(x;\theta)}$ and **learns** the value of the parameter $\boldsymbol{\theta}$ that result in the best function approximation (Reference).

- These **networks** are represented by a **composition** of many different functions.
  - For example:
    - We might have three functions $f\_1(x)$, $f\_2(y)$ where $y = f\_1(x)$, and $f\_3(z)$ where $z = f\_2(y)$.
    - These functions are **connected** in a **chain**, to form $\mathbf{f(x)} = \mathbf{f\_3(f\_2(f\_1}(x)))$.
    - In this, **f_1(x)** is the first layer, **f_2(y)** is the second layer and **f_3(z)** is the output layer.

Slide 16: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks: Gradient-Based Learning

- **Gradient Descent** is an optimization algorithm .
  - It is based on a convex function.
  - It tweaks its parameters **iteratively** to **minimize** a given function to its **local minimum**.
  - It is used in **training** a machine learning or deep learning model.

- **Gradient descent** is a method to **find** the values of a cost function's parameters (coefficients) that **minimize** the function as far as possible.

- What is a **Gradient**?
  - "A gradient measures **how much** the **output** of a **function changes** if you change the **inputs** a little bit.“ - Lex Fridman (MIT)
  - For a machine learning or deep learning model, a gradient measures the **change** in all weights with regard to the **change** in cost or error.

Slide 17: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks: Gradient-Based Learning

For the learning process of a machine learning or deep learning model:

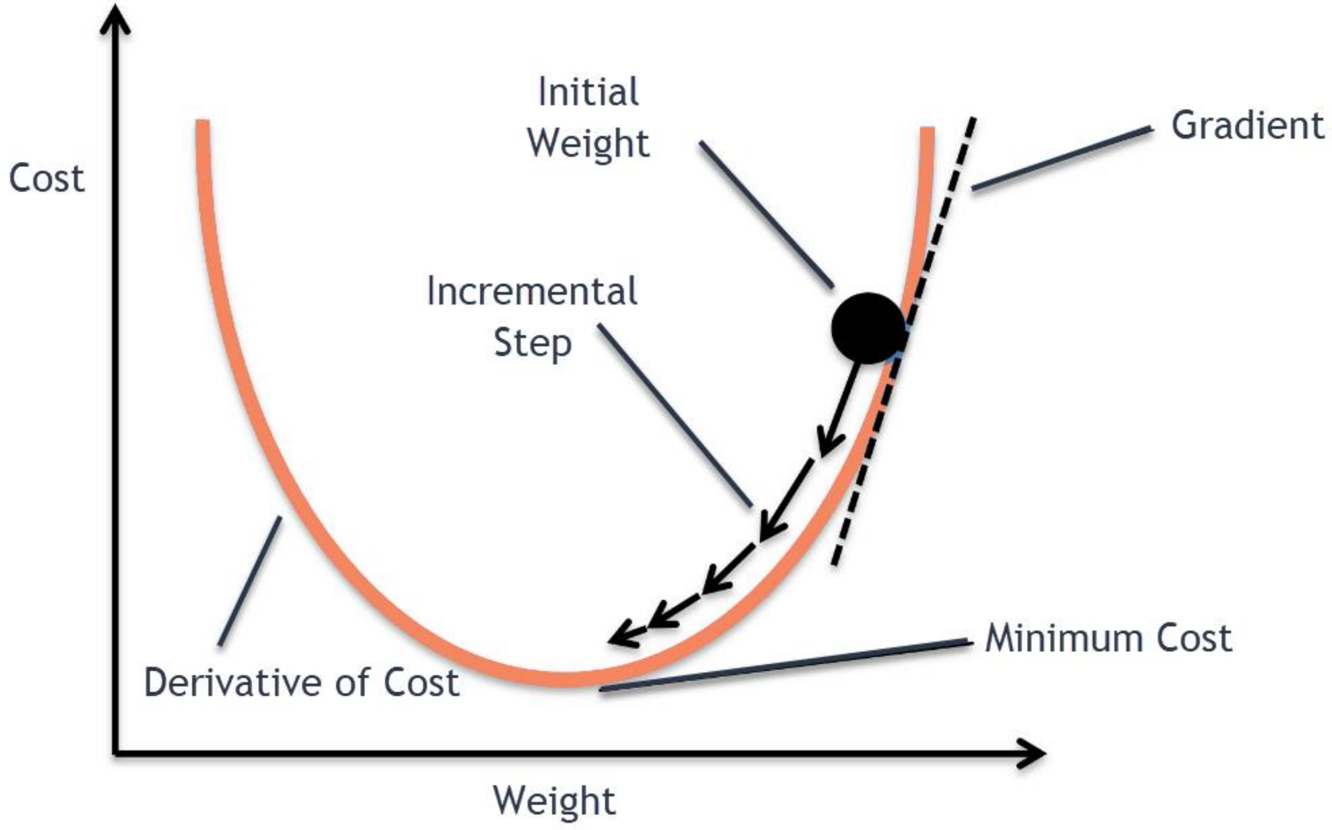
- A gradient measures the **change** in all weights with regard to the **change** in cost or error.

- **Gradient** can be illustrated as the **slope** of the **cost** function:
    - The higher the gradient: The steeper the slope → The **faster** a model can **learn**.
    - The lower the gradient: The less steep the slope → The **slower** a model can **learn**.
    - If the gradient is **near zero** or **zero**, the model **stops** learning.
        - The cost function gets to its minimum.

        OR
        - This is a serious well-recorded problem in training deep learning models.
        - The problem is called **Vanishing Gradient**.

Slide 18: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network: How Does It Learn?



Gradient-Based Learning (Source: Divakar Kapil, medium.com)

Slide 19: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks: Gradient-Based Learning

The learning process in feedforward neural networks is **gradient-based**.

However, there are **differences** between training **artificial neural networks** and training a general machine-learning model with gradient descent.

For artificial neural networks like feedforward neural networks (FFNN):

- The **nonlinearity** of a neural network causes most **loss functions** to become **non-convex**.
- Neural networks are usually trained by **using iterative, gradient-based optimizers** that **merely** drive the cost function to a **very low** value.
  - i.e., **only** as low as possible, **not** the minimum value as in the case of convex function.

Slide 20: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks: Optimization & Cost Function

- **Cost function** shows the difference between the approximation made by the model and the actual target value.

- A cost function is mostly of form **C(W, B, Sr, Er):**
  - **W** is the weights of the neural network, **B** is the biases of the network, **Sr** is the input of a single training sample, and **Er** is the desired output of that training sample.

- Many cost functions can be used for the training process of artificial neural networks:
  - **Mean Squared Error** (MSE: a.k.a. Quadratic cost function OR Sum of Squared Errors)
  - **Cross-entropy cost**
    - Also known as Bernoulli negative log-likelihood and Binary Cross-Entropy
  - Exponential Cost
  - Hellinger distance

- Among the above mentioned cost functions:
  - **MSE** is very popular for regression problems.
  - **Cross-entropy** is the most often used for classification tasks.

Slide 21: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks: Optimization & Cost Function

- An **Optimizer** or optimization algorithm is used to minimize the **cost function**:
    - It **updates** the values of the **weights** and **biases** after **every training** cycle or epoch **until** the cost function reaches the global optimum.

- Optimization algorithms (optimizers) are of **two types**:
    - First Order Optimization Algorithms
    - Second Order Optimization Algorithms

- **First Order** Optimization Algorithms (optimizers):
    - These algorithms **minimize** or **maximize** a **cost function** using its gradient values with respect to the parameters.
    - The **First Order derivative** tells us whether the function is **decreasing** or **increasing** at a particular point, in short, it gives the line which is tangent to the surface.

Slide 22: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks: Optimization & Cost Function

- **Second Order** Optimization Algorithms (optimizers):
  - Using second order derivatives to minimize the cost function and are also called Hessian.
  - Since the second derivative is **costly** to compute, the second order is **not used much**.
  - The second order derivative tells us whether the first derivative is increasing or decreasing which hints at the function's curvature.

- There are many algorithms (optimizers) that can be used for optimization:
  - Stochastic gradient descent
  - Adagrad
  - **Adam**
  - RMSProp.

- Among the above-mentioned optimizers, **Adam** is now the **most popular** optimization algorithm in training artificial neural networks.

Slide 23: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Networks: Optimization & Cost Function

Optimization Algorithms (optimizers): **Adam**

- Is an extension to **stochastic gradient descent (SGD)** algorithm.
- Can be used to **update** network weights iterative based on training data.
- Adam was proposed by Diederik Kingma (OpenAI) and Jimmy Ba (University of Toronto)
- The name is derived from "Adaptive Moment Estimation," not an acronym.

- Providing **benefits** on non-convex **optimization** problems
  - Straightforward to implement
  - Computationally efficient
  - Little memory requirements
  - … MORE

- Combining the **advantages** of two other popular optimization algorithms:
  - Adaptive Gradient Algorithm (AdaGrad)
  - Root Mean Square Propagation (RMSProp)

Slide 24: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network: How Does It Learn?

Scenario:

- It is assumed that the **desired output** of the network is **y**.
- Also assumed that the neural network produces an output **y'**.

- The **difference** between the predicted output and the desired output **(y’ - y)** is considered as the **cost or loss** represented by the **cost/loss function**.
  - The loss is high when the neural network makes a lot of mistakes
  - The loss is low when it makes fewer mistakes.

The **goal** of the training process:

- To find the **weights** and **bias** that **minimizes** the loss function over the training set.

Slide 25: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network: How Does It Learn?

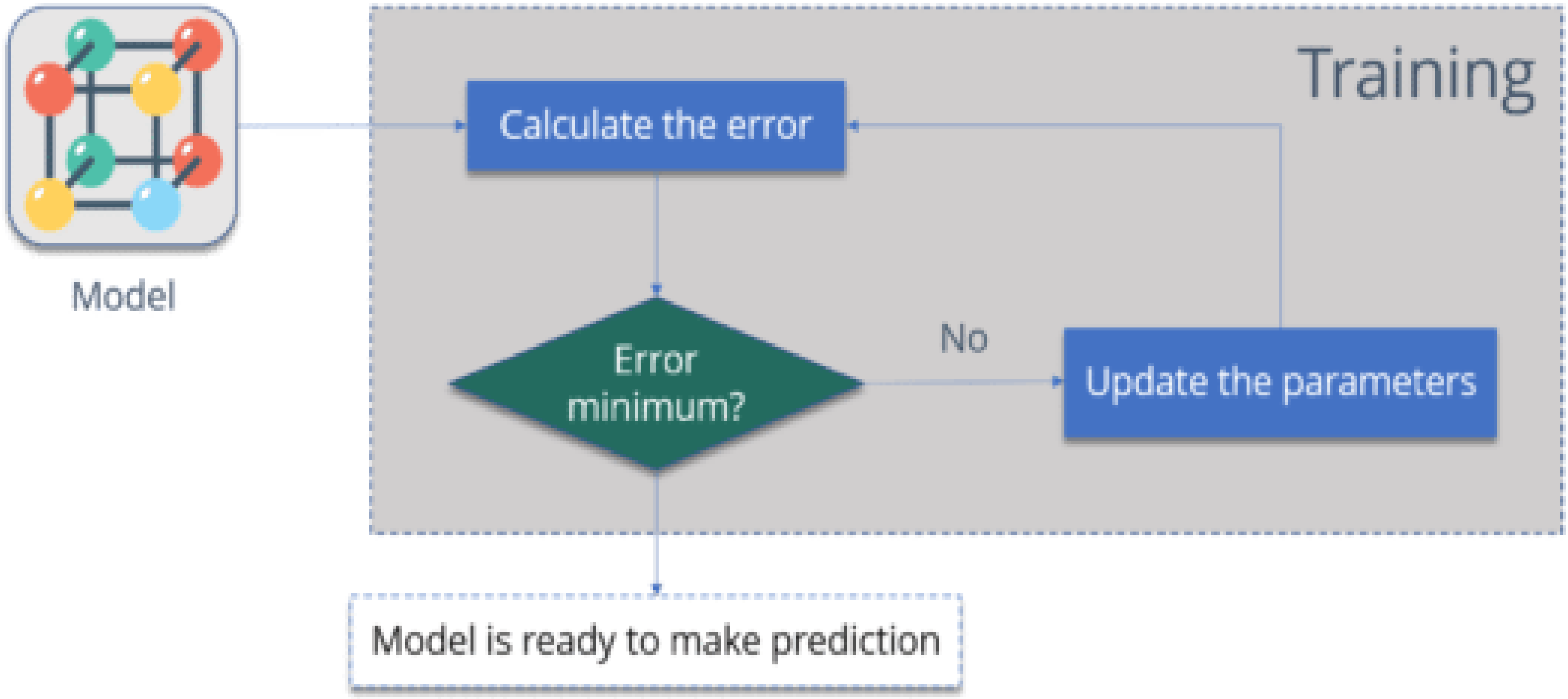
Backpropagation Algorithm

- The training is done using the Backpropagation algorithm, also called **backprop**.
    - The algorithm **iteratively** passes batches of data through the network and **updating the weights** to **decrease the error**, or cost/loss.
    - The algorithm can do this by running an **optimization algorithm** for deep learning like **Adam** or Stochastic Gradient Descent (SGD).

- The **amount** by which the weights are changed is determined by a parameter called **Learning Rate**.

Slide 26: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network: How Does It Learn?



Training A Neural Network – Backpropagation (Sources: edureka.co)

Slide 27: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network: How Does It Learn?

Backpropagation Algorithm

- The **backpropagation** algorithm was introduced in the **1970s**.
  - However, its importance was not fully recognized until a famous 1986 paper by David Rumelhart, Geoffrey Hinton, and Ronald Williams.
  - That paper describes several neural networks where backpropagation works far faster than earlier approaches to learning.

- Today, the **backpropagation** algorithm is **very popular** as a technique to train neural networks.

Slide 28: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network: How Does It Learn?

Backpropagation Algorithm

- Backpropagation is the essence of **training** neural networks.
  - It is considered the most effective technique to train neural networks.
  - It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (or iteration).
  - Proper tuning of the weights allows reducing error rates and making the model reliable by increasing its generalization.

- Backpropagation is a short form for "backward propagation of errors."
  - It is a **standard method** of **training** artificial neural networks.
  - This method helps to calculate the gradient of a loss function with respects to all the weights in the network.

Slide 29: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network: How Does It Learn?

Backpropagation Algorithm

- Input data is split into training batches.
- First, **initializing** some random value to the weights and propagating **forward**.

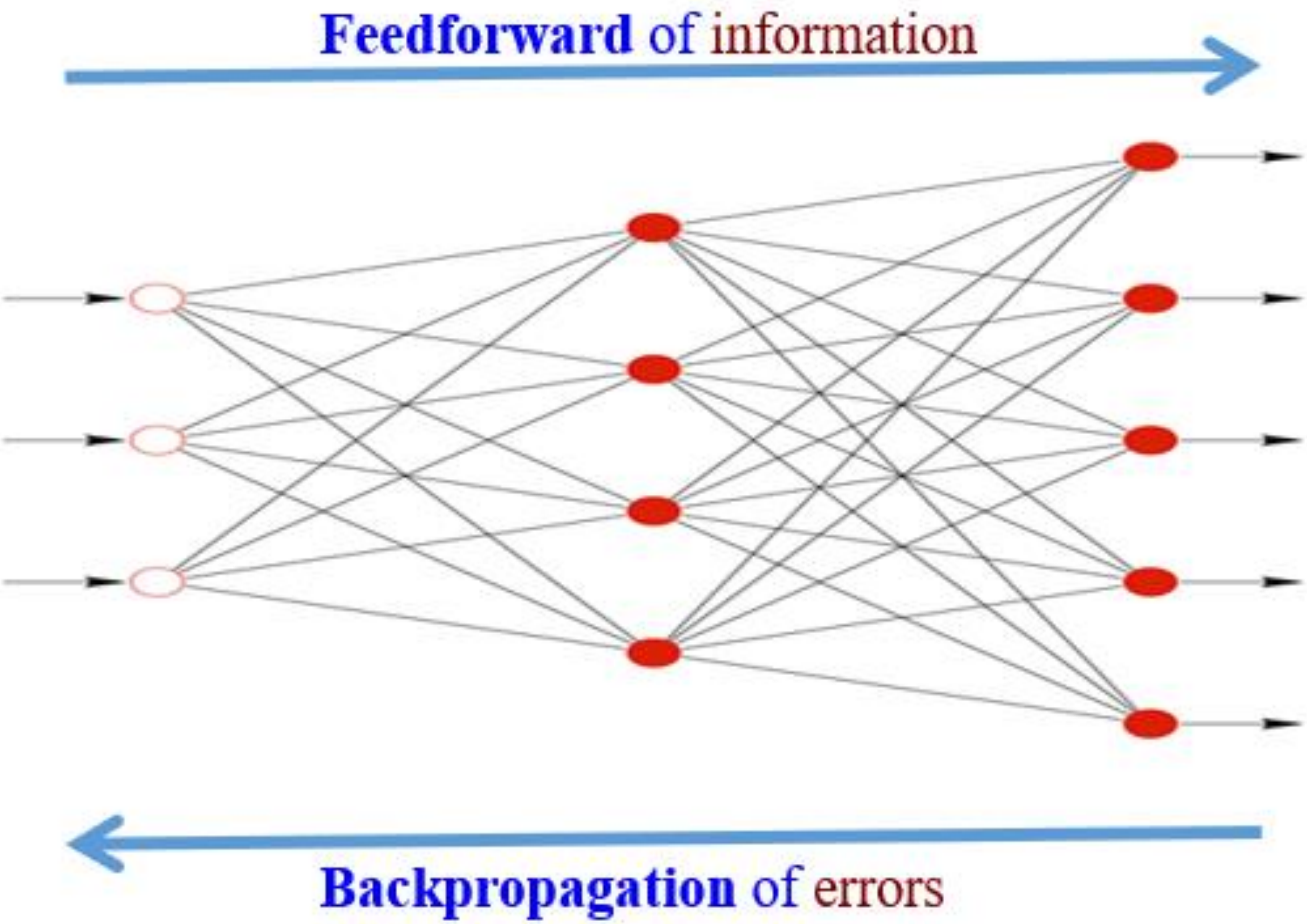
- **Forward pass**:
  - The batches are passed from input neurons through the layers of the network to the output layer to produce the outputs.

- **Output comparison**:
  - The outputs are compared with the actual values of the outcomes (or labels) of the data set.
  - The difference, i.e., error, is calculated. To reduce error, recursively **propagating backward**.

- **Backward pass**:
  - The difference, i.e., error, cost, or lost, is used to change the weights of the neurons at each layer recursively such that the error decreases gradually.

Slide 30: AI Deep Learning: Feedforward Neural Networks (FFNN)

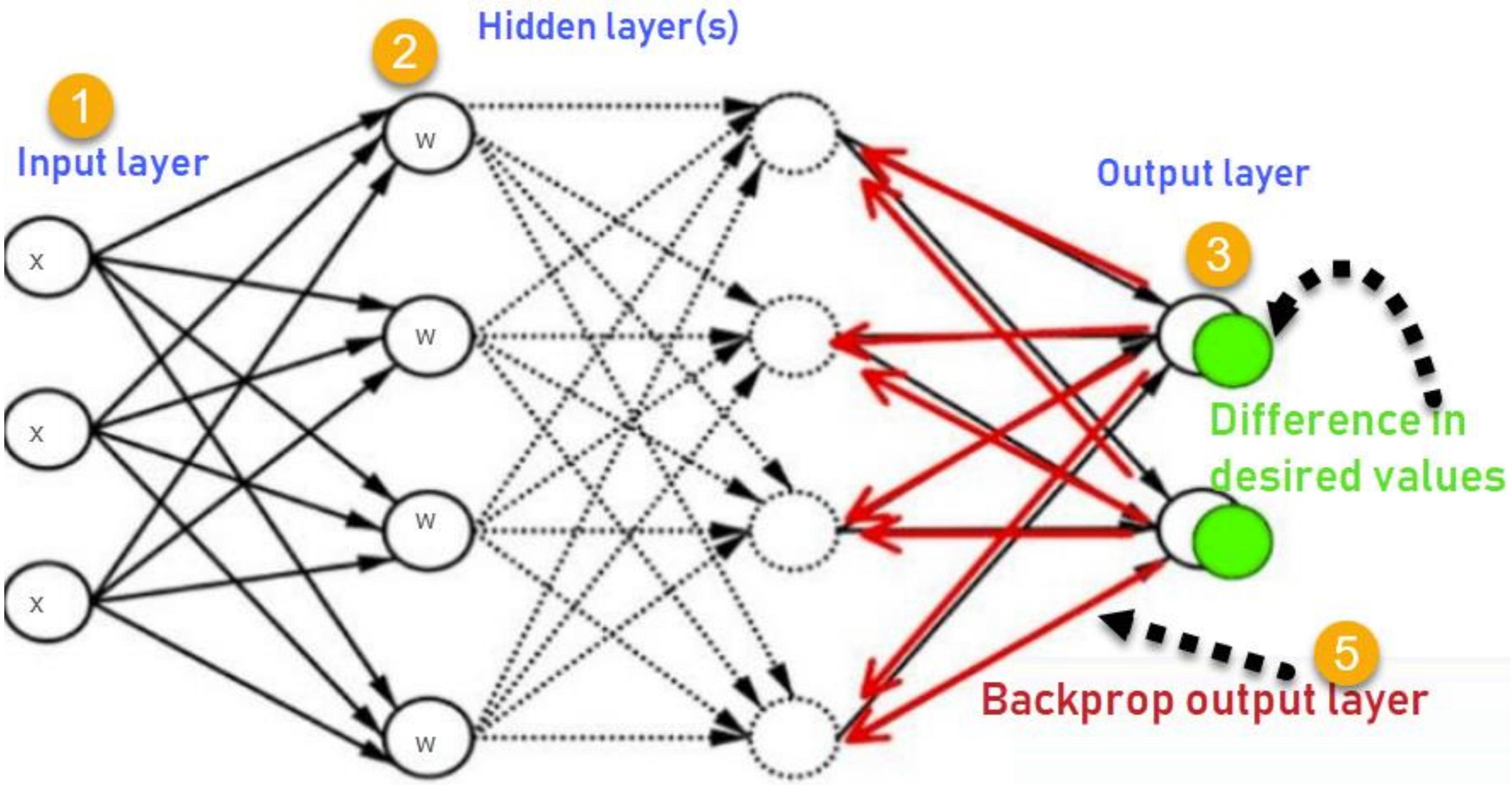
Feedforward Neural Network: How Does It Learn?



Feedforward Neural Network and Backpropagation Algorithm

Slide 31: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network: How Does It Learn?



Feedforward Neural Network and Backpropagation Algorithm (Sources: guru99.com)

Slide 32: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network: How Does It Learn?

Backpropagation Algorithm

- The **core** of **backpropagation** algorithm is an expression for the partial derivative **∂C/∂w** :
  - Is the partial derivative of the **cost function C** with respect to (w.r.t.) any weight **w** (or bias b) in the network.
  - The expression tells us **how quickly** the **cost changes** when we **change** the weights and biases.

- The partial derivative **∂C/∂w** of the cost function C w.r.t. any weight w (or bias b)
  - Is similar to the case **of y = f(t)**
    - Where **y** is the **distance** that one object is traveling during the time period of t.
    - The **change of distance** per time unit (**speed**): **y’ = f’(t)**.
    - The **change of the speed** per time unit (acceleration): **y” = f”(t)**.

Slide 33: AI Deep Learning: Feedforward Neural Networks (FFNN)

Feedforward Neural Network: How Does It Learn?

Backpropagation Algorithm

- Most prominent **advantages** of Backpropagation are:
  - Backpropagation is fast, simple and easy to program
  - It has no parameters to tune apart from the numbers of input
  - It is a flexible method as it does not require prior knowledge about the network
  - It is a standard method that generally works well
  - It does not need any special mention of the features of the function to be learned.