MLAI 504
NEURAL NETWORKS & DEEP LEARNING

Dr. Zein Al Abidin IBRAHIM

zein.ibrahim@ul.edu.lb

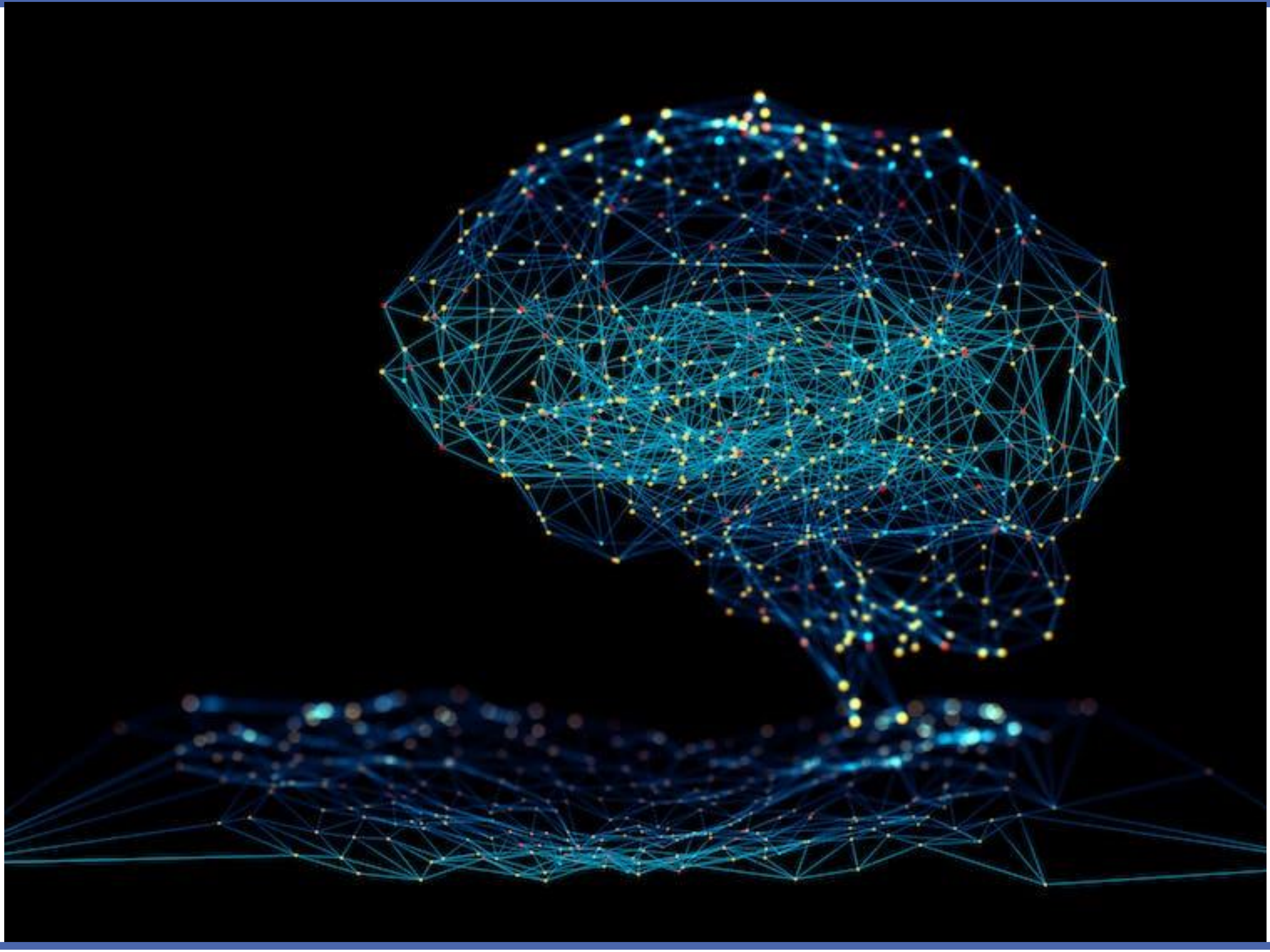
MLAI – 2023/2024

Introduction to Neural Networks

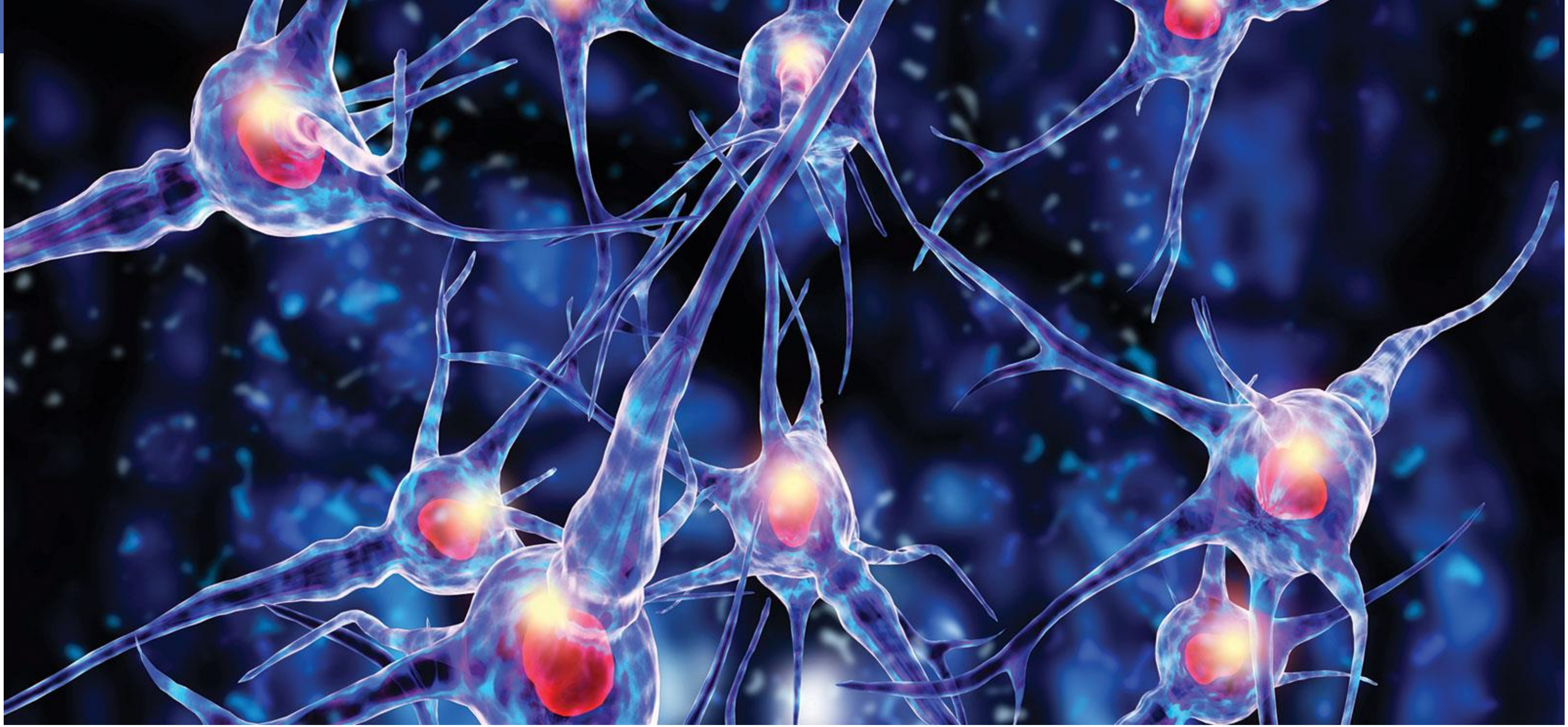
Neural Networks and Deep Learning

- What is Neural Network?
- Properties of Neural Networks
- Looking inside the human brain
- How to model a neuron in ANN?
- Directed graph to represent a NN?
- What is Feedback and where to use it?
- How is knowledge represented ?
- Learning Process and task

Outline



Brain



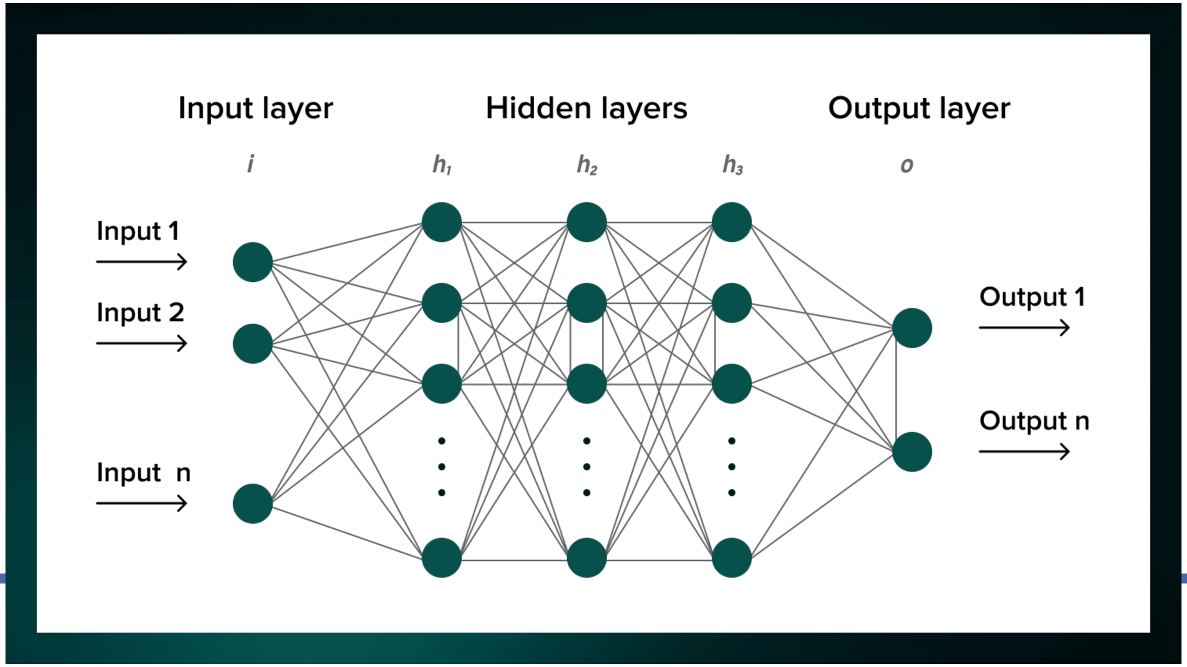
Brain

Brain

- Idea → inspired from human Brain
- A brain is composed of a vary large number of processing units → called ***neurons***.
- Number approximately = 10^{11} = 10 milliard neurons
- Neurons works in parallel ⇒
  - Computational power of brain is very high
  - In addition → large connectivity of neurons
- Connections called ***synapses***
- Neurons have connections to around 10^4 other neurons.
- Brain take approximately 100-200ms to perform perceptual recognition

What is Neural Networks ?

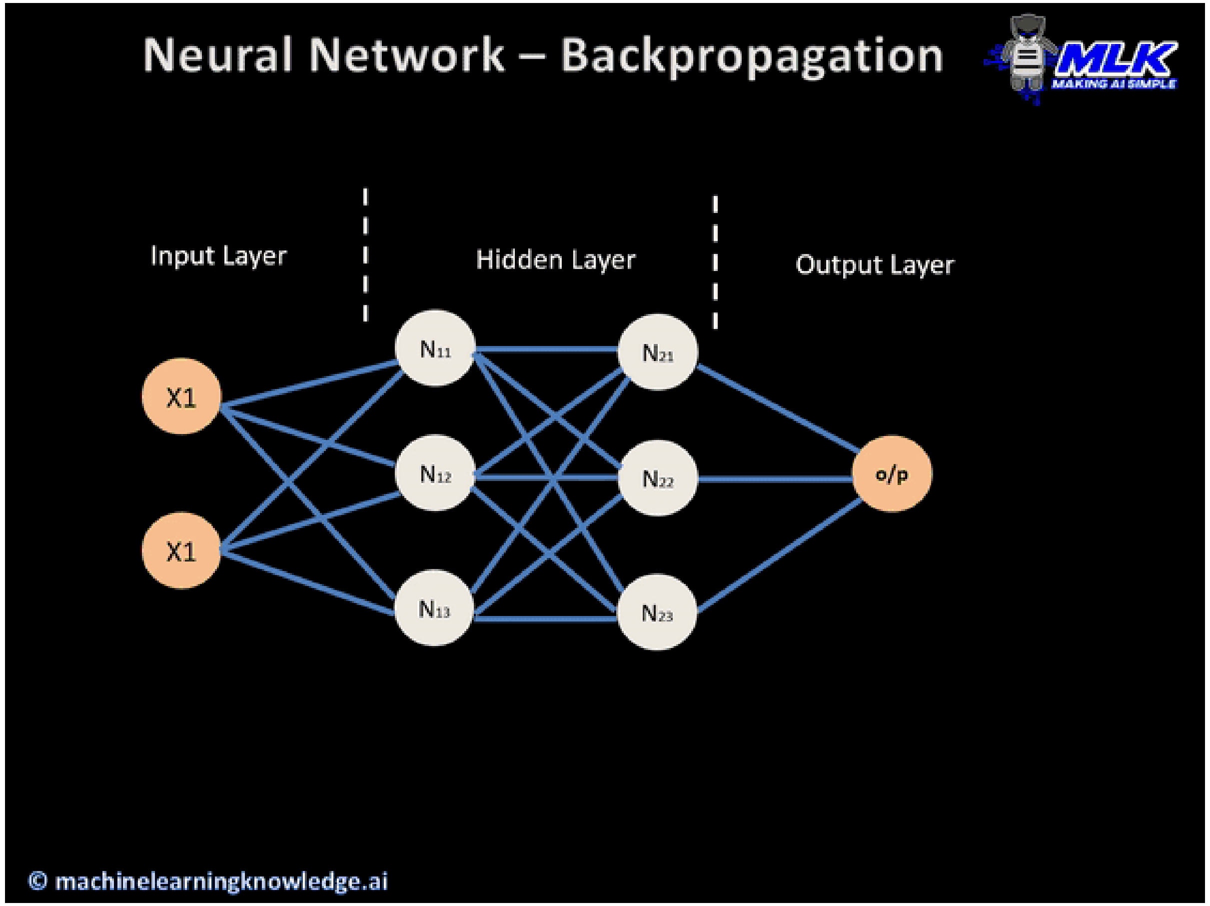
- NN → massive number of parallel distributed processing units
- Can store experiential knowledge (experiences)
- Similar to brain in two main things:
  1. Knowledge → acquired from its environment through a learning phase.
  2. Interneuron connection weights used to store the acquired knowledge.

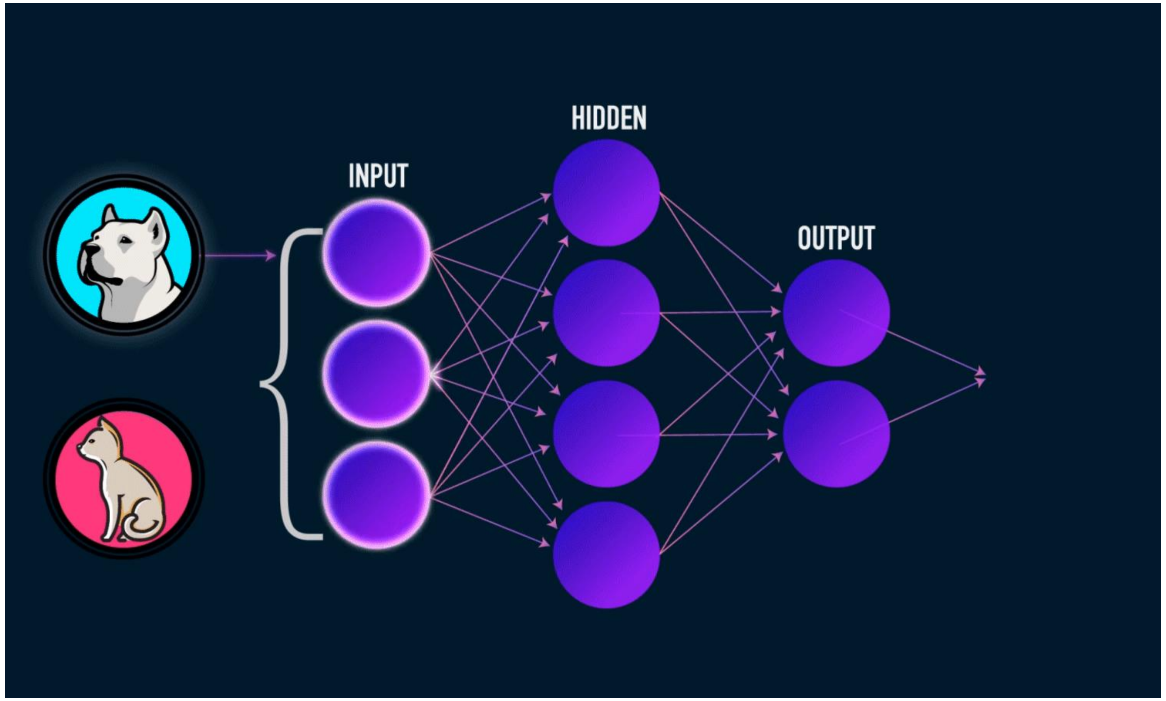


What is ANN?

- **Several properties and capabilities of Neural Networks:**
  - **Generalization** → production of reasonable outputs for inputs not encountered during training (learning).
  - **Nonlinearity** → Can solve nonlinear problems.
  - **Input-Output Mapping** → In supervised Learning, a mapping function that maps the input to the output is extracted.
    - Training example are fed to the network that updates the synaptic weights.
    - Updates done to minimize the difference between predicted and real label.
    - Finish when updating reaches stable state.

Why Neural Network?





Why Neural Network?

Prediction phase

Training phase

- **Several properties and capabilities of Neural Networks:**
  - **Adaptivity** → Weights in NN can be updated to take into accounts changes in the environment.
    - Not every change in the environment needs model updates. It should be a change over long period of time.
  - **Evidential Response** → NN can give decision confidence.
  - **Contextual Information** → Neurons are not independent. They are affected by the global activity of all other neurons in the network.

Why Neural Network?

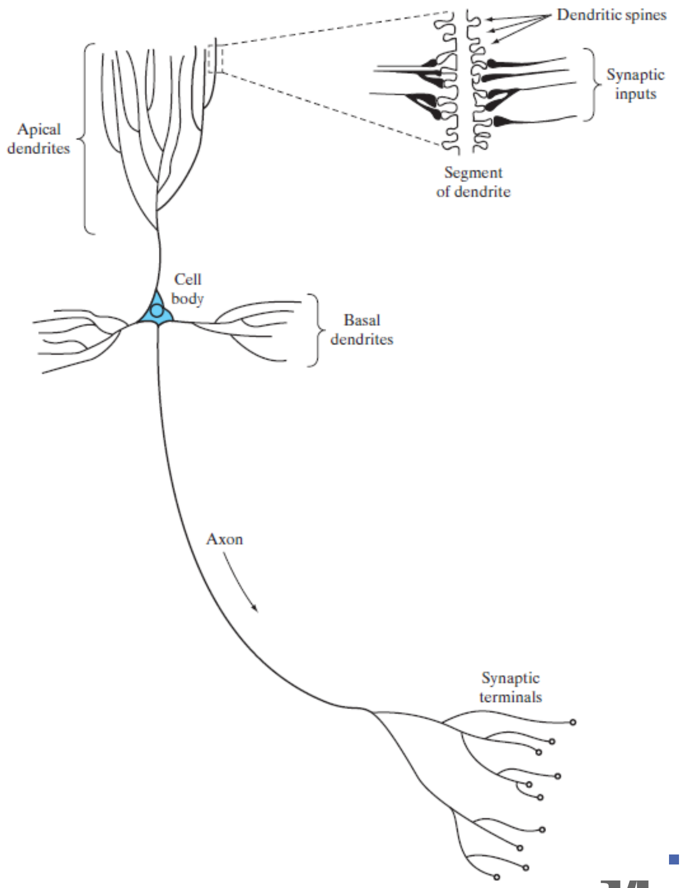
- **Several properties and capabilities of Neural Networks:**
  - **Fault-Tolerance** → ability of a NN to continue functioning even when some of its components or connections fail.
  - **VLSI Implementability** (Very-Large Scale Integration) → process of creating integrated circuits by combining thousands or even millions of transistors on a single chip.
  - **Uniformity of analysis and design** → use of consistent mathematical and computational methods throughout the process of designing, training, and testing a neural network.
  - **Neurobiological Analogy** → ANN tries to imitate the human brain.

Why Neural Network?

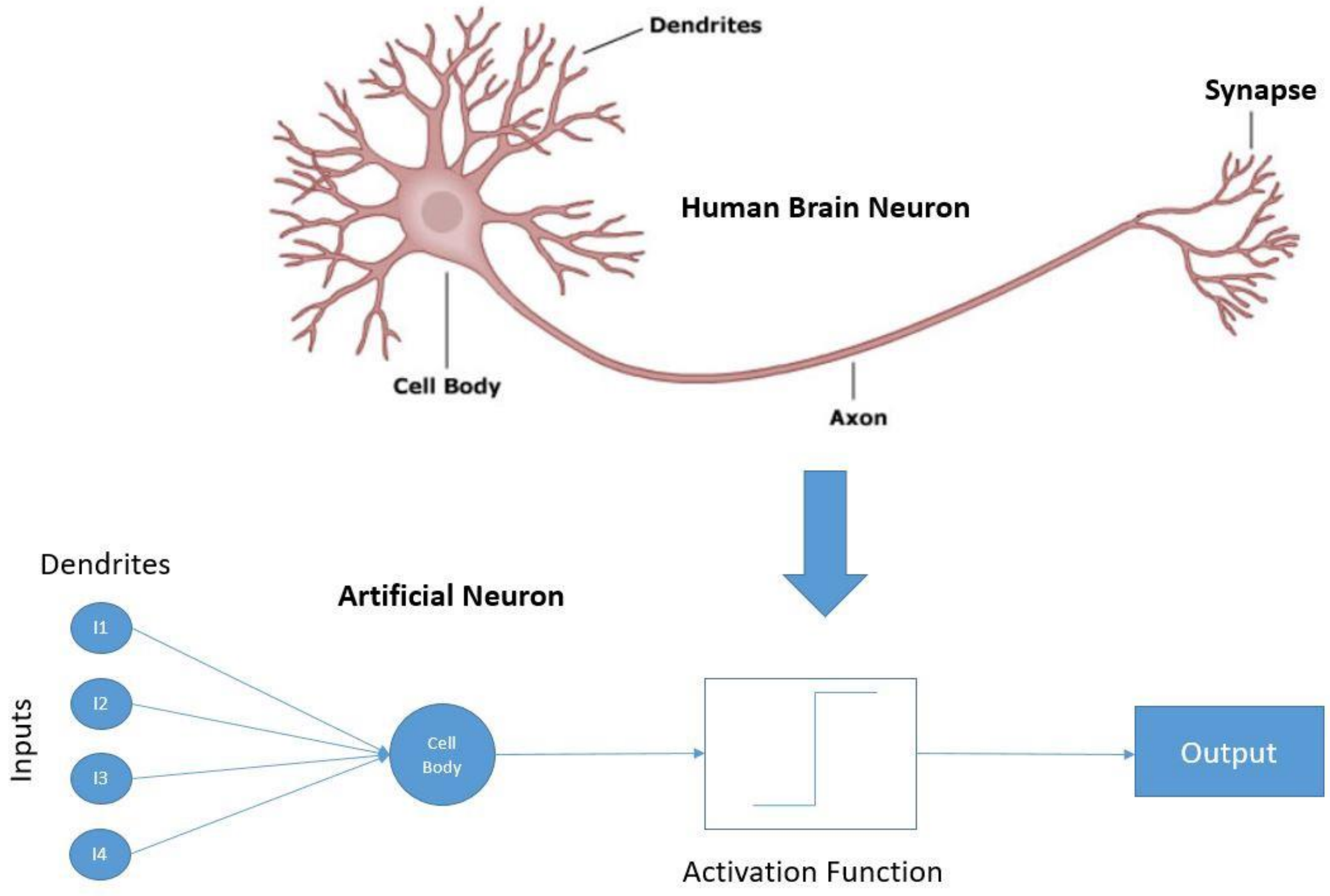
- The human nervous system may be viewed as a three-stage system.
  - Central to the system is the brain (neural net) which continuously receives information, perceives it and make appropriate decisions.
  - Forward arrows (left to right) are information holding signals .
  - Feedback arrows (right to left).
  - Receptors convert stimuli from the human body into electrical pulses that transfer information to the brain.
  - Effectors convert the electrical impulses generated by the neural network into visible response as system outputs.

Looking Inside Human Brain

- **Synapses or nerve endings** are elementary structural and functional units that mediate the interaction between neurons
- **Plasticity** permits the developing nervous system to adapt to its surrounding environment.
  - In an adult brain plasticity can be accounted for by two mechanisms:
    - The creation new synaptic connections between neurons.
    - The modification of existing synapses.
  - **Axons** are the transmission lines.
  - **Dendrites** are the receptive zones.
    - Information is received through the dendritic spines.



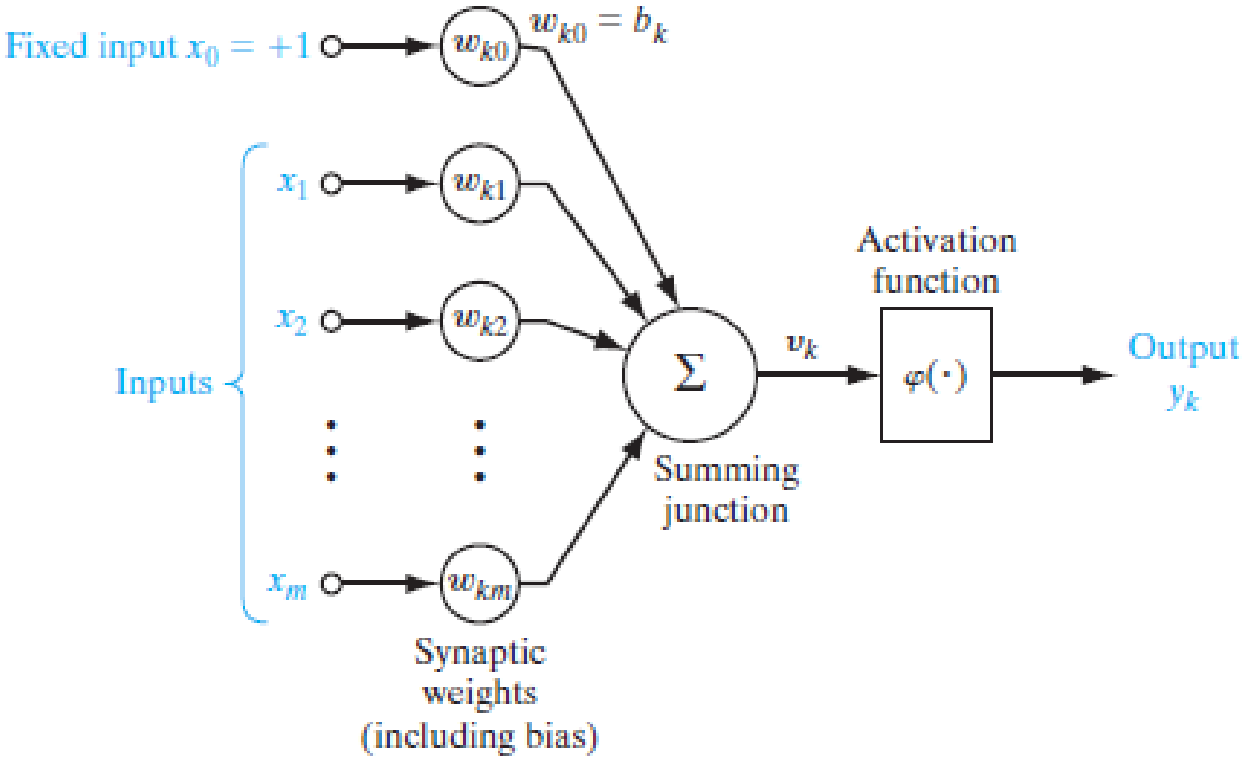
Looking Inside Human Brain



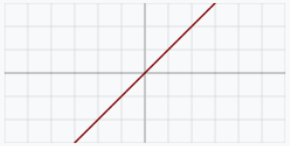
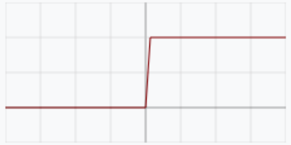
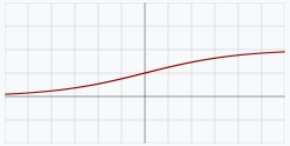
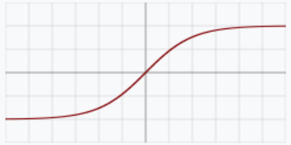
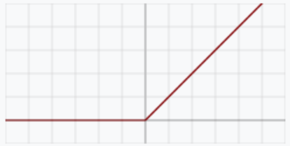
Looking Inside Human Brain

➢ Another representation for the bias is to consider it as an additional neuron, the summation this time Starts from zero.

- $v_k = \sum_{j=0}^{m} w_{kj}\, x_j$
- $y_k = \varphi(v_k)$
- $x_0 = +1, w_{k0} = b_k$

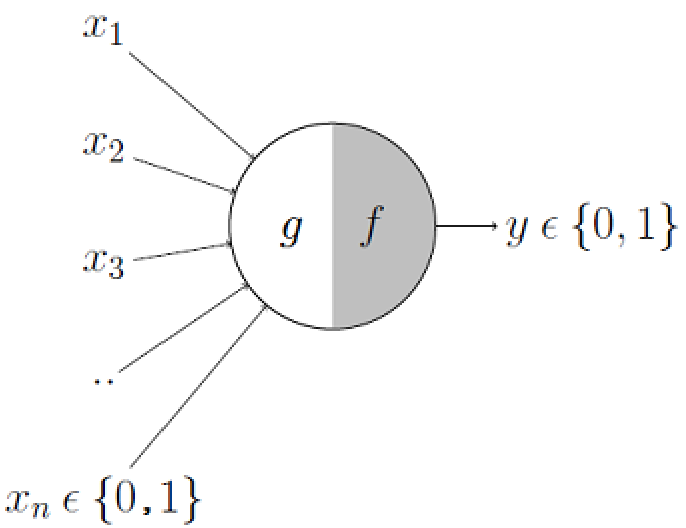


How to Model a Neuron

Name	Plot	Function, $f(x)$	Derivative of f, $f'(x)$	Range
Identity		x	1	$(-\infty, \infty)$
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0 & \text{if } x \neq 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$\{0, 1\}$
Logistic, sigmoid, or soft step		$\sigma(x) = \dfrac{1}{1 + e^{-x}}$ [1]	$f(x)(1 - f(x))$	$(0, 1)$
tanh		$\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f(x)^2$	$(-1, 1)$
Rectified linear unit (ReLU)[11]		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$

Some activation functions

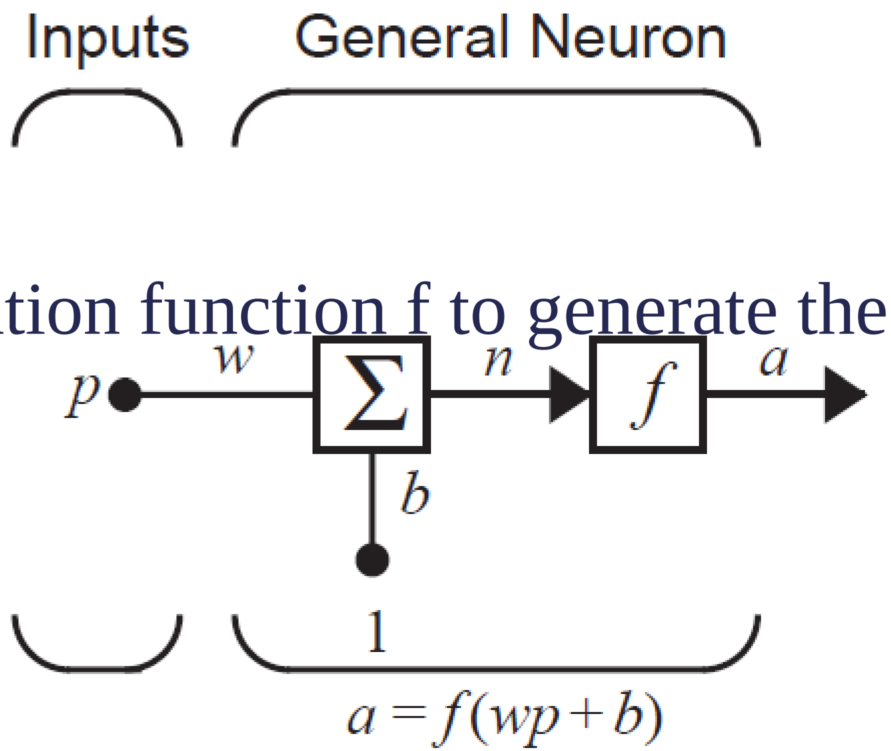
- The first computational model of a neuron was proposed by Warren MuCulloch (neuroscientist) and Walter Pitts (logician) in 1943.
- Inputs → Boolean
- Output → Boolean
- Activation → thresholding
- What we can do with → OR, AND, >
- No learning from data
- Just a theoretical model



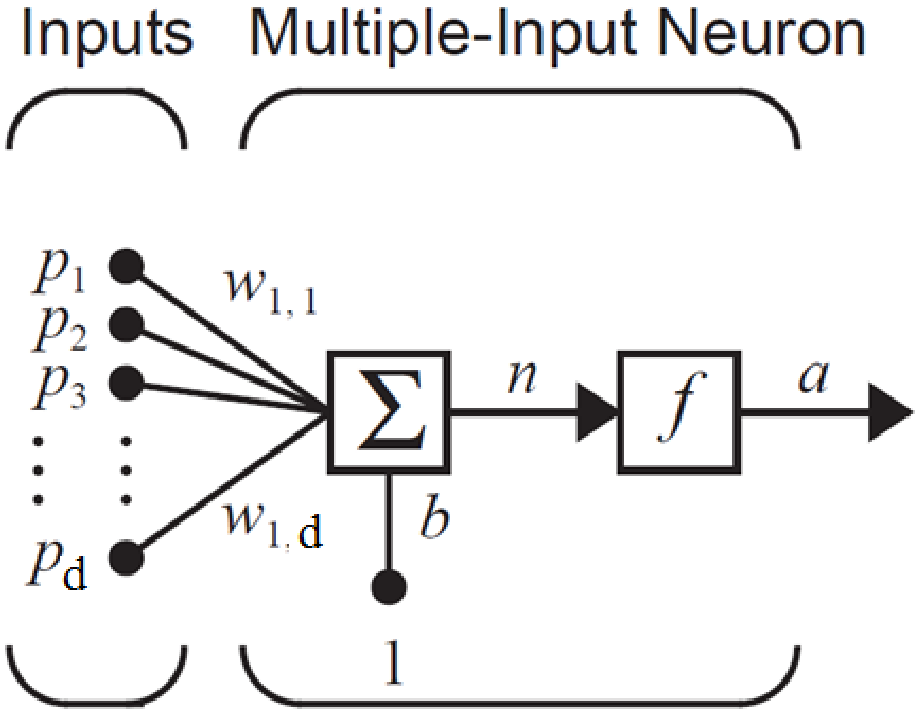
McCulloch-Pitts model

One-input Neuron: Example

- 1D pattern p
- The associated weight = w
- The bias value is b
- The output is n
- The value n passes through the activation function f to generate the output a
- Ex: w=3, p=2, b=-1.5 ⇒
  - a=f(3(2)-1.5)=f(4.5)



Multiple-input Neuron: Example



$$a = f(\mathbf{W}\mathbf{p}+b)$$

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \cdots + w_{1,d}p_d + b$$

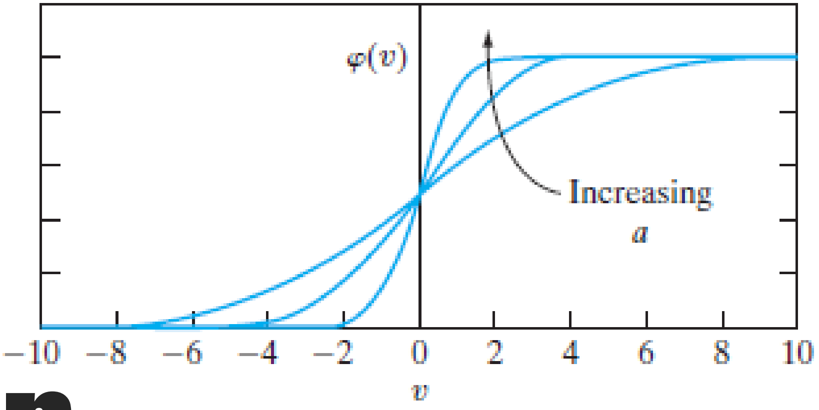
$$n = \mathbf{W}\mathbf{p} + b \qquad a = f(\mathbf{W}\mathbf{p} + b)$$

- Output of the neuron can be written:
  - $y = w^T x$ with $w=[w_0, w_1, \ldots.w_d]$ & $x=[1, x_1, \ldots.,x_d]^T$
  - Weights we need to be learned from training data such that the patterns of the training data are correctly classified
- Ex: when d = 1 ⇒ one-dimension patterns
  - $y = w\,x + w_0$ is the equation of a line
  - The line separate the space in two zones positive-side & negative-side
  - Find weights in such a way →for any new input x, assign it to one of two classes depending in which side is.

The Neuron in the Space

- **Sigmoid Function:** S shaped graph, is by far the most common form of activation used in the construction of a neural network.
  - It is defined by a strictly increasing function that exhibit a graceful balance between linear and nonlinear behavior.
  - An example of the sigmoid function is defined by:
    - $\varphi(k) = \frac{1}{1+\exp(-av)}$
  - Where a is the slope parameter of the sigmoid function.
    - The slop at the origin equals $\frac{a}{4}$.
  - As slope parameter approaches infinity, the sigmoid function becomes the threshold function.
  - Sigmoid function assumes a continuous range of values from $0\ to\ 1$
  - Sigmoid is differentiable

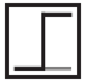
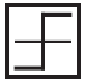
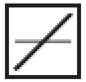
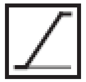
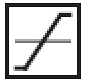
(better for Neural networks) while threshold function is not.



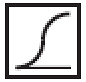
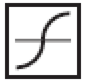
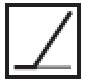
Types of Activation Function

- The activation function can have values between $+1\ and - 1$
  - In this case the activation function is an odd function of the induced local field v.
  - $\varphi(v) = \begin{cases} +1, & v > 0 \\ 0, & v = 0 \\ -1 & v < 0 \end{cases}$
  - This is commonly referred to as the **signum function.**
  - For the corresponding form of a sigmoid function, we my use the hyperbolic tangent function:
    - $\varphi(v) = tang(v)$
      - ✓ This allows the sigmoid function to assume negative values.

Types of Activation Function

Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins

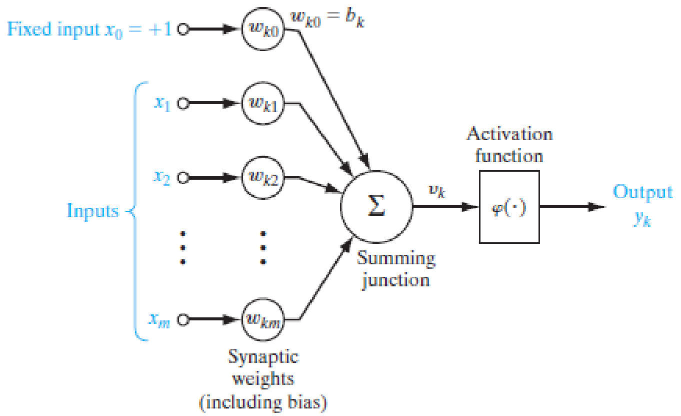
Activation or Transfer functions

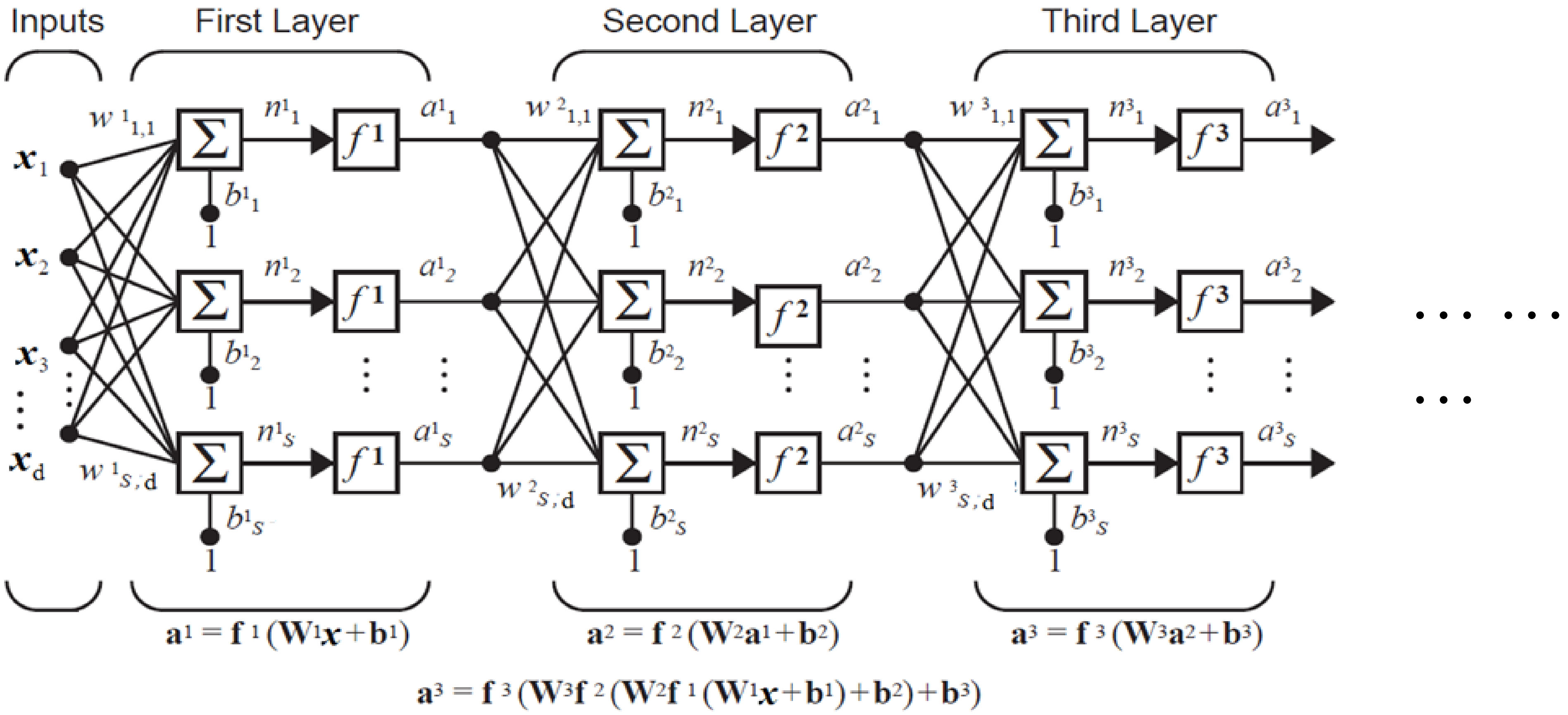
Name	Input/Output Relation	Icon	MATLAB Function
Log-Sigmoid	$a = \dfrac{1}{1+e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \dfrac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n<0$ $a = n \quad 0 \le n$		poslin
Competitive	$a = 1$ neuron with max n $a = 0$ all other neurons	C	compet

Activation or Transfer functions

Stochastic Model of Neuron

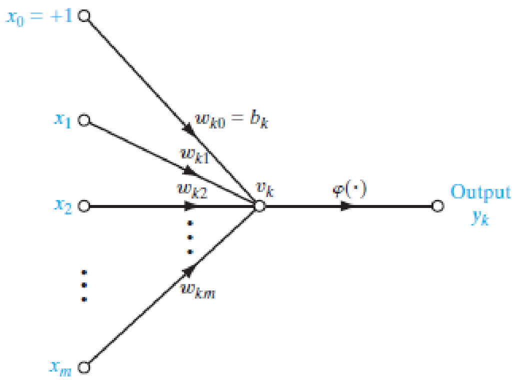
- The neuron is deterministic in the model shown here.
  - That is the input-output behavior is precisely defined for all inputs.
- The McCulloch-Pitts model is given a probabilistic interpretation as follows:
  - The neuron is permitted to reside in only two state $+1\ and -1$ (fires or no).
  - The decision to fire (from off to on) is probabilistic (a threshold is used).
  - Let x denote the sate of the neuron and $P(v)$ denote the probability of firing where v is the induced local field of the neuron:
  - $x = \begin{cases} +1\ \ with\ probability & P(v) \\ -1, with\ probability & 1 - P(v) \end{cases}$
  - Where $P(v)$ is the sigmoid shaped function
  - $P(v) = \frac{1}{1+\exp(\frac{-v}{T})}$
  - Where T is the pseudo temperature used to control the noise level and therefore uncertainty in firing, If $T \rightarrow 0$ the model becomes deterministic.





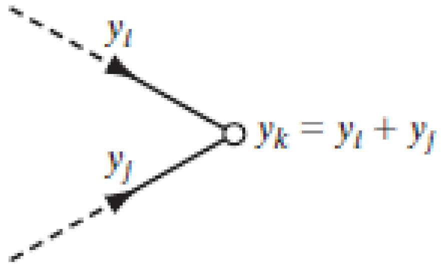
How to represent a Neural Networks: directed Graphs

- **Please note the following rules:**
  - **Rule 1**: A signal flows along a link only in the direction defined by the arrow on the link (Figures (a) and (b)).
  - **Rule 2**: A node signal equals the algebraic sum of all signals entering the node via the incoming links (Figure (c)).
  - **Rule 3:** The signal at a node is transmitted to each outgoing links (Figure (d))
  - **Rule 4:** The output of a neuron is calculated after all the inputs are fed (see figure below).

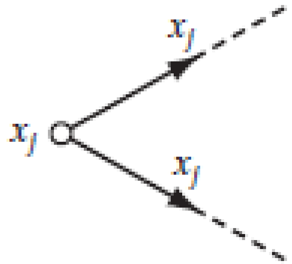


(a)

(b)



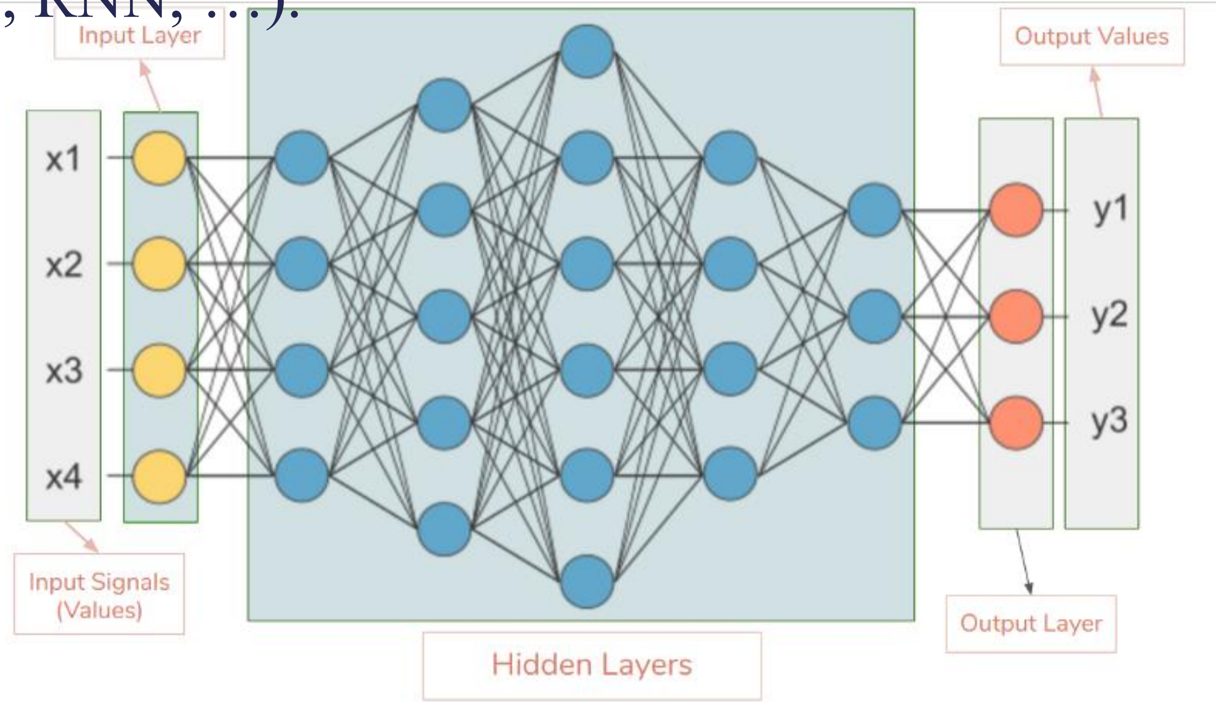
(c)



(d)

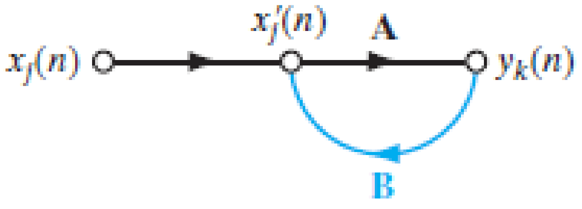
How to represent Neural Networks: directed Graphs

- The directed graph can be complete (known as fully-connected or dense) or incomplete.
  - **Fully-connected** → each neuron in layer n is connected to each neuron in layer n+1 (see figure below).
  - **Incomplete (known as partially complete)**→ a neuron in layer n is connected to some neurons in layer n+1 (used in CNN, RNN, …).
    - referred to as an **architectural graph**



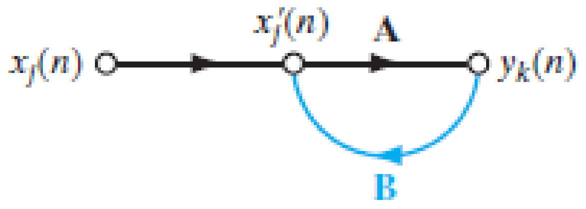
How to represent Neural Networks: directed Graphs

- **Feedback →** output of an element in the system influences in part the input applied to that particular element.
  - Plays a major role in a special class of neural networks called **recurrent networks.**
  - Used during back-propagation algorithm
  - …



What is Feedback and where to use it

- The input-output relationships is given by:
  - $y_k(n) = \boldsymbol{A}[x_j'(n)]$
  - $x_j'(n) = x_j(n) + \boldsymbol{B}[y_k(n)]$
    - Where the Brackets here emphasize that $\boldsymbol{A}$ and $\boldsymbol{B}$ act as operators.
  - Eliminating $x_j'(n)$ we get
  - $y_k(n) = \frac{\boldsymbol{A}}{1-\boldsymbol{AB}}[x_j(n)]$
    - Where $\frac{\boldsymbol{A}}{1-\boldsymbol{AB}}$ is referred to as the closed loop operator of the system, and $\boldsymbol{AB}$ as the open-loop operator.
    - In general, open loop is non-commutative, that is $\boldsymbol{AB} \neq \boldsymbol{BA}$

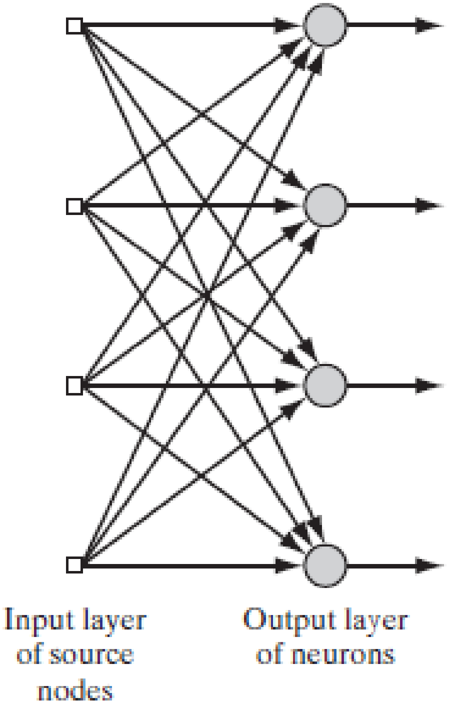


What is Feedback and where to use it

- Here are some network architectures:
  - **Feedforward Neural Networks**
  - **CNN→ Convolutional Neural Networks**
  - **RNN → Recurrent Neural Networks**
  - **LSTM → Long Short-Term Memory Networks**
  - **Autoencoders**
  - **GAN → Generative Adversarial Networks**
  - **Reinforcement Learning Networks**
  - **...**

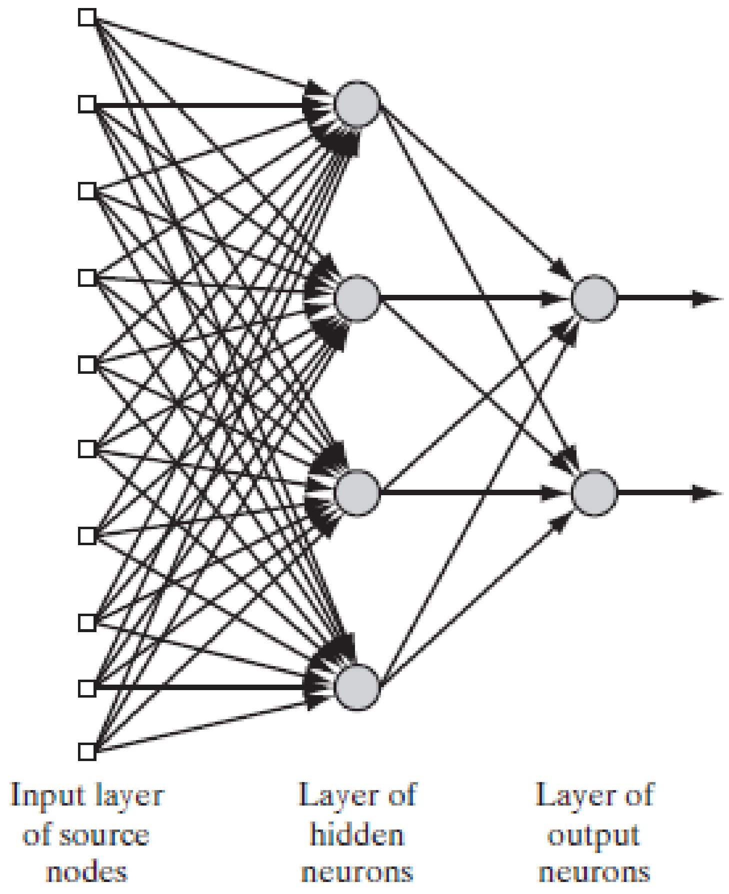
Some Network Architectures

- **(i) Single Layer Feedforward networks.**
  - In a layered neural network, the neurons are organized in the form of layers.
  - In the simplest form we have an input layer of source nodes that projects directly onto output layer of neurons (computational nodes).
    - But not vice-versa. In other words, its strictly of a **feedforward** type.
  - The example here is for a single layer network, where the single layer is referring to the output layer (neurons).
    - We do not count the input layer nodes because no computation is performed there.



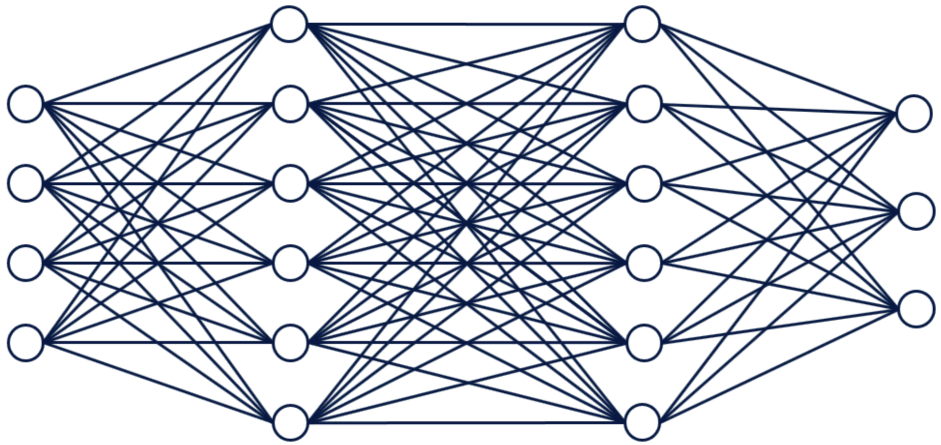
Some Network Architectures

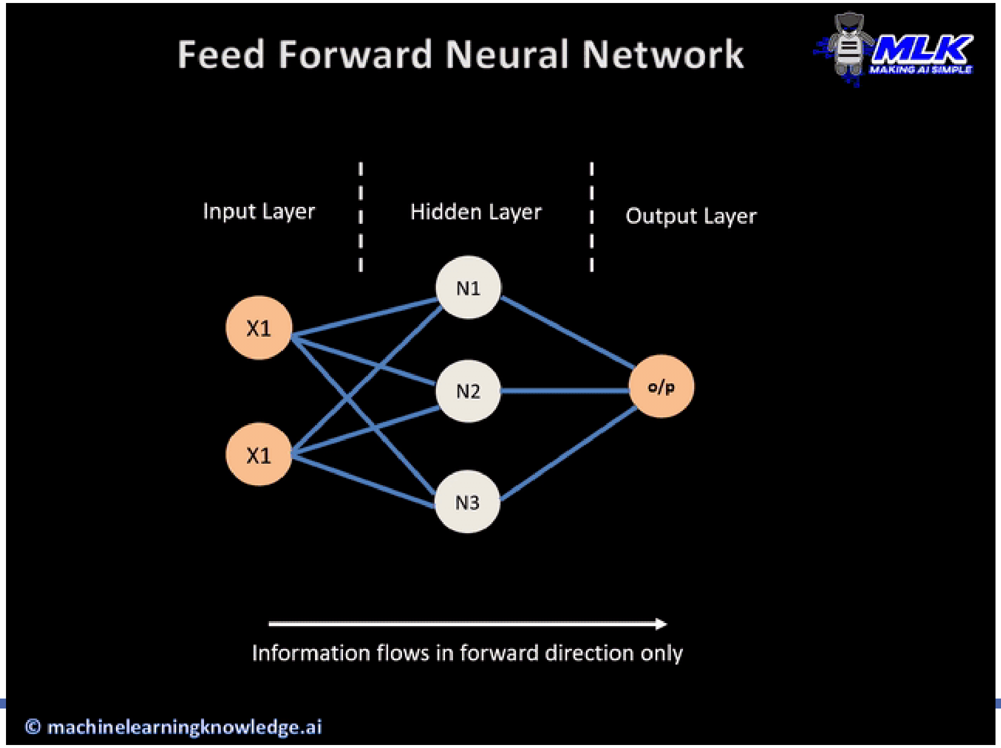
- **(ii) Multilayer Feedforward Networks**
  - It is different by the presence of one or more **hidden layers** whose computation nodes are called hidden neurons.
  - They are called hidden since they are not seen directly from either the input or the output of the network.
  - The function of the hidden neurons is to intervene between the external input and the network output in some useful manner.
  - By adding one or more hidden layers, the network is enabled to extract a **global perspective** despite its local connectivity.
    - This is due to the extra synaptic connection and the **extra dimension** of the neural interaction.
  - The network here is Feedforward network with one hidden layer $10-4-2$
    - 10 source nodes (input layer) 4 hidden neurons and 2 output neuron.
    - The terminology is $\mathrm{m}-h1-h2-q$



Fully connected network

Some Network Architectures

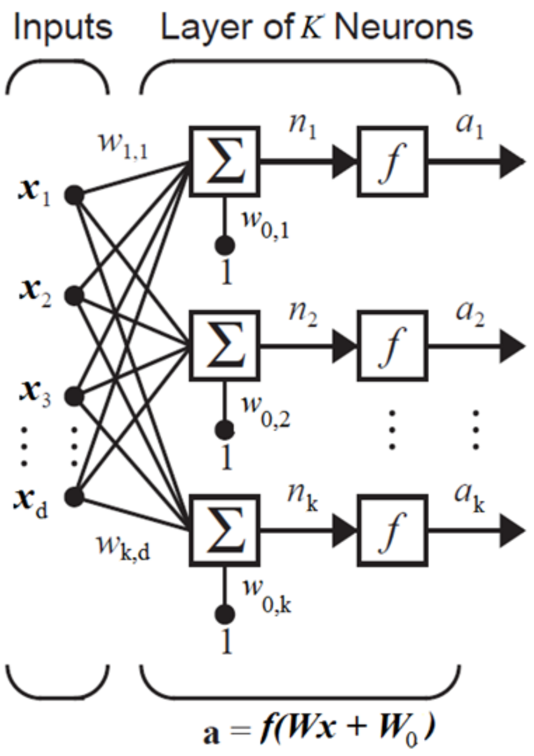


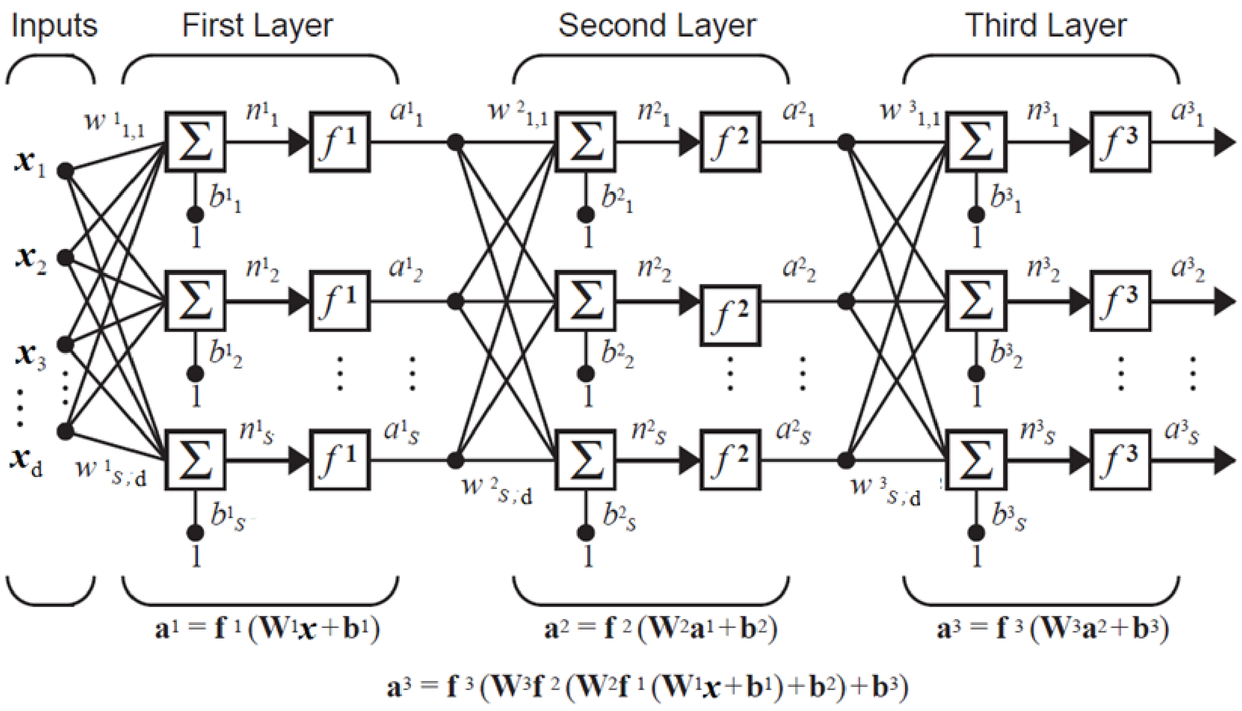


Feedforward Network

$$W = \begin{bmatrix} w_{1,1} & \cdots & w_{1,d} \\ \vdots & \ddots & \vdots \\ w_{k,1} & \cdots & w_{k,d} \end{bmatrix}$$

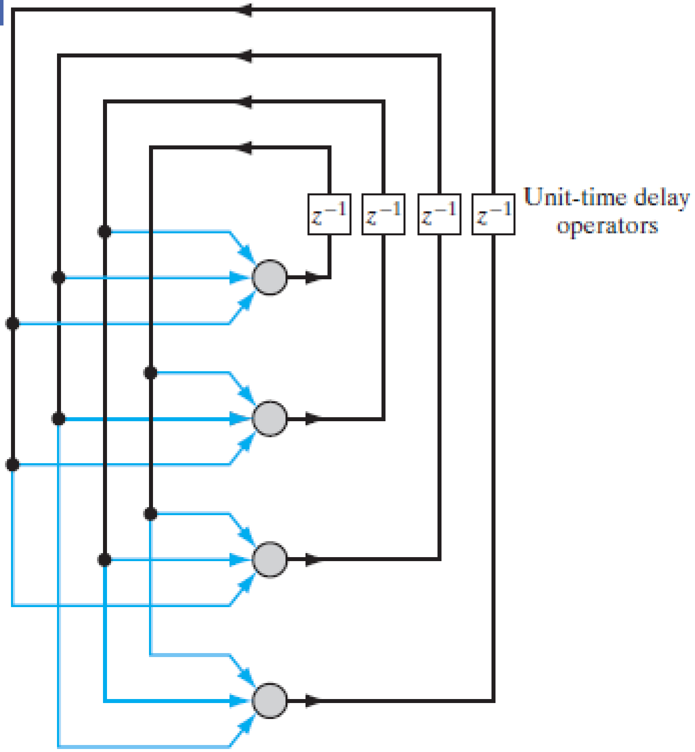
$$x = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_d \end{bmatrix} \quad w_0 = \begin{bmatrix} w_{0,1} \\ w_{0,2} \\ \cdot \\ \cdot \\ \cdot \\ w_{0,k} \end{bmatrix}$$

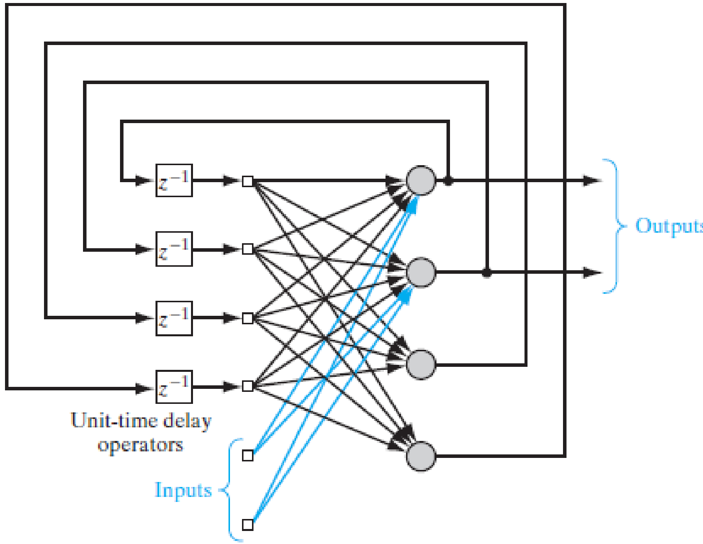




Single-layer and multi-layer

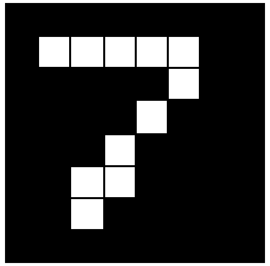
- **(iii) Recurrent Networks**
  - Network having at least one **feedback loop.**
  - Ex: a single layer of neurons with each neuron feeding its output back to its inputs of all other neurons (top figure).
  - Ex: output of a neuron is fed back to its own input (bottom figure) → called **self feedback loops.**

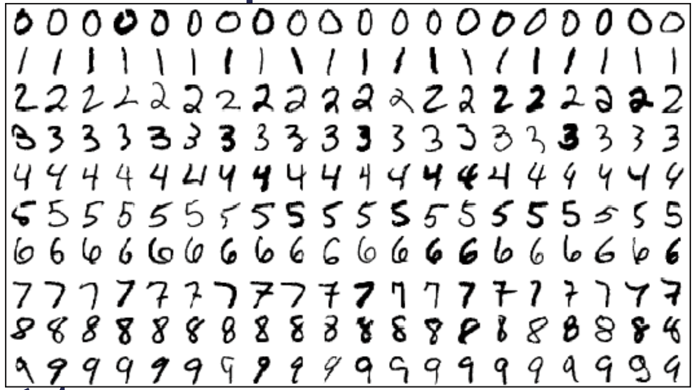




Some Network Architectures

- Consider a handwritten-digit recognition problem.
  - Input → pixels of the image each pixel as separate feature
  - Output → one of the 10 digits
  - Training set → large variety of handwritten digits that are representative of a real-world situation.





- Design of the network
  - Input dimension = number of pixels in the image.
  - Output = 10 neurons each represent one of the 10 digits

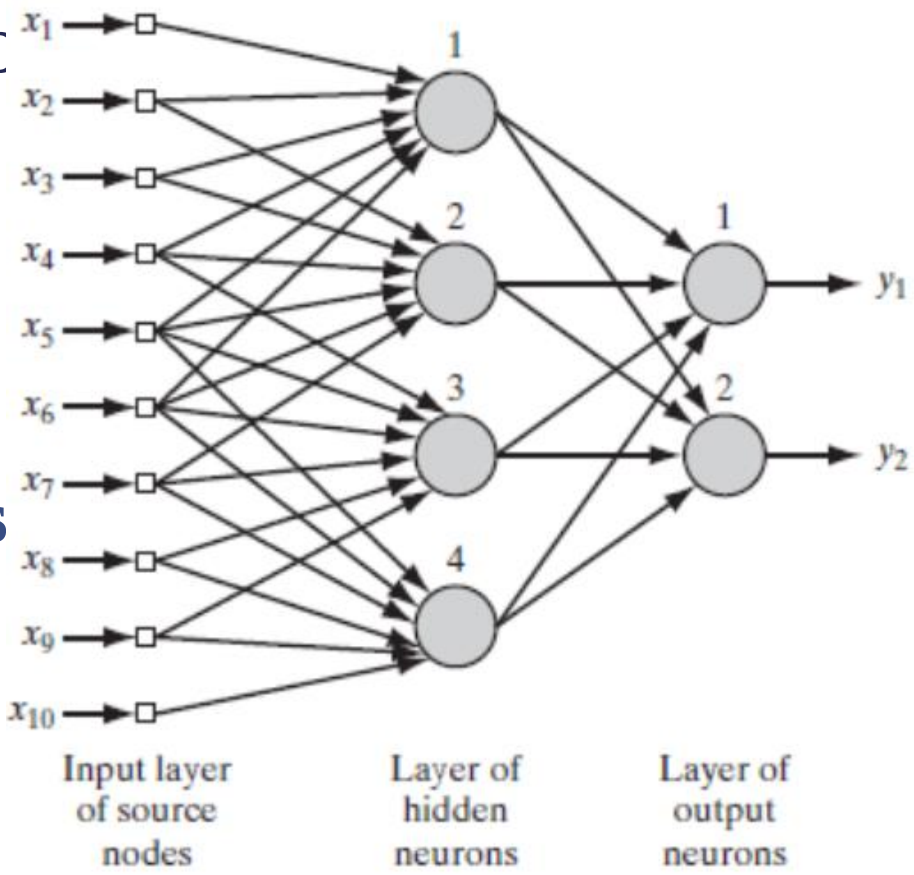
Knowledge Representation

- When representing knowledge respect the following rules:
  - Rule 1. Similar inputs(i.e. patterns drawn) from similar classes should usually produce similar representations inside the network and should therefore be classified as belonging to the same class.
  - Rule 2. Items to be categorized as separate classes should be given widely different representations in the network.
  - Rule 3. If a particular feature is important, then there should be large number of neurons involved in the representation of that item in the network.
  - Rule 4. Prior information and invariances should be built into the design of a neural network whenever they are available, so as to simplify the network deign by its not having to learn them.

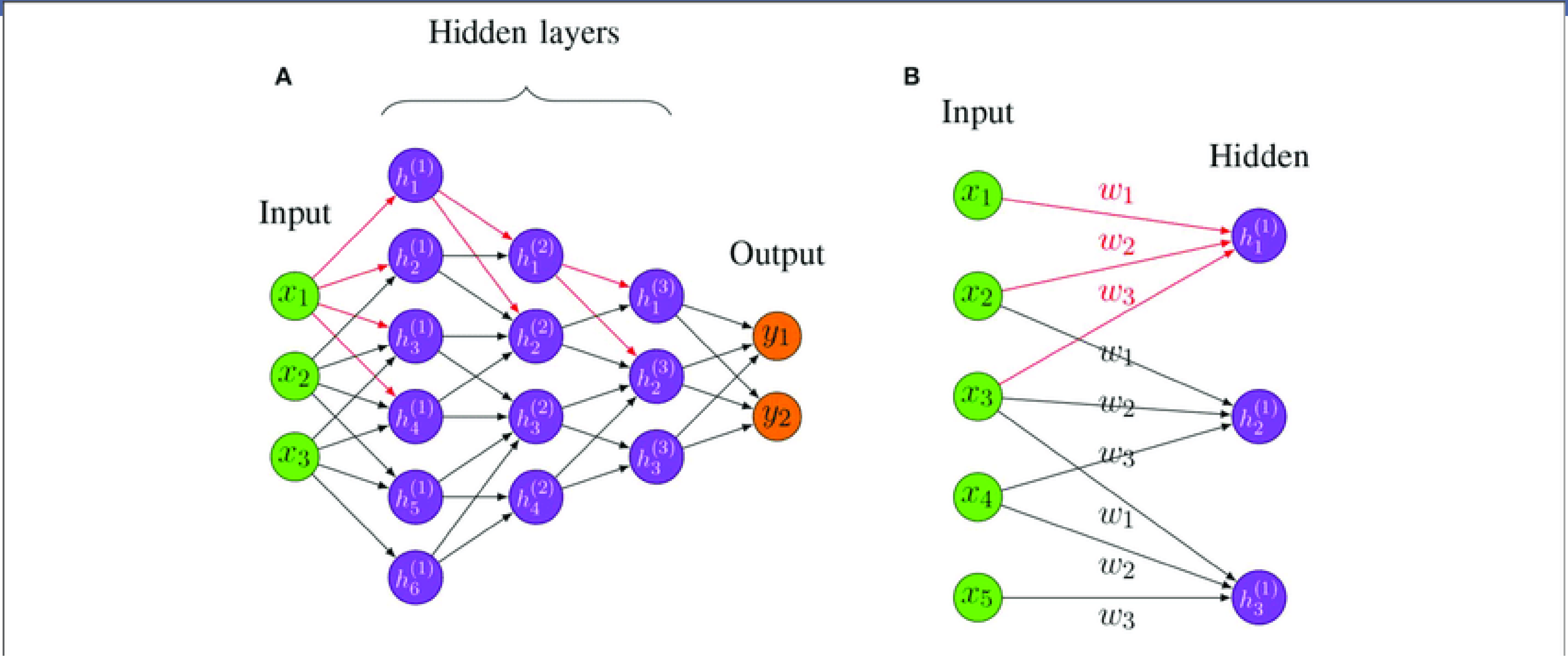
Knowledge Representation

Building prior information into NN design

- Examples of how to build prior informatic
- We have ad hoc procedures:
  - **Restricting the network architecture which use of <u>local connections</u> known as receptive**
  - **Constraining the choice of synaptic weights through the use of <u>weight sharing</u>.**



Partially connected feedforward network.



Example: Local connections and weights sharing

Building Invariances into NN Design

- Problem to consider during NN design:
  - Network trained to detect objects → what if objects appear rotated, translated, scaled, different colors than during training?
  - → The classifier should be **invariant** to these transformations.
- How to do that?
  - **Invariant by structure**
  - **Invariance by training**
  - **Invariant by feature space**

- Weights of neurons are created so that transformed versions of the same input are forced to produce the same output.
- Disadvantage → the number of synaptic connections becomes prohibitively large even for images of moderate size.

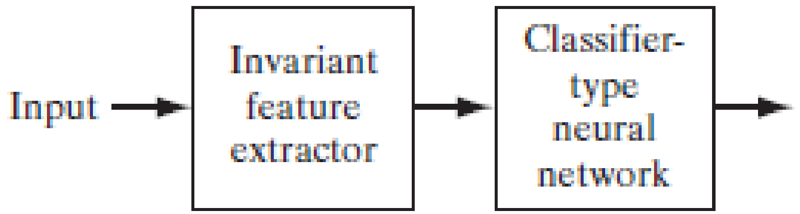
➢ **Invariance by training**

- Train the network to recognize the image and its rotations (different aspect) views.
- Data Augmentation → Generate more samples from your dataset by applying several type of transformations (rotated, scaled, translated, …).
- Disadvantages → Computational cost, overfitting, …

Building Invariances into NN Design

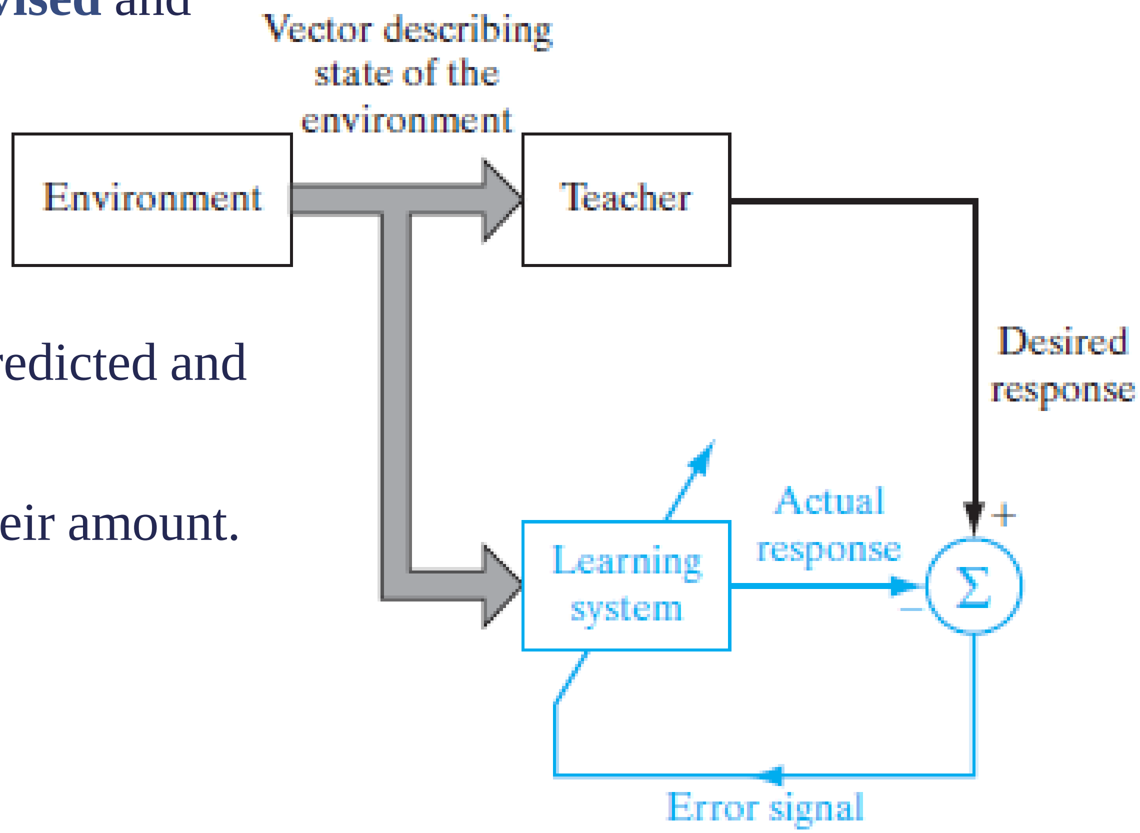
Invariant by feature space

- Extract features to characterize the essential information content of patterns that are invariant to transformations of the input.
- Advantages
  - The number of features applied may be reduced to realistic levels.
  - The requirements imposed to the network are relaxed.
  - Invariance for all objects with respect to known transformation is assured.
- Disadvantage
  - Not easy and should pass by feature engineering step to study the set of features.



Building Invariances into NN Design

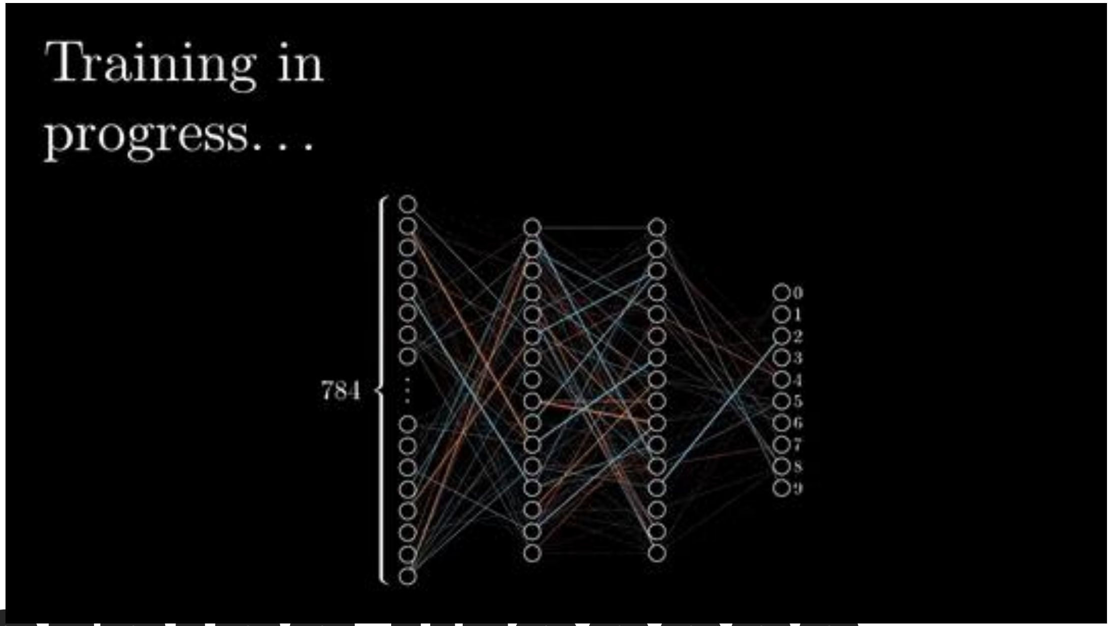
- Learning can be with a teacher or without a teacher.
  - Learning with a teacher can be categorized into **supervised** and **reinforced learning.**
- **Learning with a Teacher → Main steps**
  - Pattern in dataset are fed into the network
  - The error is calculated as the difference between the predicted and real label.
  - Weights are updated in backward manner relative to their amount.
  - Updates is done until reaching an optimum value

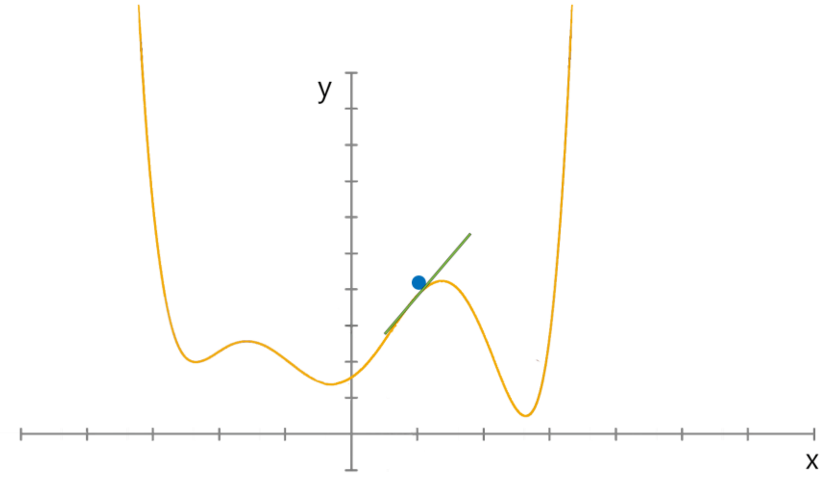


Learning Process

Learning with a Teacher

- Updating the weights is done in the direction of the gradient
- The update is done in one of the three ways:
  - Batch-training → one update after processing a batch of samples.
  - Stochastic training → one update after processing one individual sample.
  - Mini-batch training → one update after processing a small subset of data

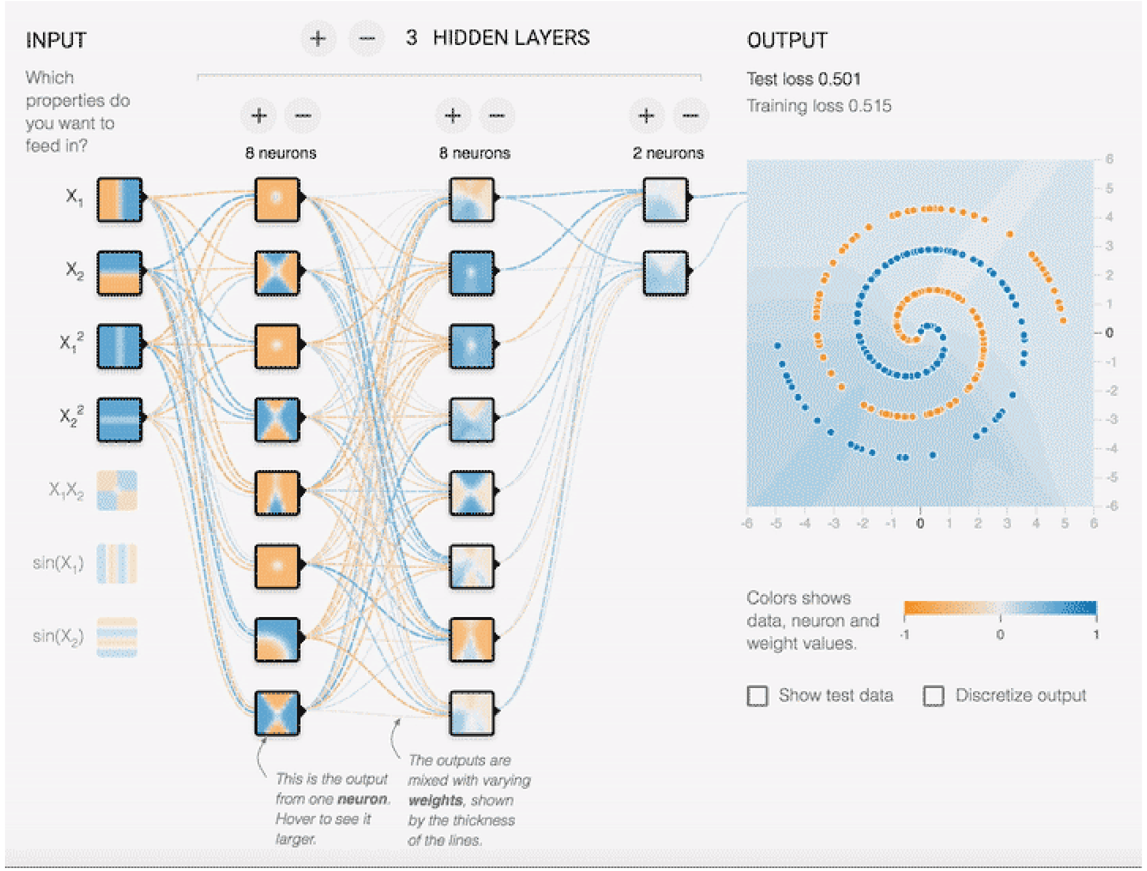




Learning Process

Error function

Weights updating



Learning Process