ARTIFICIAL INTELLIGENCE

ARTIFICIAL NEURAL NETWORKS-II

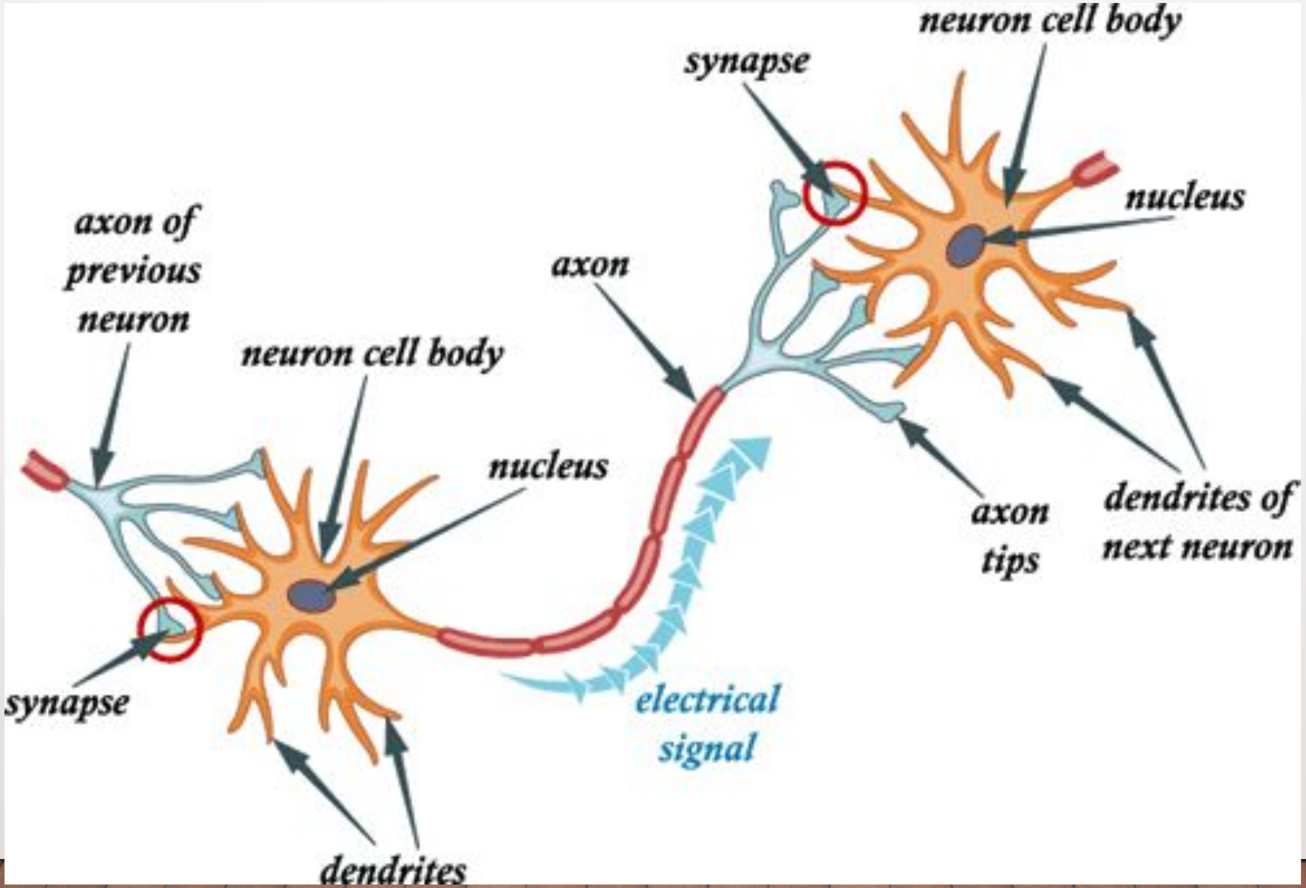
CS-632

Dr. Muhammad Aqib

University Institute of Information Technology

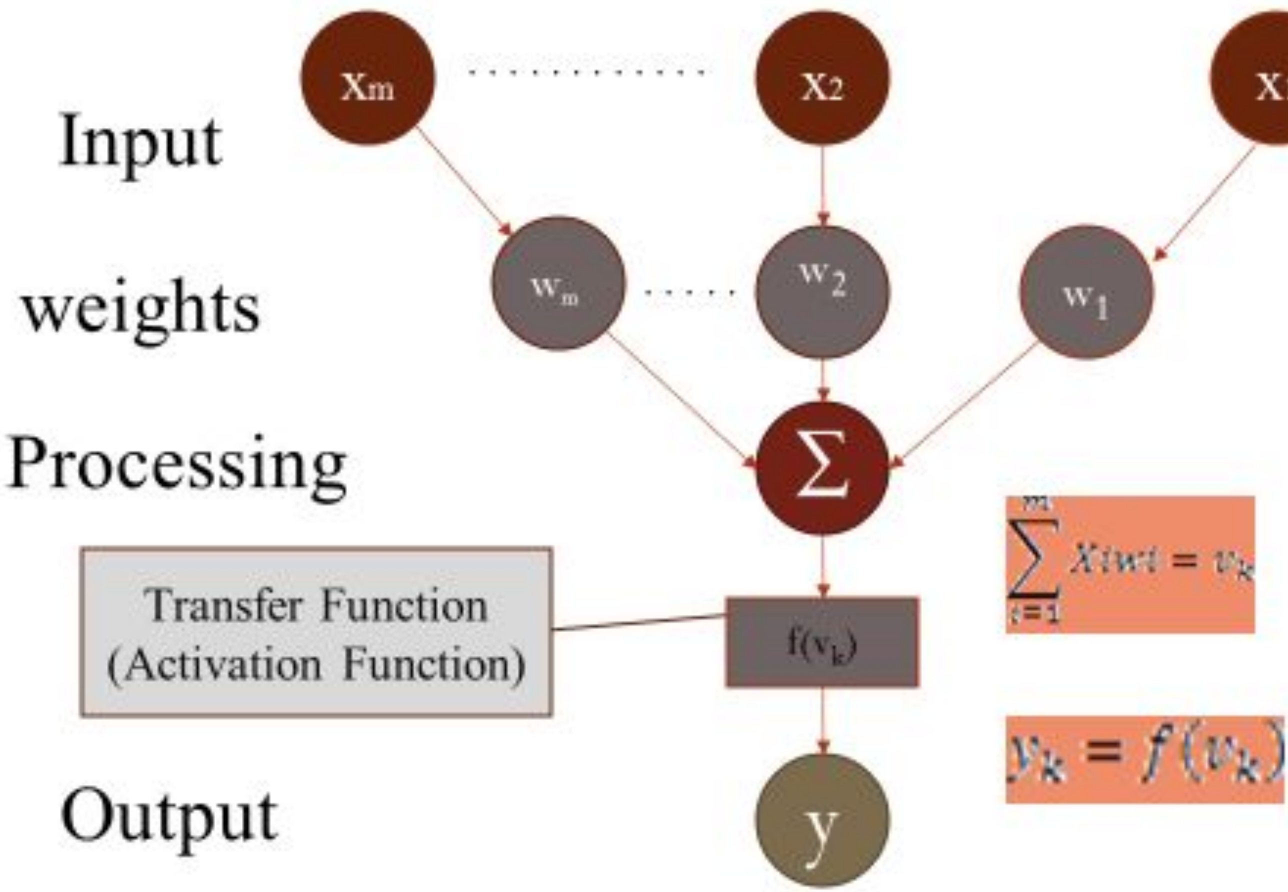
PMAS-Arid Agriculture University Rawalpindi

THE PARTS OF A NEURON



How do ANNs work?

The signal is not passed down to the next neuron verbatim



ACTIVATION FUNCTIONS

- Step(x) = 1 if x >= t, else 0
- Sign(x) = +1 if x >= 0, else –1
- Sigmoid(x) = 1/(1+e-x)
- Identity Function A(x) = X

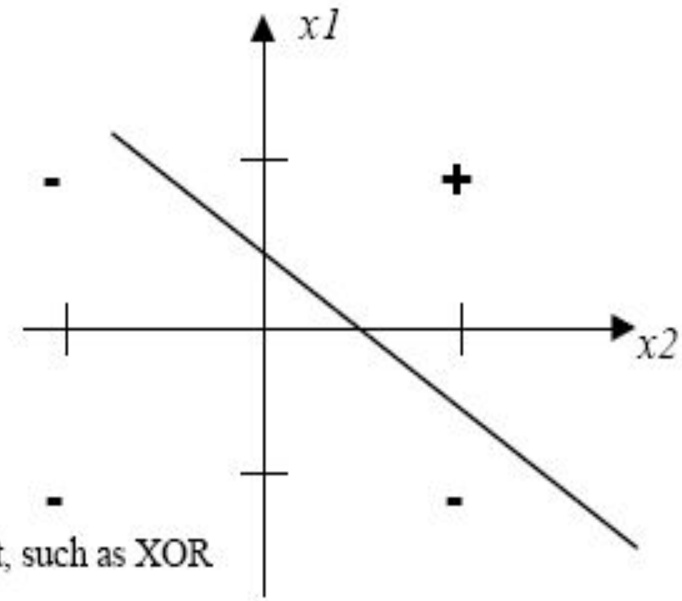
REPRESENTATIONAL POWER OF PERCEPTRONS

- Perceptrons can represent the logical AND, OR, and NOT functions as above.
- we consider 1 to represent True and -1 to represent False.

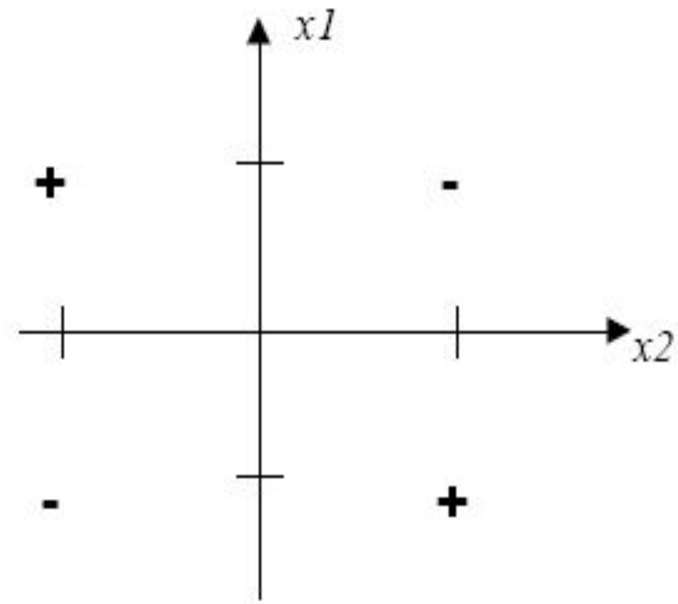
- Here there is no way to draw a single line that separates the "+" (true) values from the "-"

(false) values.

In general, perceptrons can learn linearly separable functions, such as AND:



But not those that are not, such as XOR



TRAIN A PERCEPTRON

- At start of the experiment there are W random values
- Than the training begins with objective of teaching it to differentiate two classes of inputs I and II
- The goal is to have the nodes output o = 1 if the input is of class I , and to have

  o= -1 if the input is of class II
- You can free to choose any inputs (X**i**) and to designate them as being of class I or II

TRAIN A PERCEPTRON

- If the node happened to output 1 signal when given a class II input or output -1 signal when given a class I input the weight Wi has no change.

- Then the training is complete.

SINGLE LAYER PERCEPTRON

- For a problem which calls for more then 2 classes, several perceptrons can be combined into a network.

- Can distinguish only linear separable functions.

SINGLE LAYER PERCEPTRON

Single layer, five nodes. 2 inputs and 3 outputs

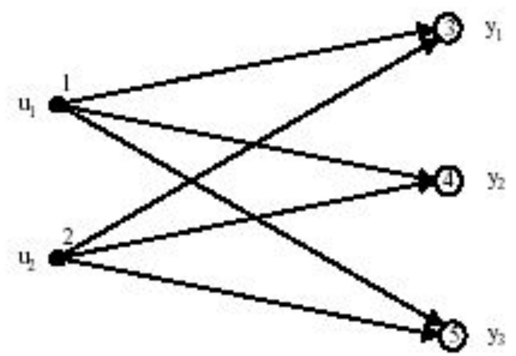


Figure 6: Single layer perceptron network with three output neurons.

Recognizes 3 linear separate classes, by means of 2 features

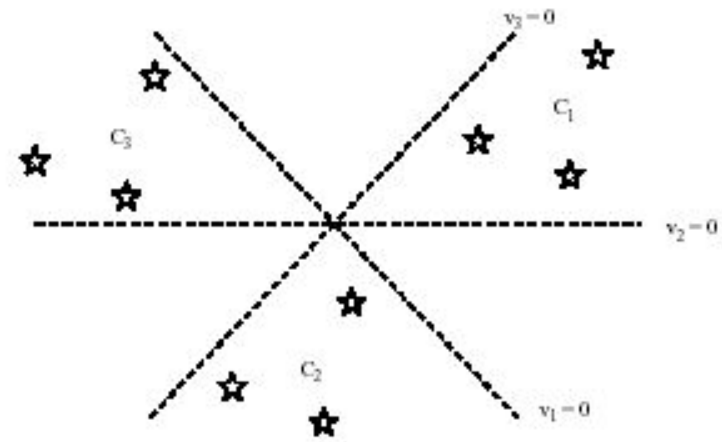
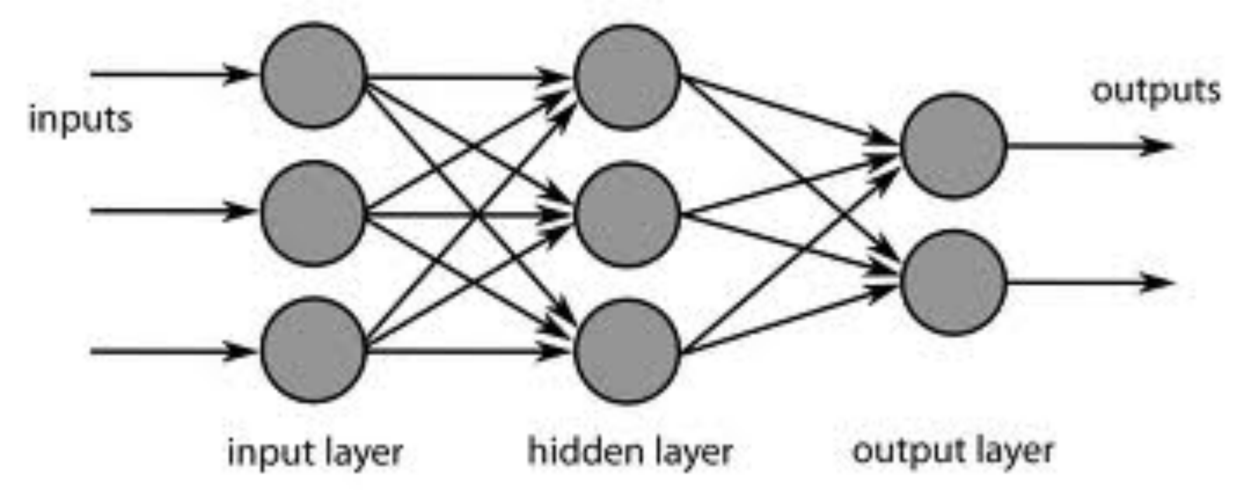


Figure 7: Three linearly separable classes.

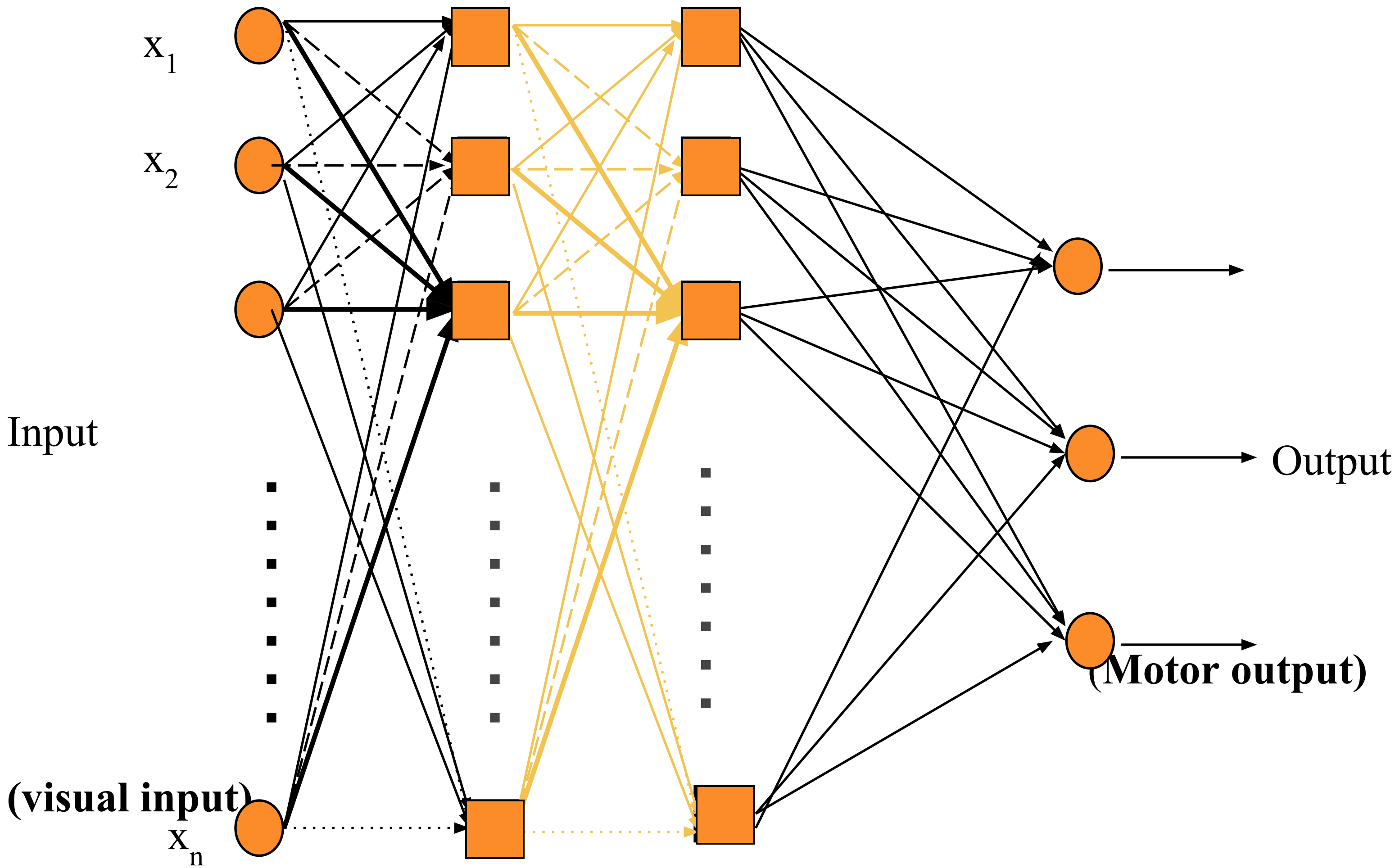
MULTI-LAYER NETWORKS

MULTI-LAYER NETWORKS

- A Multi layer perceptron can classify non linear separable problems.
- A Multilayer network has one or more hidden layers.



Multi-layer networks

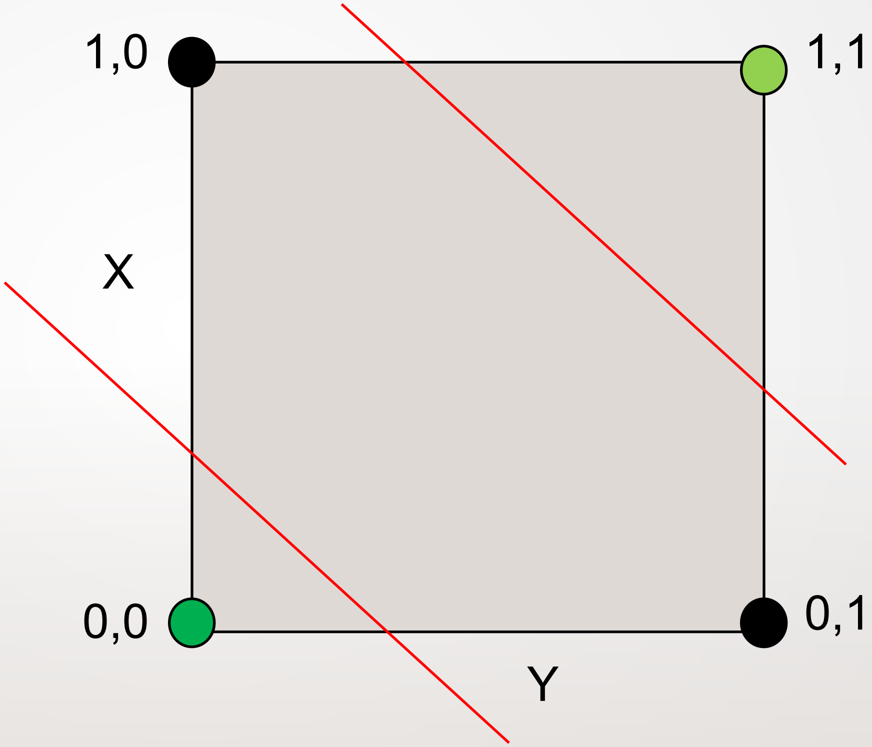


Hidden layers

EXAMPLE

Logical XOR
Function

X	Y	output
0	0	0
0	1	1
1	0	1
1	1	0



XOR

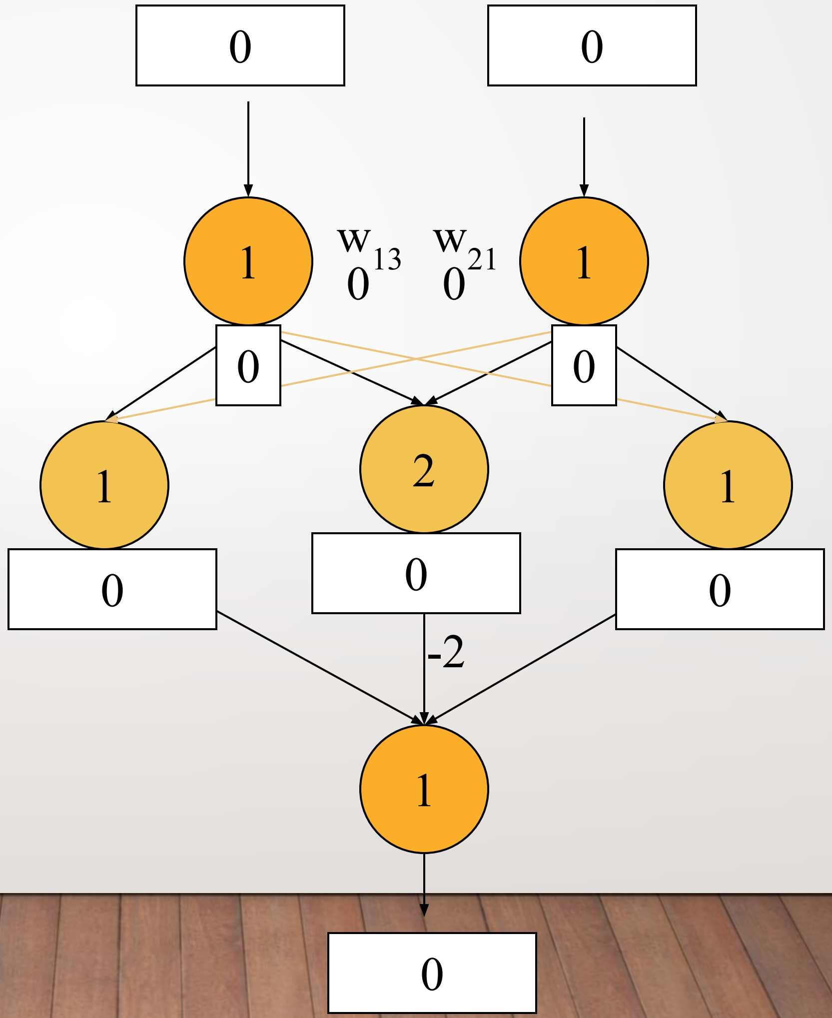
- XOR

Activation Function:

if (input >= threshold), fire

else, don’t fire

All weights are 1, unless otherwise labeled.



XOR

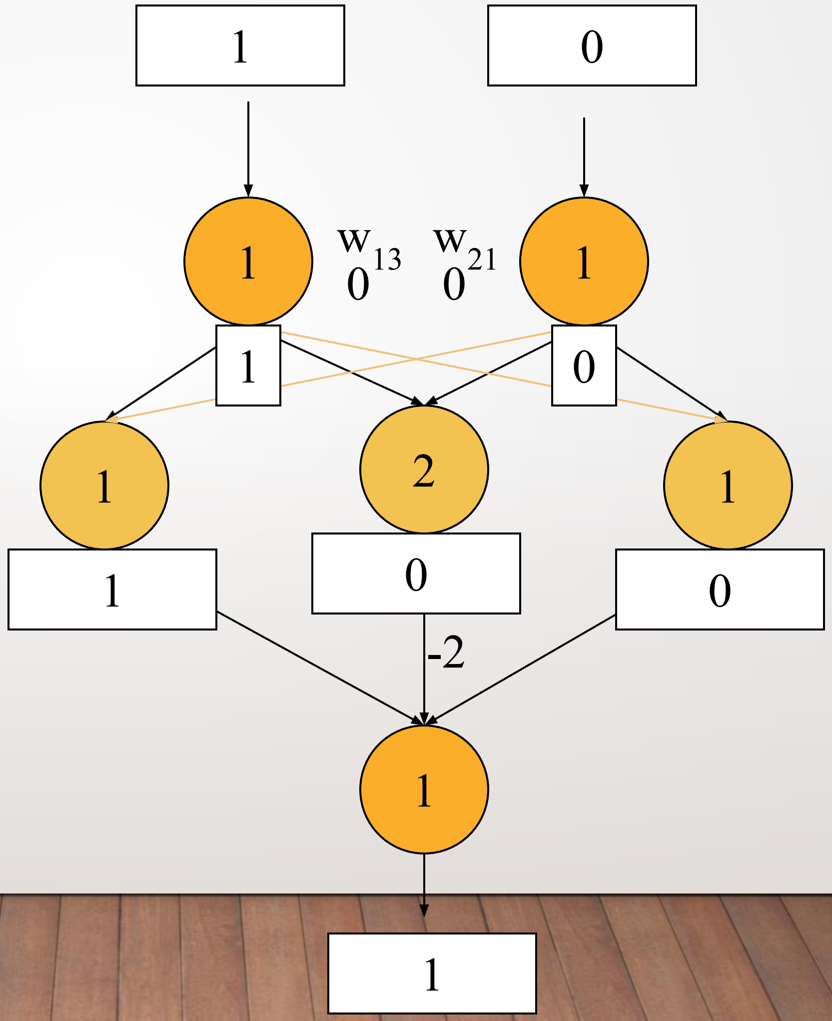
- XOR

Activation Function:

if (input >= threshold), fire

else, don’t fire

All weights are 1, unless otherwise labeled.



XOR

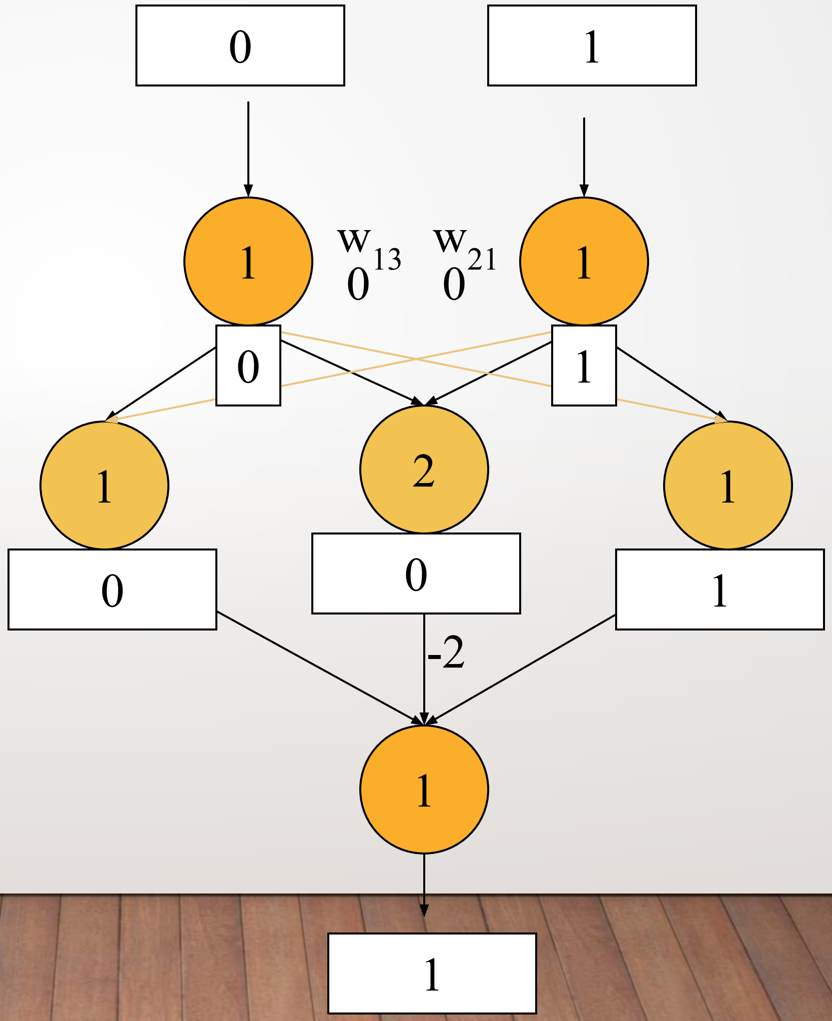
- XOR

Activation Function:

if (input >= threshold), fire

else, don't fire

All weights are 1, unless otherwise labeled.



XOR

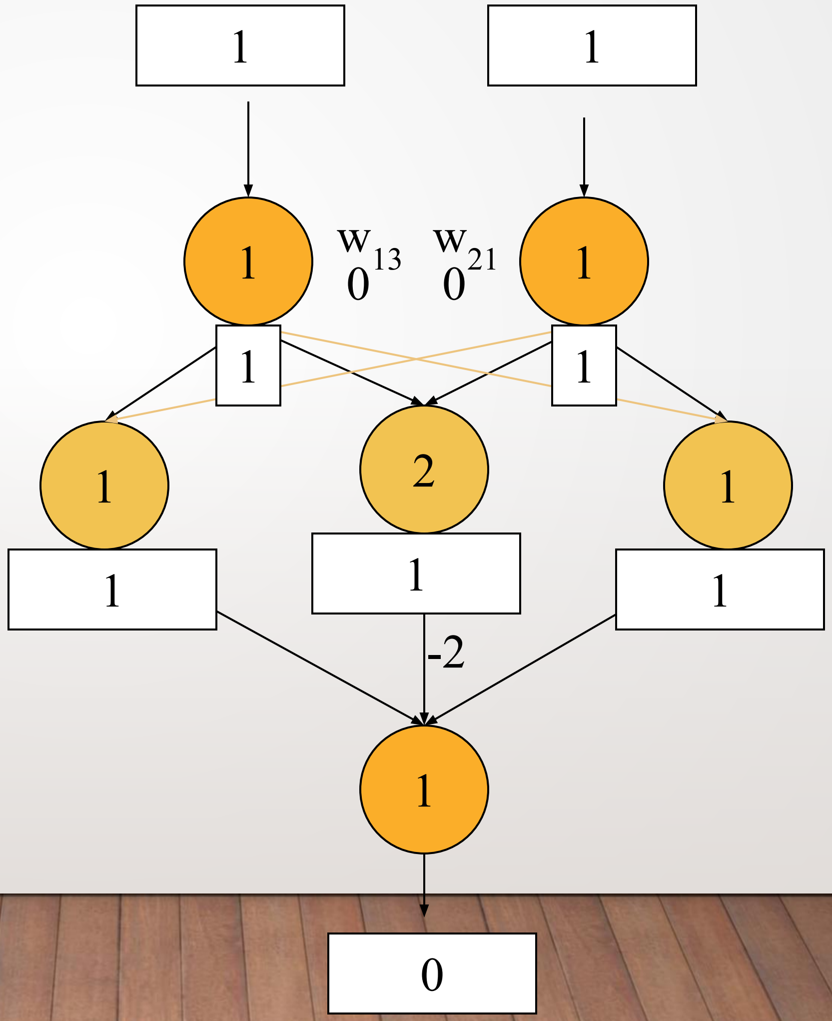
- XOR

Activation Function:

if (input >= threshold), fire

else, don't fire

All weights are 1, unless otherwise labeled.



Example: A Classification Task

- A typical neural network application is classification. Consider the simple example of classifying trucks given their masses and lengths

How do construct a neural network that can classify any Lorry and Van?

MASS	LENGTH	CLASS
10.0	6	LORRY
20.0	5	LORRY
5.0	4	VAN
2.0	5	VAN
2.0	5	VAN
3.0	6	LORRY
10.0	7	LORRY
15.0	8	LORRY
5.0	9	LORRY

TRAINING MULTILAYER PERCEPTRON

- The training of multilayer networks raises some important issues:
- How many layers ?, how many neurons per layer ?
- **Too few neurons** makes the network unable to learn the desired behavior.
- **Too many neurons** increases the complexity of the learning algorithm.

TRAINING MULTILAYER PERCEPTRON

- A desired property of a neural network is its ability to generalize from the training set.
- If there are **too many neurons**, there is the danger of over fitting.

TRAINING PERCEPTRONS

- Learning involves choosing values for the weights
- The perceptron is trained as follows:
  - First, inputs are given random weights (usually between –0.5 and 0.5).
  - An item of training data is presented. If the perceptron mis-classifies it, the weights are modified according to the following:

$$w_i \leftarrow w_i + \left(a \times x_i \times \left(t - o\right)\right)$$

    - where *t* is the target output for the training example, *o* is the output generated by the preceptron and *a* is the learning rate, between 0 and 1 (usually small such as 0.1)
- Cycle through training examples until successfully classify all examples
  - Each cycle known as an **epoch**

BACKPROPAGATION

- Multilayer neural networks learn in the same way as perceptrons.
- However, there are many more weights, and it is important to assign credit (or blame) correctly when changing weights.
- *E* sums the errors over all of the network output units

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

BACKPROPAGATION ALGORITHM

- Create a feed-forward network with n_{in} inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers
- Until termination condition is met, Do
  - For each <x,t> in training examples, Do

    Propagate the input forward through the network:

    1. Input the instance x to the network and compute the output o_u of every unit u in the network

    Propagate the errors backward through the network:

    2. For each network output unit k, calculate its error term δ_k

    $$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$

    3. For each hidden unit h, calculate its error term δ_h

    $$\delta_h \leftarrow o_h(1-o_h)\sum_{k \in outputs} w_{kh}\delta_k$$

    4. Update each network weight w_{ji}

    $$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

    where

    $$\Delta w_{ji} = \alpha \delta_j x_{ji}$$

PROBLEMS WITH TRAINING:

- Nets get stuck
  - Not enough degrees of freedom
  - Hidden layer is too small
- Training becomes unstable
  - too many degrees of freedom
  - Hidden layer is too big / too many hidden layers
- Over-fitting
  - Can find every pattern, not all are significant. If neural net is "over-fit" it will not generalize well to the testing dataset

Thank You