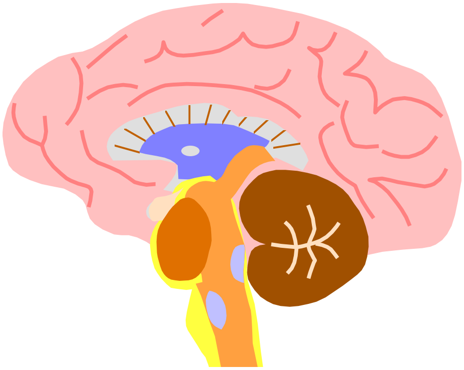
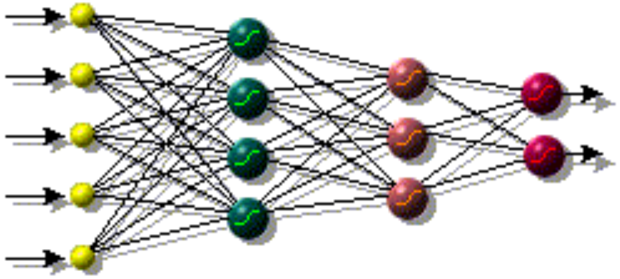
Artificial Neural Networks

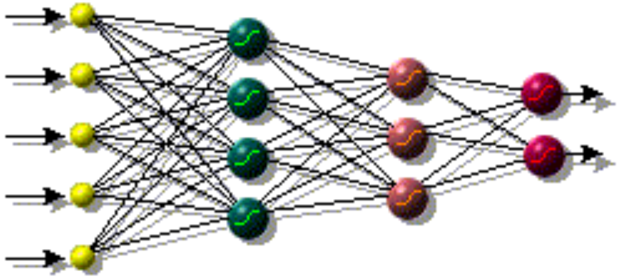
System that can acquire, store, and utilize experiential knowledge





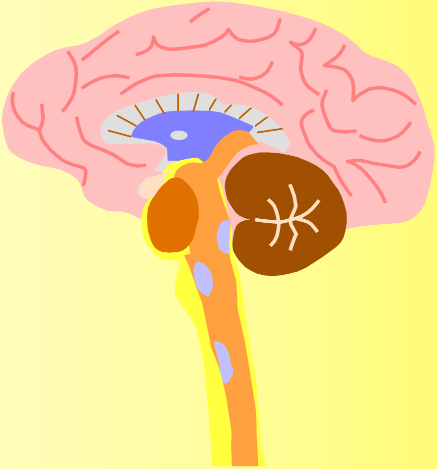
Overview

- The Brain
- Brain vs.. Computers
- The Perceptron
- Multi-layer networks
- Some Applications



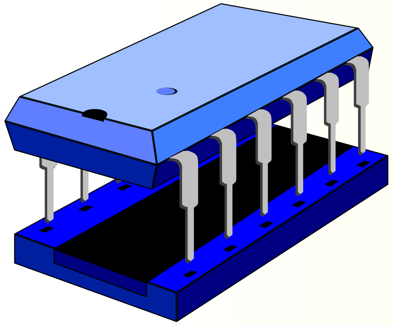
Brain and Machine

- The Brain
  - Pattern Recognition
  - Association
  - Complexity
  - Noise Tolerance



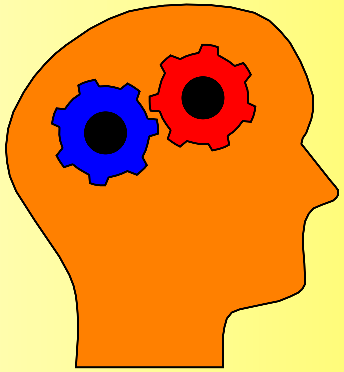
- The Machine
  - Calculation
  - Precision
  - Logic

The contrast in architecture



- The Von Neumann architecture uses a single processing unit;
  - Tens of millions of operations per second
  - Absolute arithmetic precision

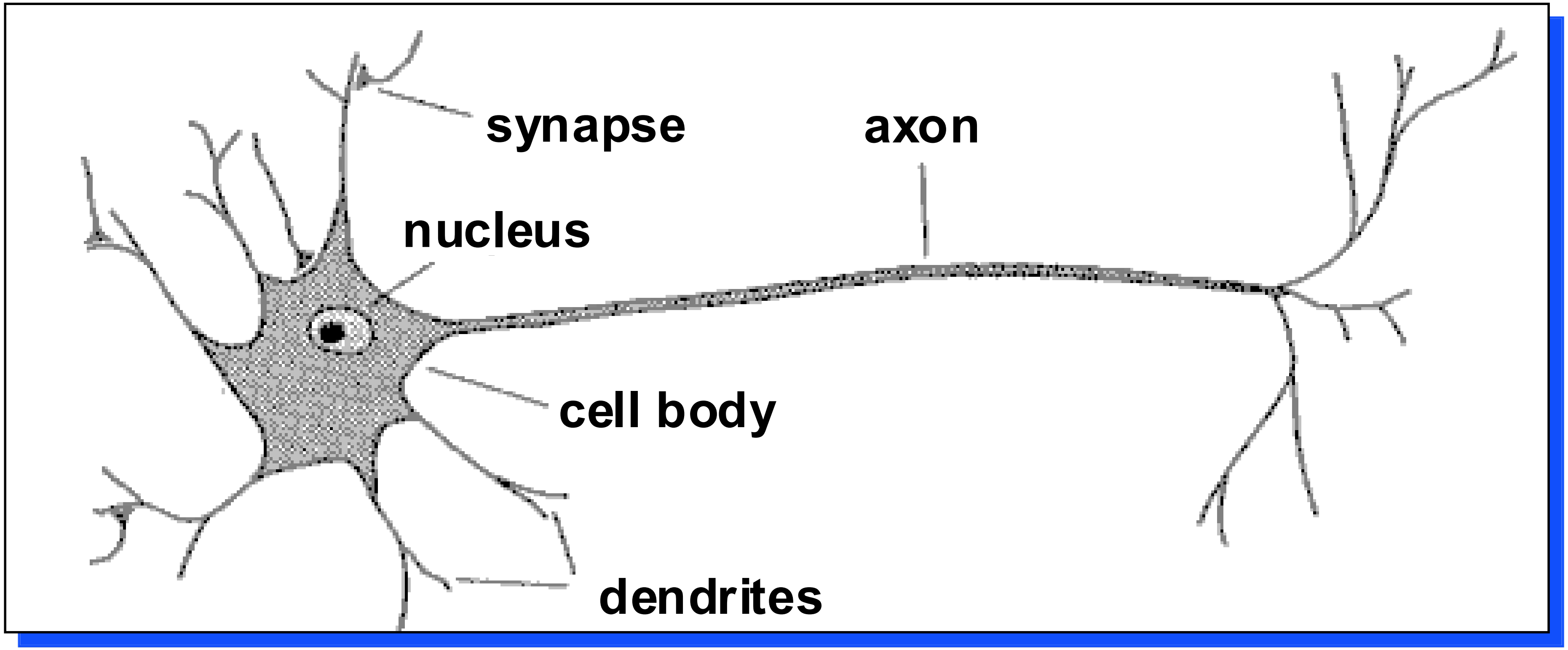
- The brain uses many slow unreliable processors acting in parallel



Features of the Brain

- Ten billion (10^{10}) neurons
- On average, several thousand connections
- Hundreds of operations per second
- Die off frequently (never replaced)
- Compensates for problems by massive parallelism

The Structure of Neurons



The Structure of Neurons

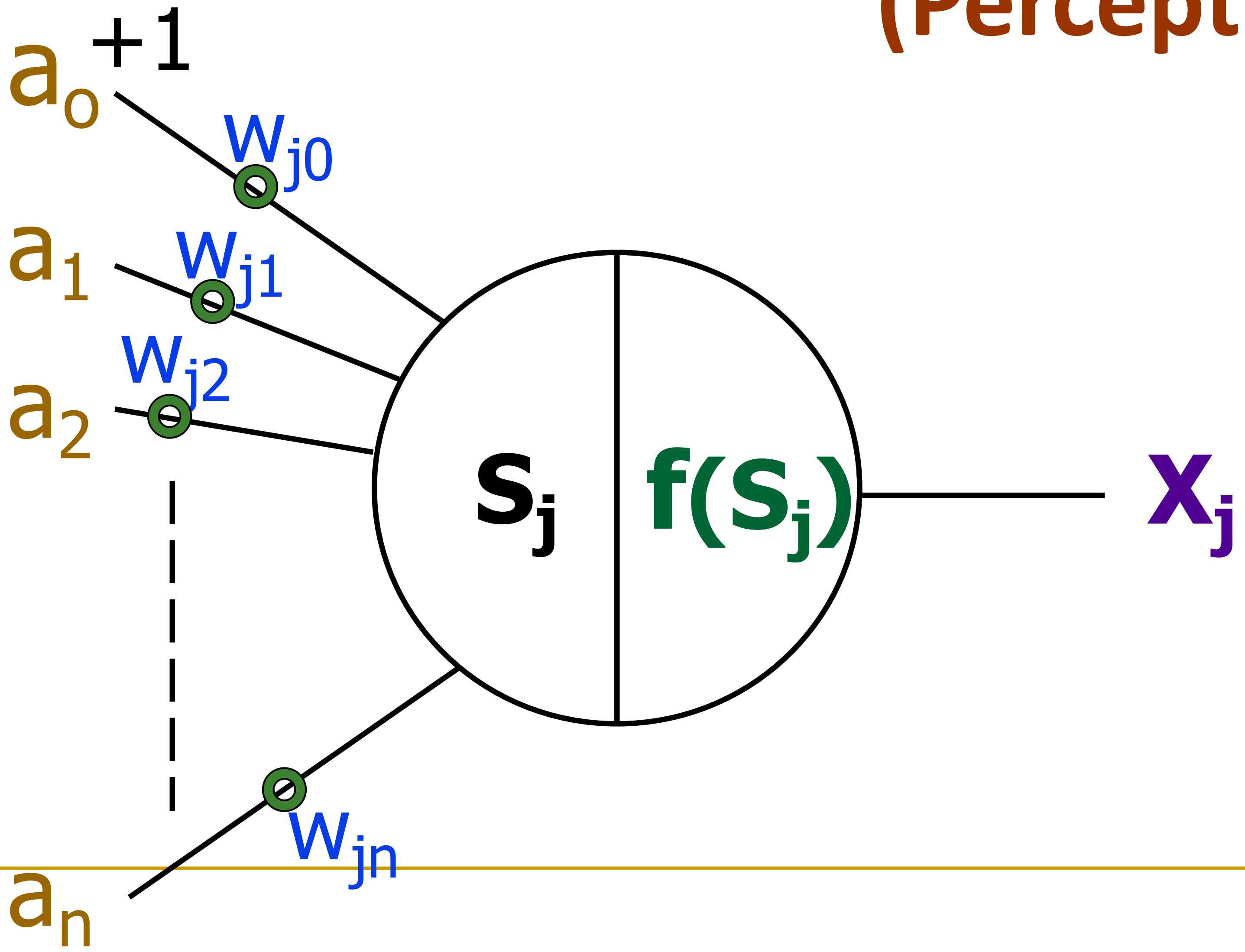
A neuron has a cell body, a branching **i**nput structure (the dendr**I**te) and a branching **o**utput structure (the ax**O**n)

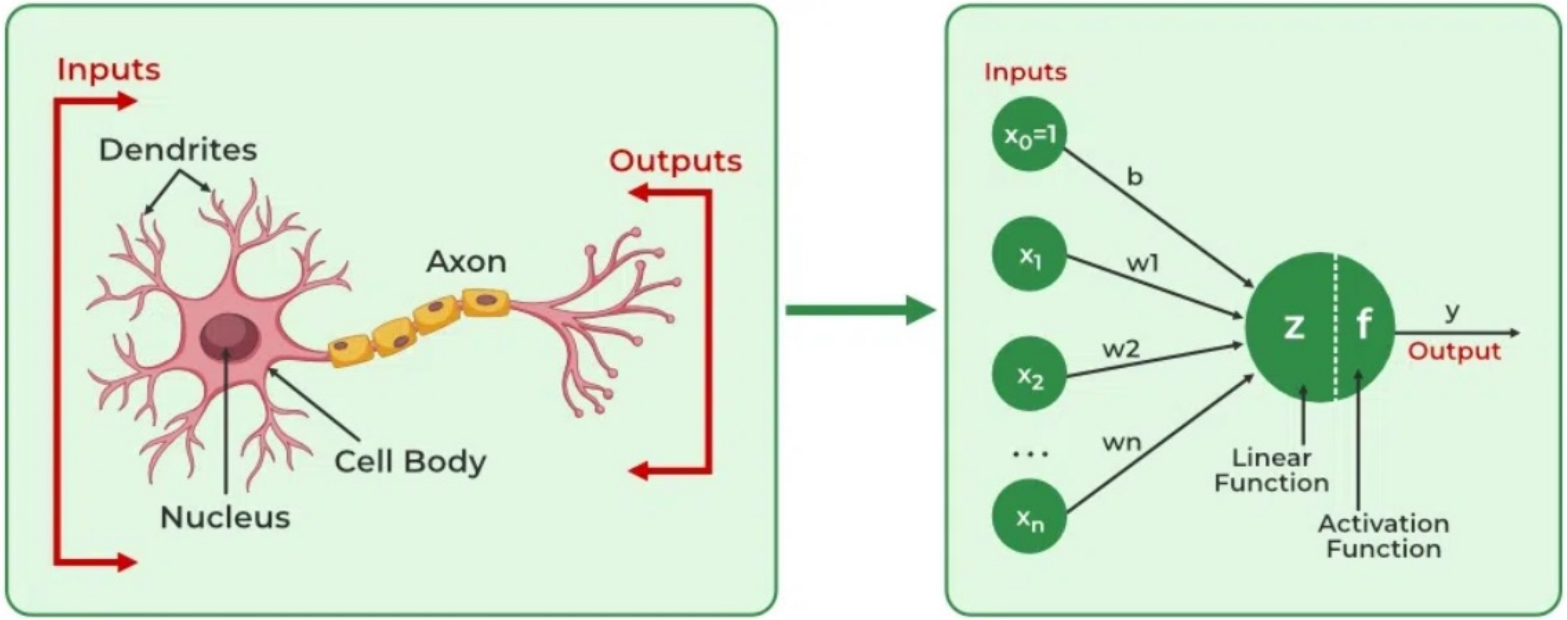
- **Axons connect to dendrites via synapses**
- **Electro-chemical signals are propagated from the dendritic input, through the cell body, and down the axon to other neurons**

The Structure of Neurons

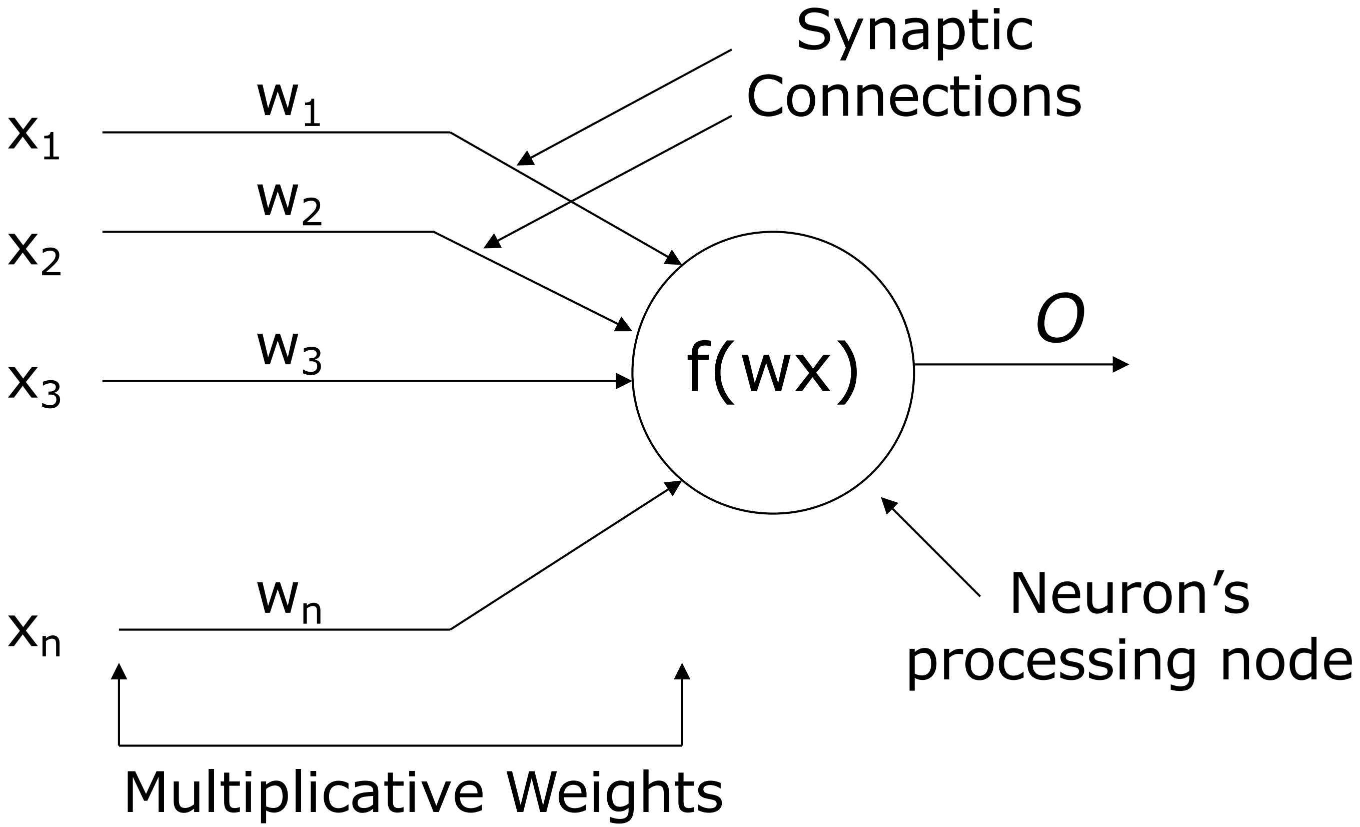
- A neuron only fires if its input signal exceeds a certain amount (the threshold) in a short time period
- Synapses vary in strength
  - Good connections allowing a large signal
  - Slight connections allow only a weak signal
  - Synapses can be either excitatory or inhibitory

The Artificial Neuron (Perceptron)




Inputs
Dendrites
Outputs
Axon
Cell Body
Nucleus
Inputs
$x_0=1$
b
x_1
w1
x_2
w2
...
wn
x_n
z
f
y
Output
Linear Function
Activation Function

General Symbols

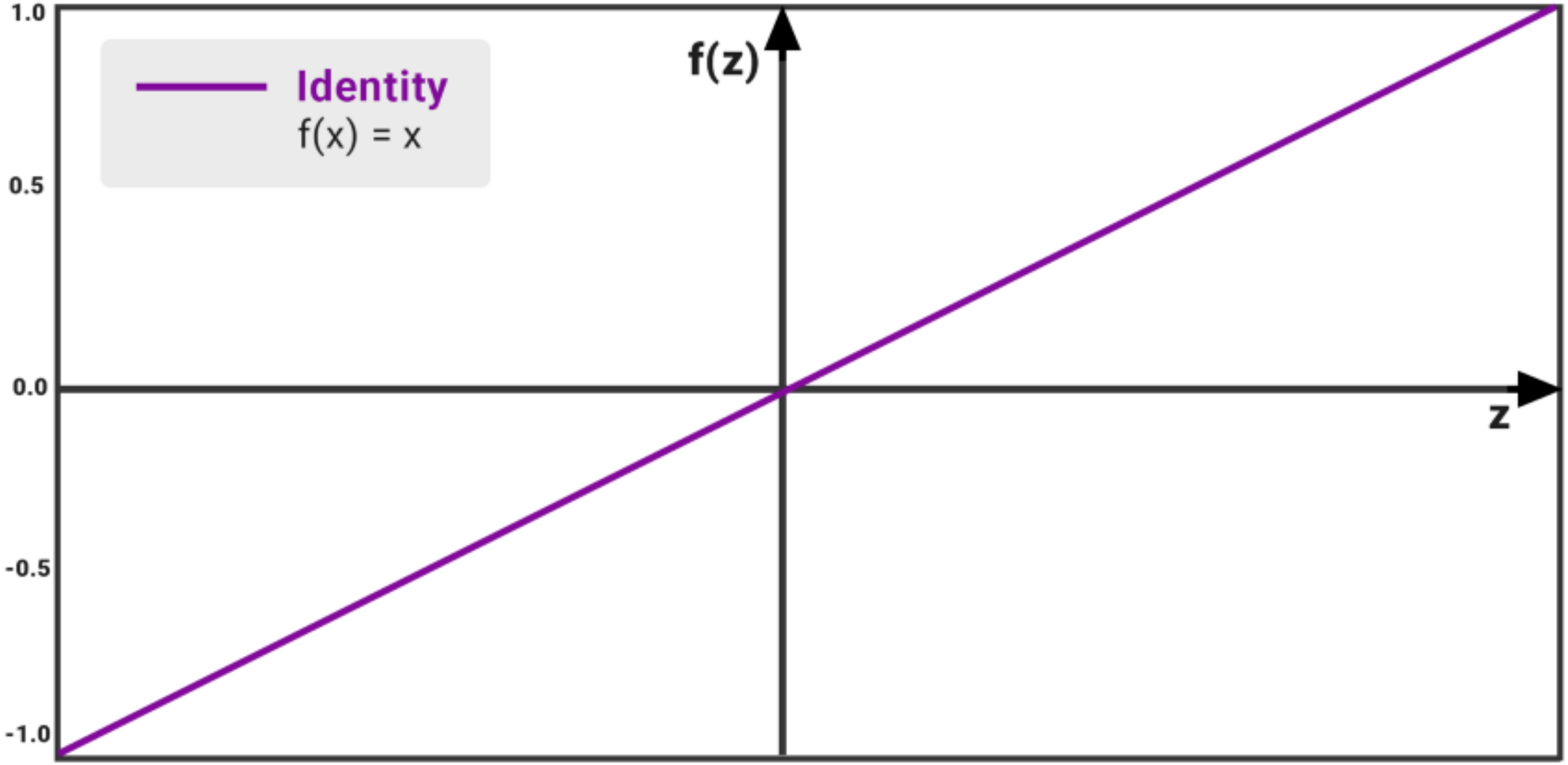


- **W**=[$w_1, w_2, \ldots w_n$]^T
- **X**=[$x_1, x_2, \ldots x_n$]^T

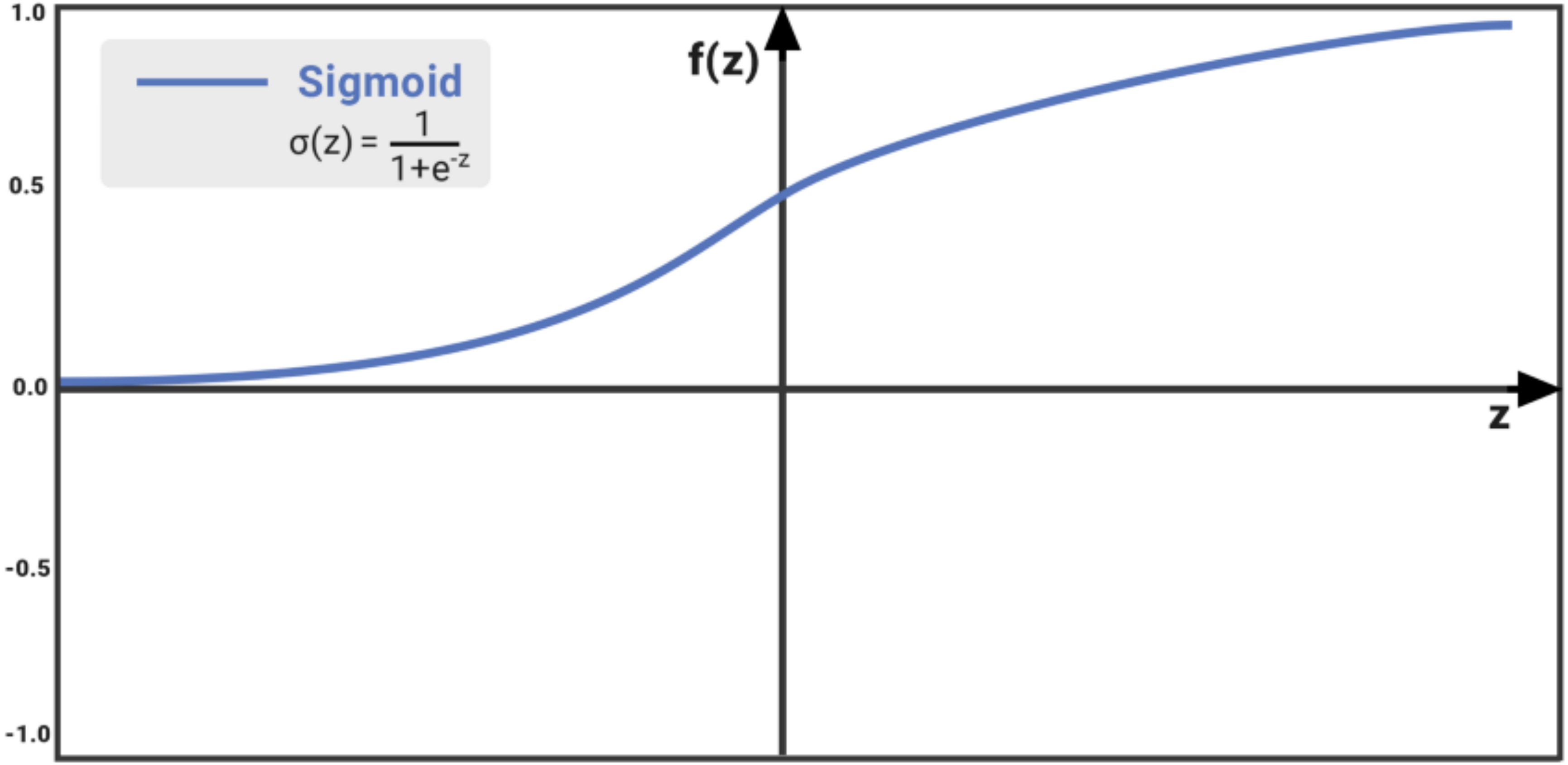
$$O = f(W^T X)$$

$$O = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

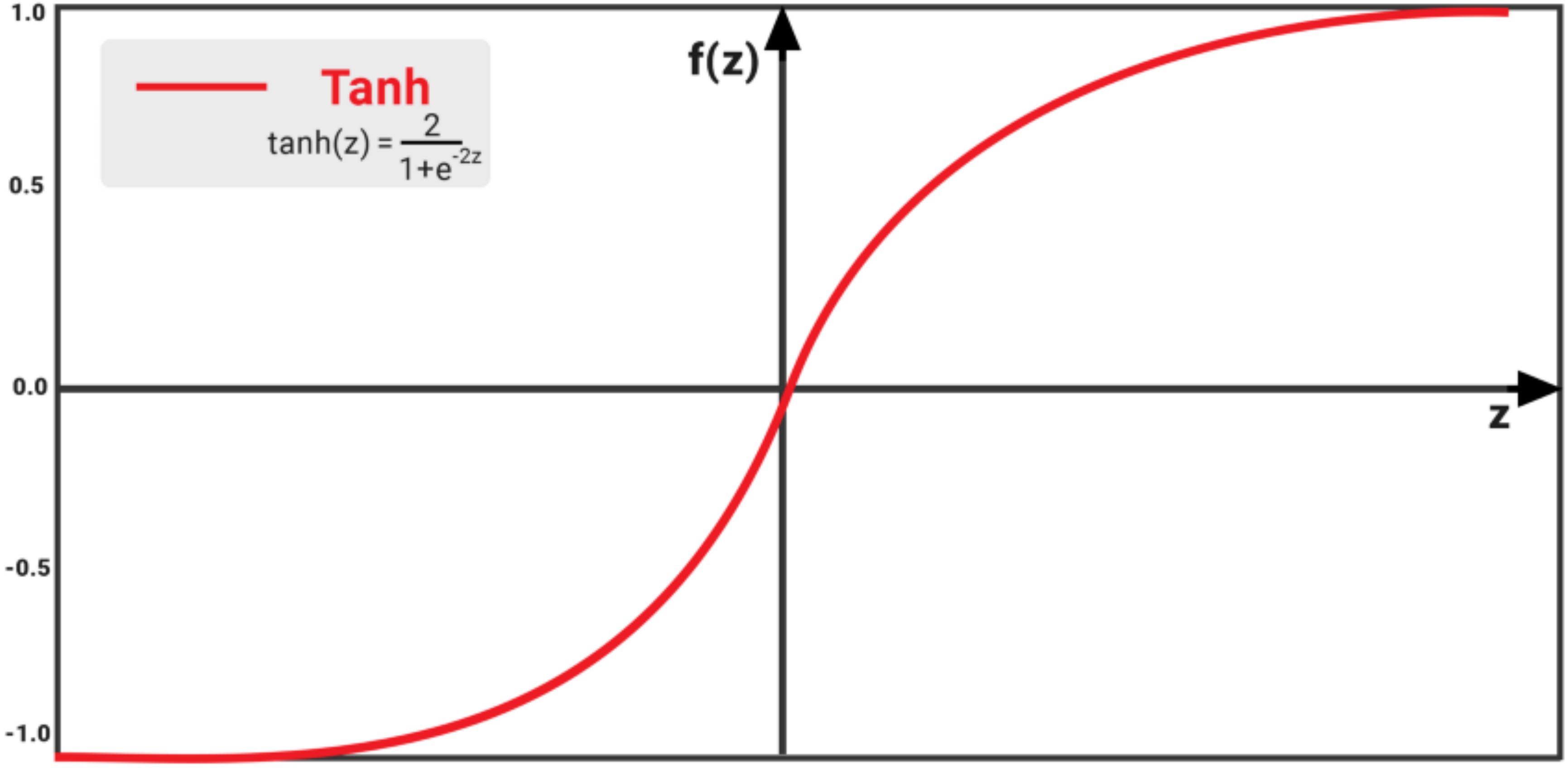
Linear Function



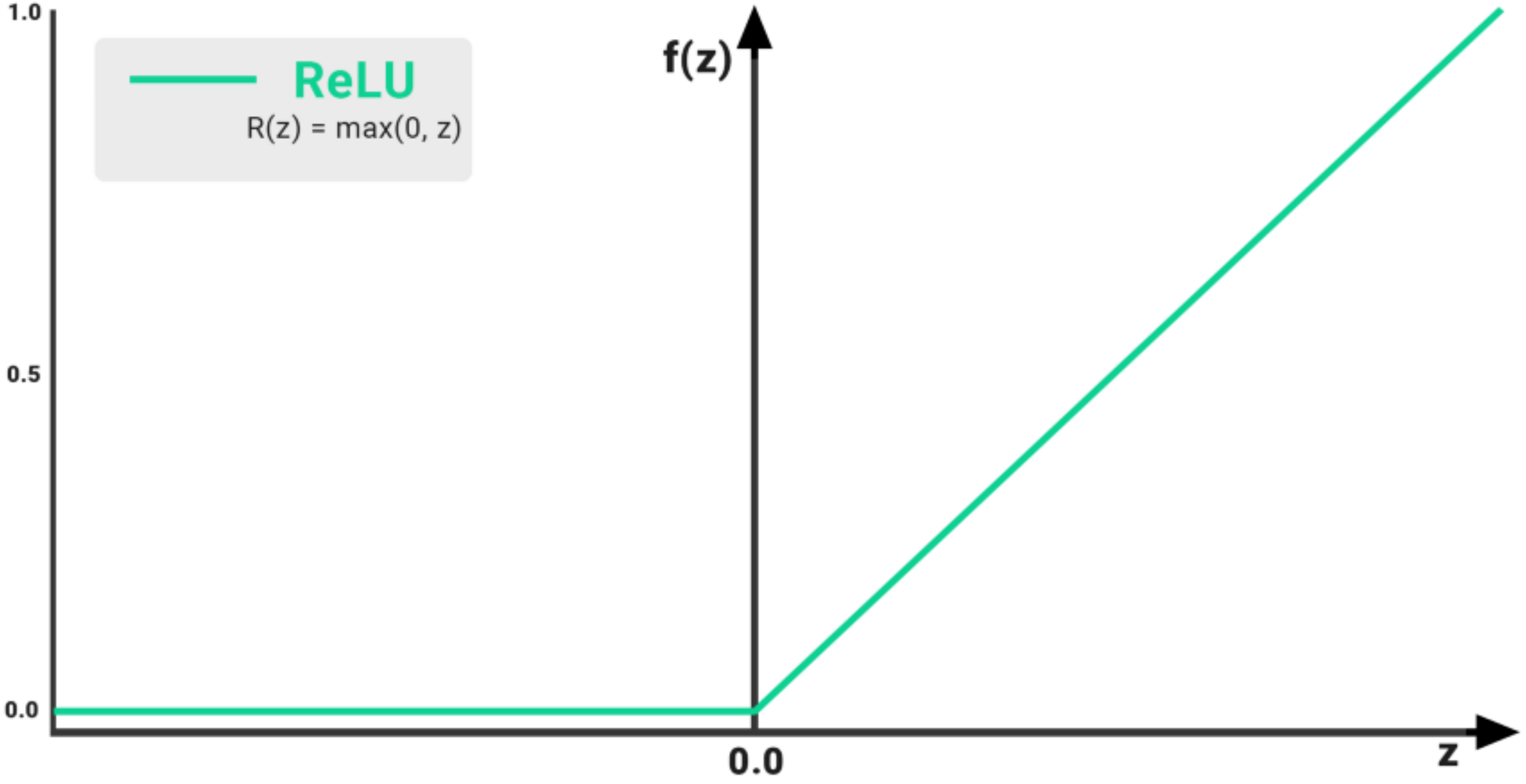
Sigmoid



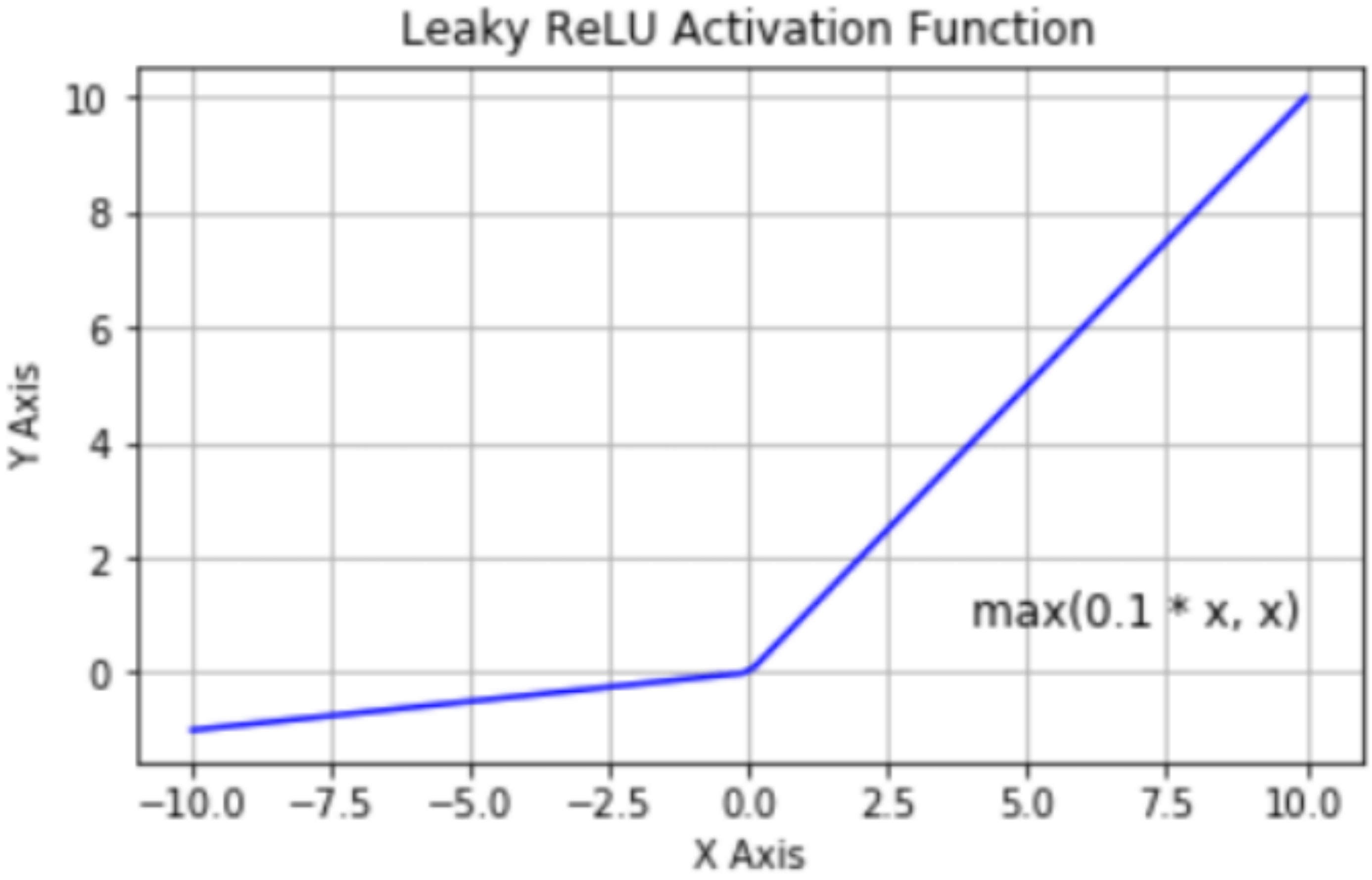
Tanh



ReLU



Leaky ReLU


Leaky ReLU Activation Function
10
8
6
4
2
0
Y Axis
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
X Axis
max(0.1 * x, x)

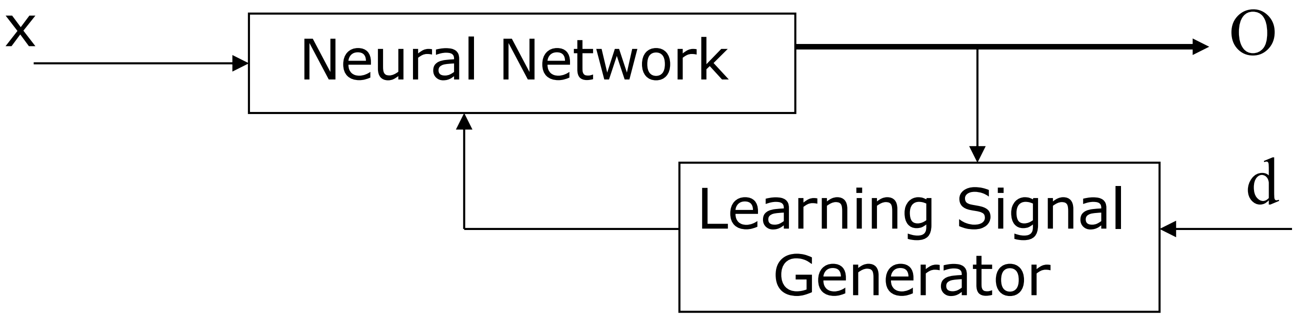
Soft Max

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

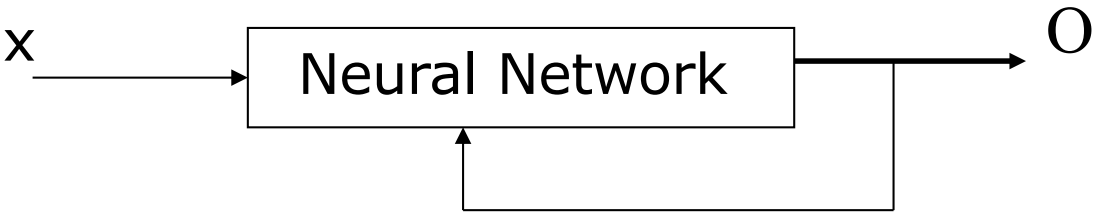
Learning

- Leaning in network is to find W – weights that best approximate training patterns
- Supervised learning
- like class room teaching
- Training set is needed
- Teacher estimates negative error gradient direction and reduces error accordingly
- Unsupervised learning
- Like video lecture
- No feedback for error
- Used for clustering

Supervised/Unsupervised



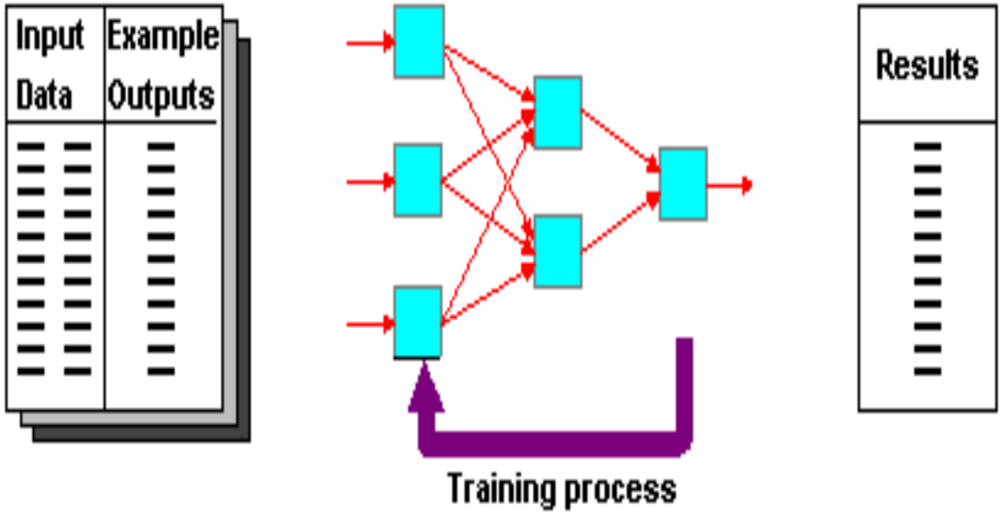
Supervised Learning

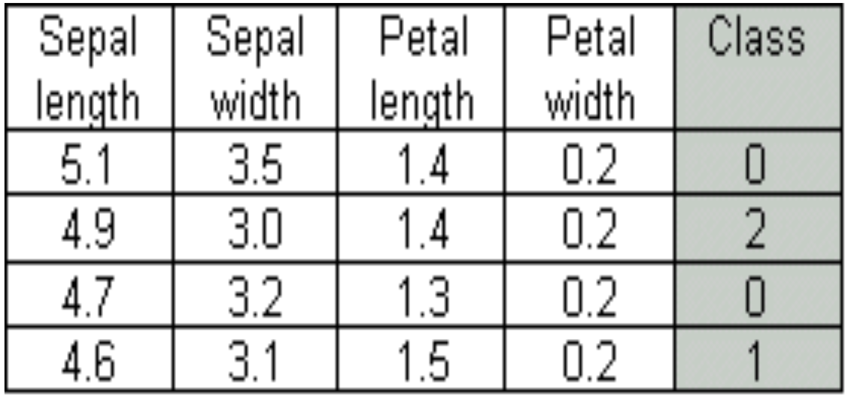


Unsupervised Learning

Supervised Vs Unsupervised

- Training and test data sets
- Training set; input & target



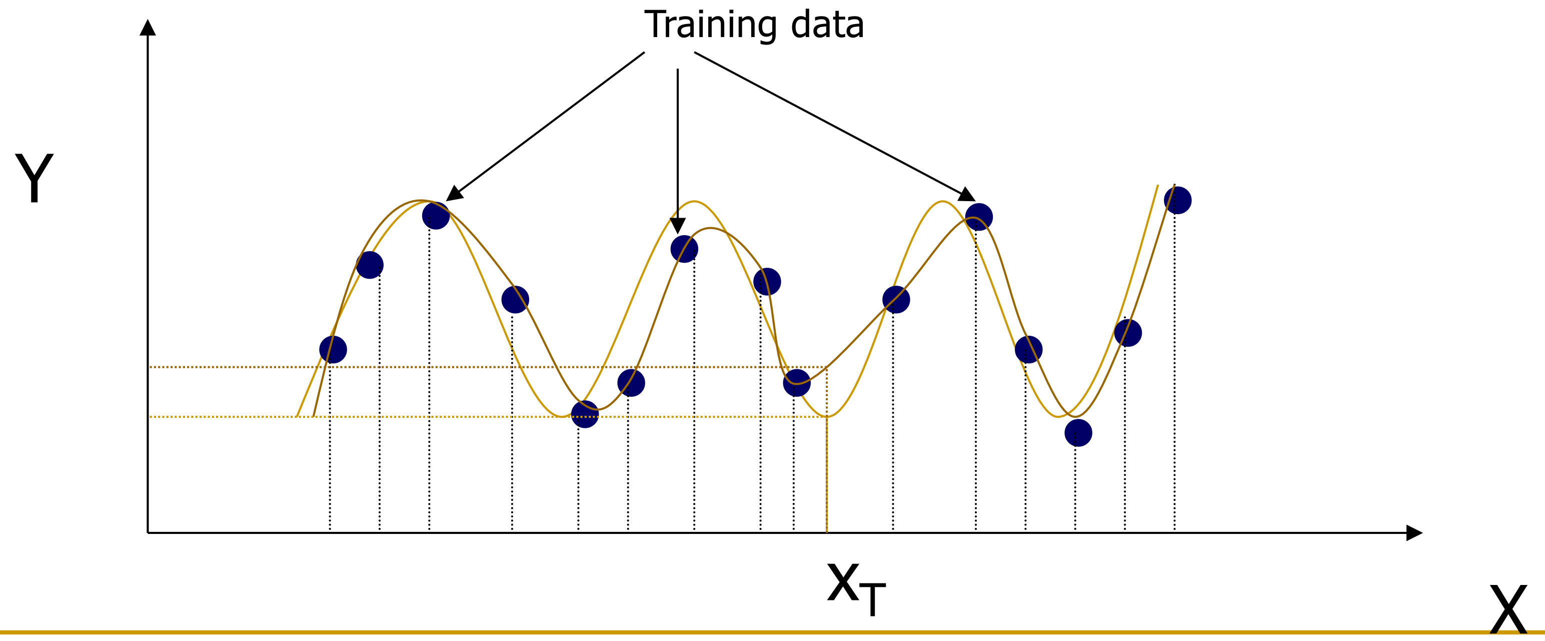


Sepal length	Sepal width	Petal length	Petal width	Class
5.1	3.5	1.4	0.2	0
4.9	3.0	1.4	0.2	2
4.7	3.2	1.3	0.2	0
4.6	3.1	1.5	0.2	1

- Function approx Vs Grading

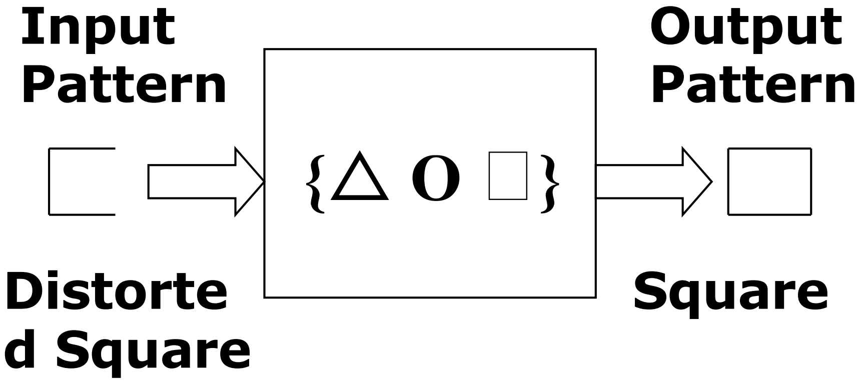
Generalization

- **The network is said to generalized well when it sensibly interpolates input patterns that are new to the network**



Neuro Processing

- Association response–auto association–hetero association



Input Pattern

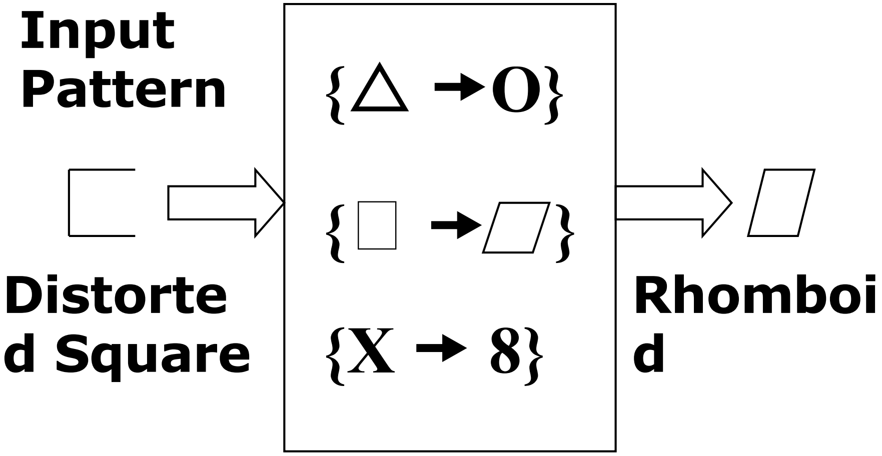
Distorted Square

{△ O □}

Output Pattern

Square

Auto association



Input Pattern

Distorted Square

{△ →O}

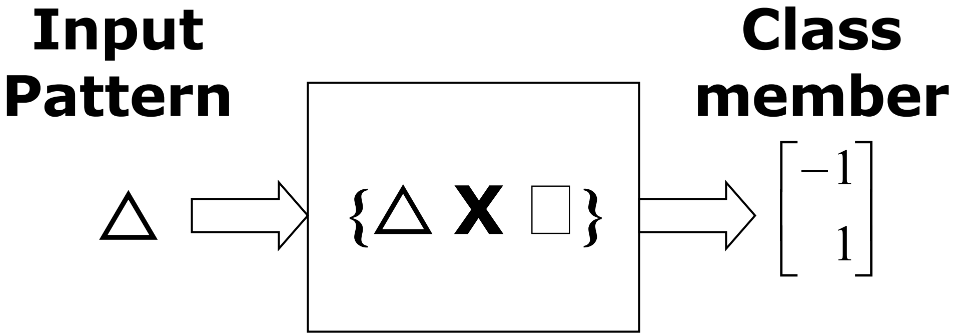
{□ →▱}

{X → 8}

Rhomboid

Hetero association

- Classification response – classification - recognition



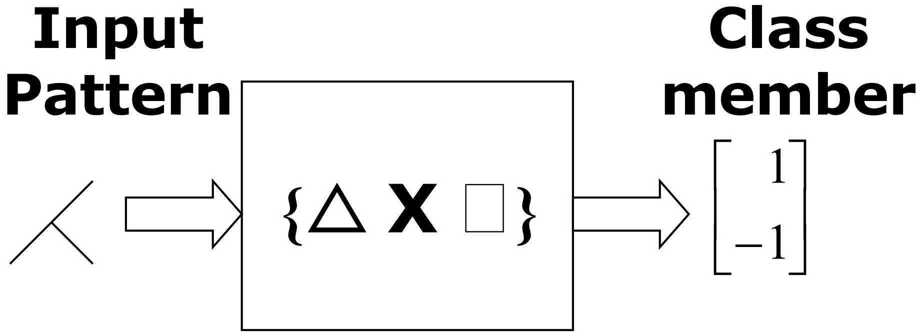
Input Pattern

{△ X □}

Class member

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

classification



Input Pattern

{△ X □}

Class member

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

recognition

Learning in Neural Networks

- Learn values of weights from I/O pairs
- Start with random weights
- Load training example's input
- Observe computed input
- Modify weights to reduce difference
- Iterate over all training examples
- Terminate when weights stop changing OR when error is very small

Learning Algorithm

Epoch : Presentation of the entire training set to the neural network.

In the case of the AND function an epoch consists of four sets of inputs being presented to the network (i.e. [0,0], [0,1], [1,0], [1,1])

Error: The error value is the amount by which the value output by the network differs from the target value. For example, if we required the network to output 0 and it output a 1, then Error = -1

Learning Algorithm

Target Value, T : When we are training a network we not only present it with the input but also with a value that we require the network to produce. For example, if we present the network with [1,1] for the AND function the target value will be 1

Output , O : The output value from the neuron

Ij : Inputs being presented to the neuron

Wj : Weight from input neuron (I_j) to the output neuron

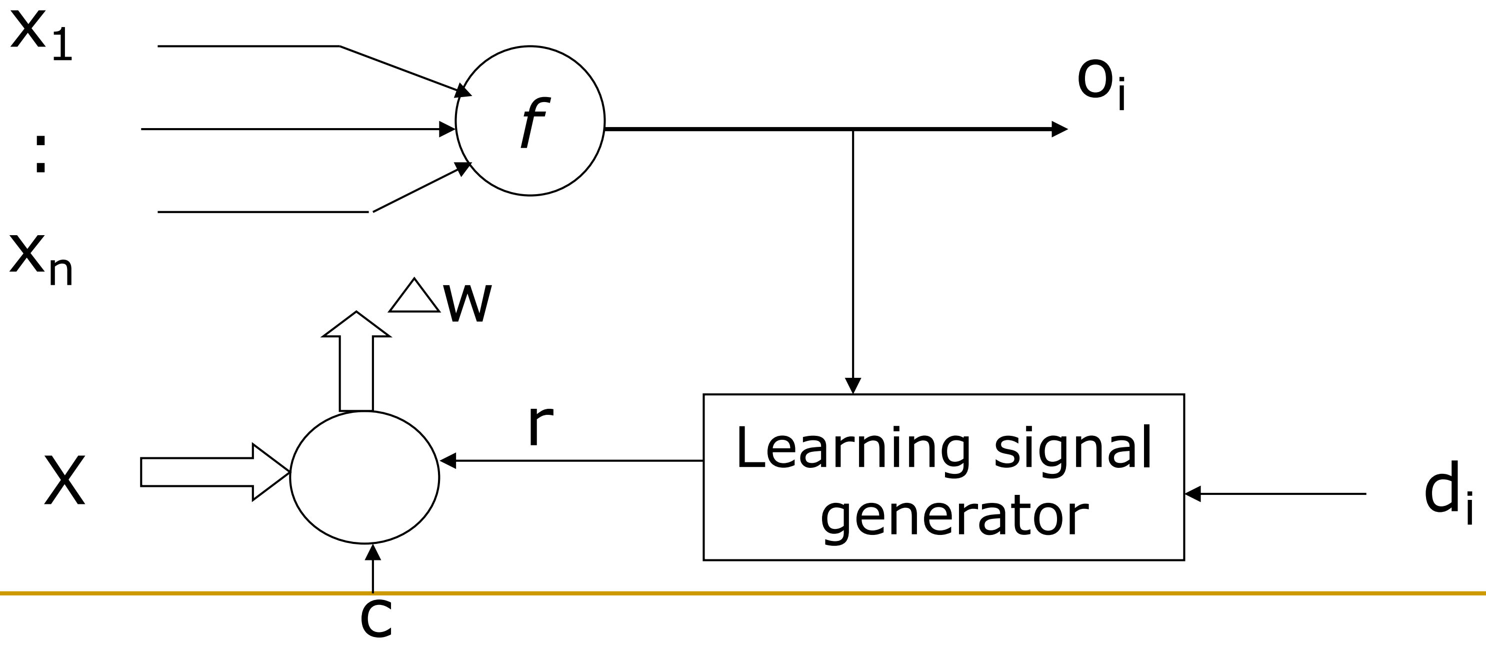
LR : The learning rate. This dictates how quickly the network converges. It is set by a matter of experimentation. It is typically 0.1

Learning Rules

- **General Rule: Weight vector increases in proportion to the product of input X and learning signal r,**

where r = r(w_i,X,d_i) so

change in weight = [cr(w_i,X,d_i)] X, C learning constant



1. Hebbian learning rule

- **In this rule learning signal r is,**

$$r = f(W_i^T X)$$

$$\Longrightarrow \Delta W_i = cf(W_i^T X)X$$

-Unsupervised learning
-Initial weights random, around zero

$$\Longrightarrow \Delta w_{ij} = cf(W_i^T X)x_j$$ **i=1,..m**

$$\Longrightarrow \Delta w_{ij} = co_i x_j$$ **J= 1,..n**

Perceptron learning rule

- In this rule learning signal r, is

-Supervised learning

$$r = d_i - o_i$$

-Weights can be any value initially

$$\Delta W_i = c[d_i - \mathrm{sgn}(W_i^T X)]X$$

for i=1,..,m

$$\Delta w_{ij} = c[d_i - \mathrm{sgn}(W_i^T X)]x_j$$

for j=1,..,n

Delta learning rule

- In this rule learning signal r is,

$$r = [d_i - f(W_i^T X)] f^{'}(W_i^T X)$$

$f^{'}(W_i^T X)$ **-Derivative of activation function**

- Weights can be initialized any any values randomly
- Can be generalized for multilayer networks
- Supervised training rule
- Only valid for continuous activation function
- Derived from least square error between o_i and d_i

Other learning rules

- Widrow Hoff learning rule – f(x)=x – special case of delta learning rule
- LMS (least mean square) learning rule

$$\Delta w_{ij} = c(d_i - W_i^T X)x_j$$

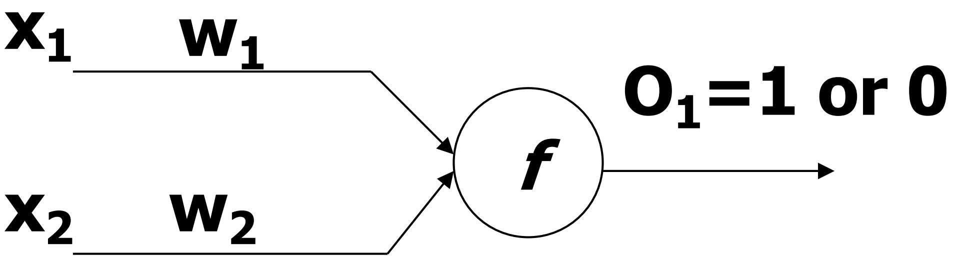
- Correlation learning rule $r_i = d_i$
- Winner all take learning rule
- Here, neuron with maximum response due to X is found.
- Weight are modified to match that X.
- i.e.

$$\Delta w_{mj} = \alpha(x_j - w_{mj})$$

$$\Delta w_{ij} = cd_i x_j$$

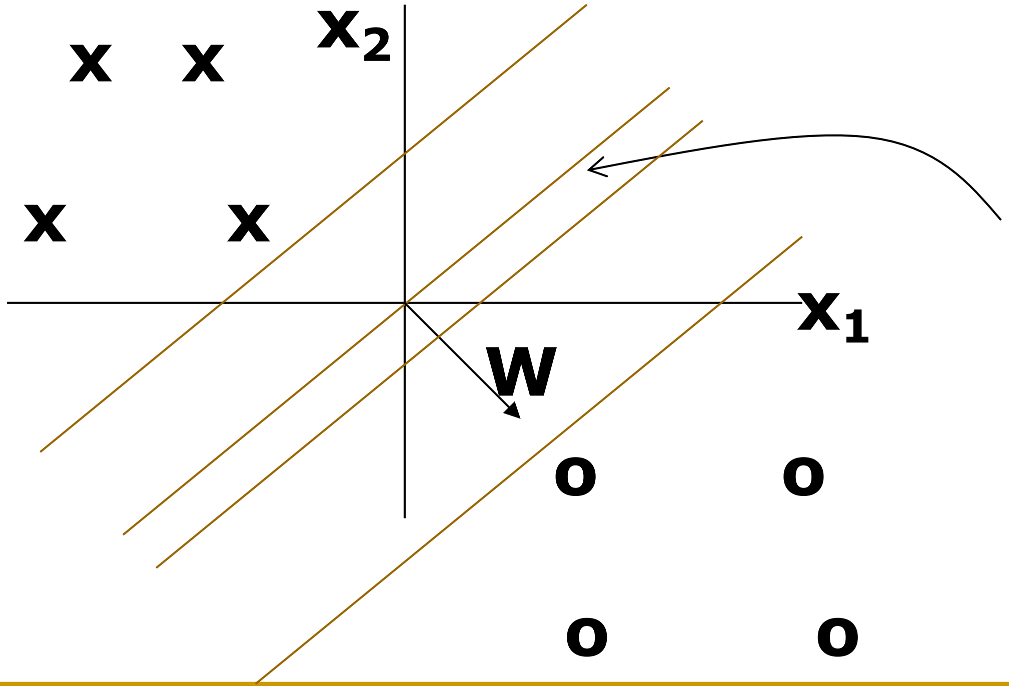
Single perceptron as classifier

- What is net doing while learning?
- Consider two input one output network



$w_1x_1+w_2x_2>0$ class 1 : $o_1=1$

$w_1x_1+w_2x_2<0$ class 2 : $o_1=0$



$w_1x_1+w_2x_2=0$

$WX^T=0$

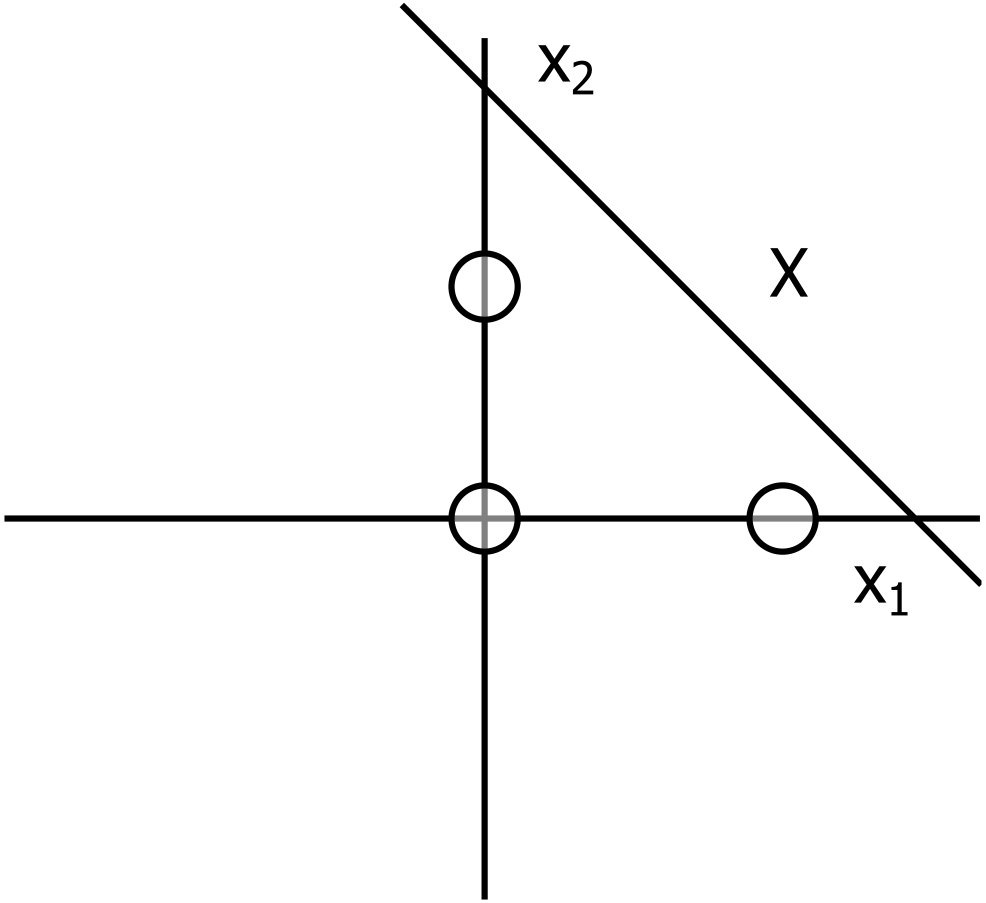
W is perpendicular to X

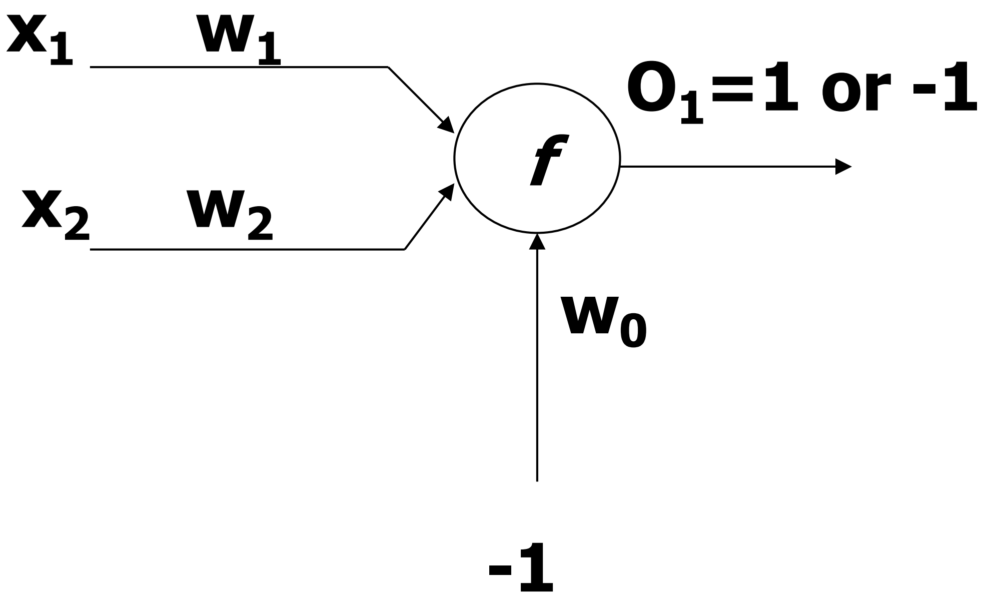
- In practical application line may not pass through origin
- Bias is used to achieve this
- Bias also needs to train
- Conclusion: Classes must be linearly separable

Example

- **Construct ANN to solve AND gate**

P	X_1	X_2	D
1	0	0	-1
2	0	1	-1
3	1	0	-1
4	1	1	+1





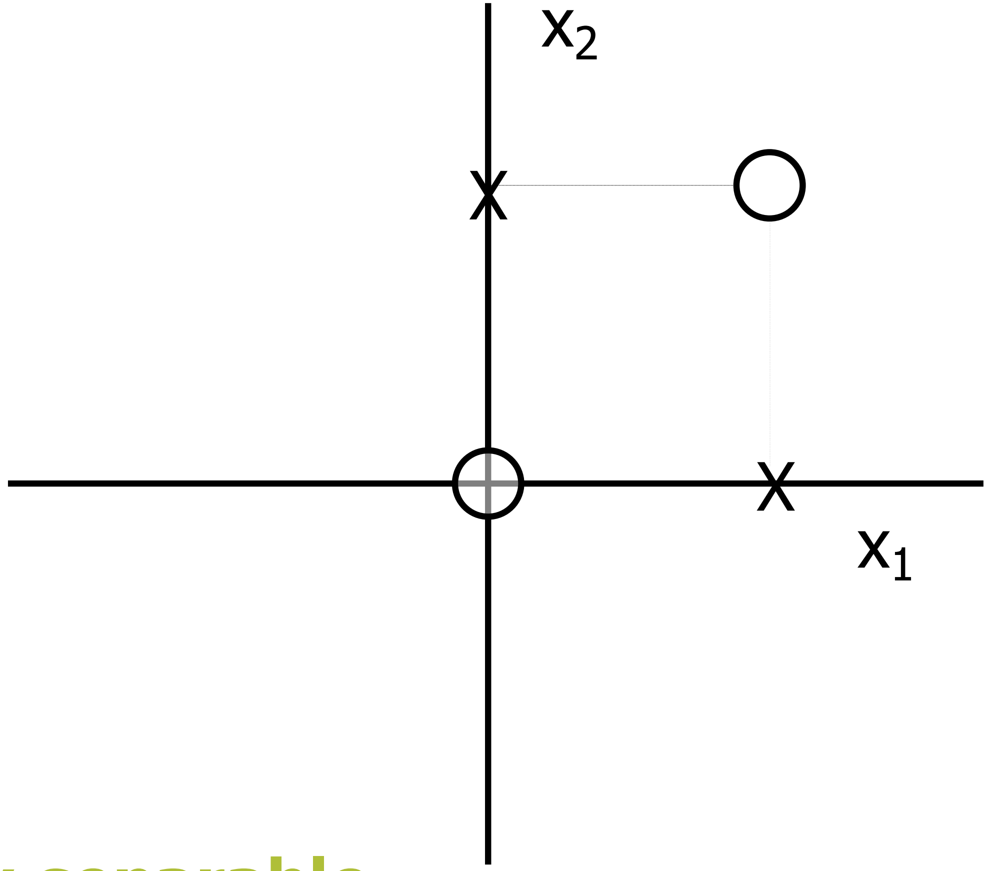
Boundary

$$w_1x_1+w_2x_2-w_0=0$$

world is not that simple…

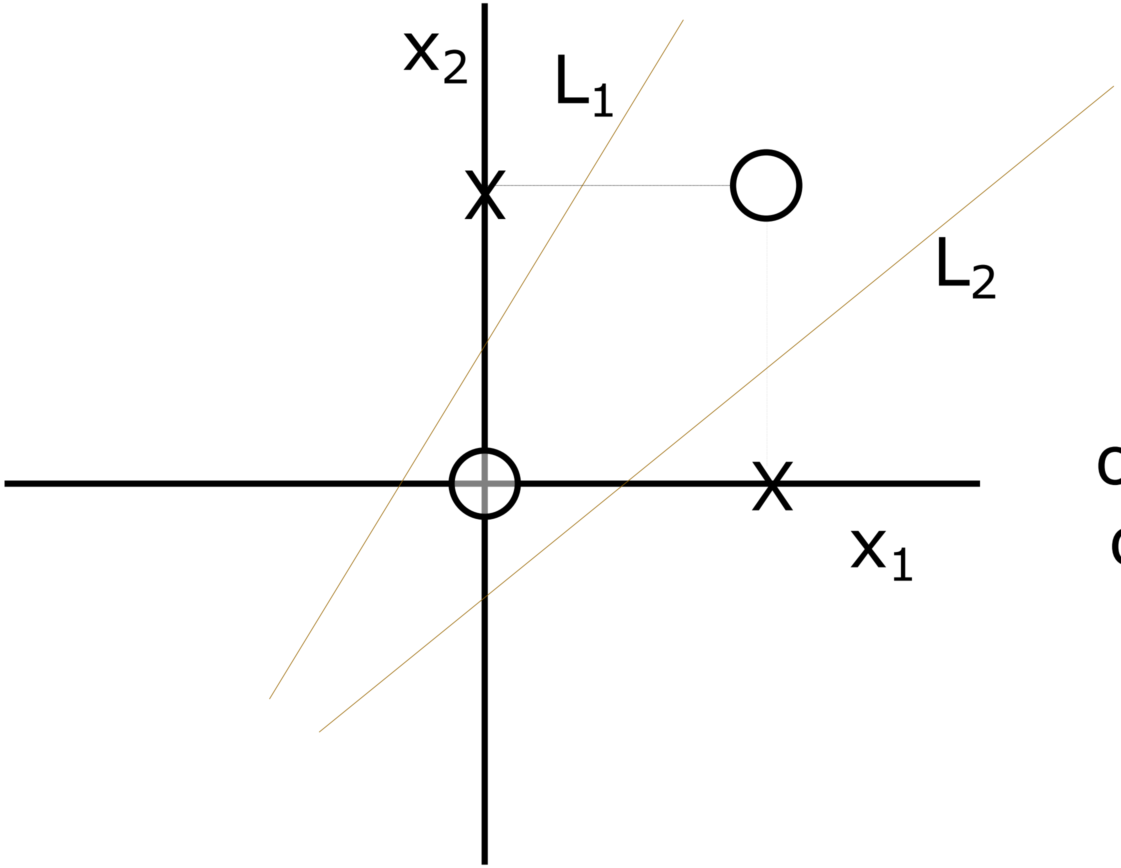
- **Ex-OR gate**

P	X_1	X_2	D
1	0	0	-1
2	0	1	+1
3	1	0	+1
4	1	1	-1



Patterns are not linearly separable

Hidden transformation



L1: -2x1+x2-1/2=0
L2:   x1-x2-1/2=0

o_1=sgn(-2x1+x2-1/2)
o_2=sgn(x1-x2-1/2)

Image space

Pattern Space		Image Space		Class
x_1	x_2	o_1	o_2	-
0	0	-1	-1	2
0	1	1	-1	1
1	0	-1	1	1
1	1	-1	-1	2

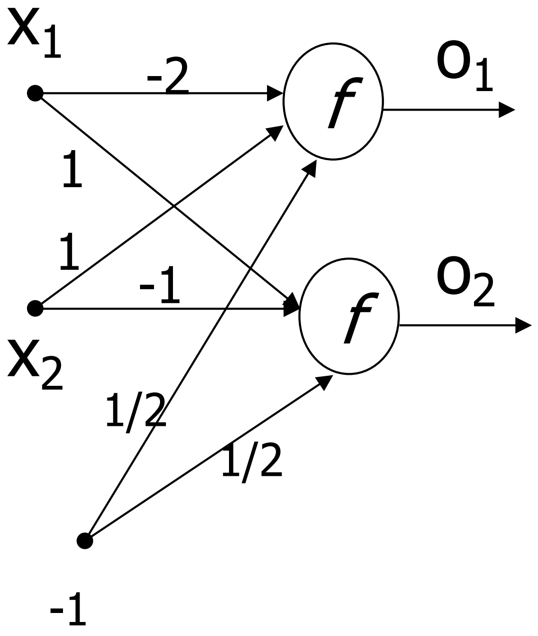
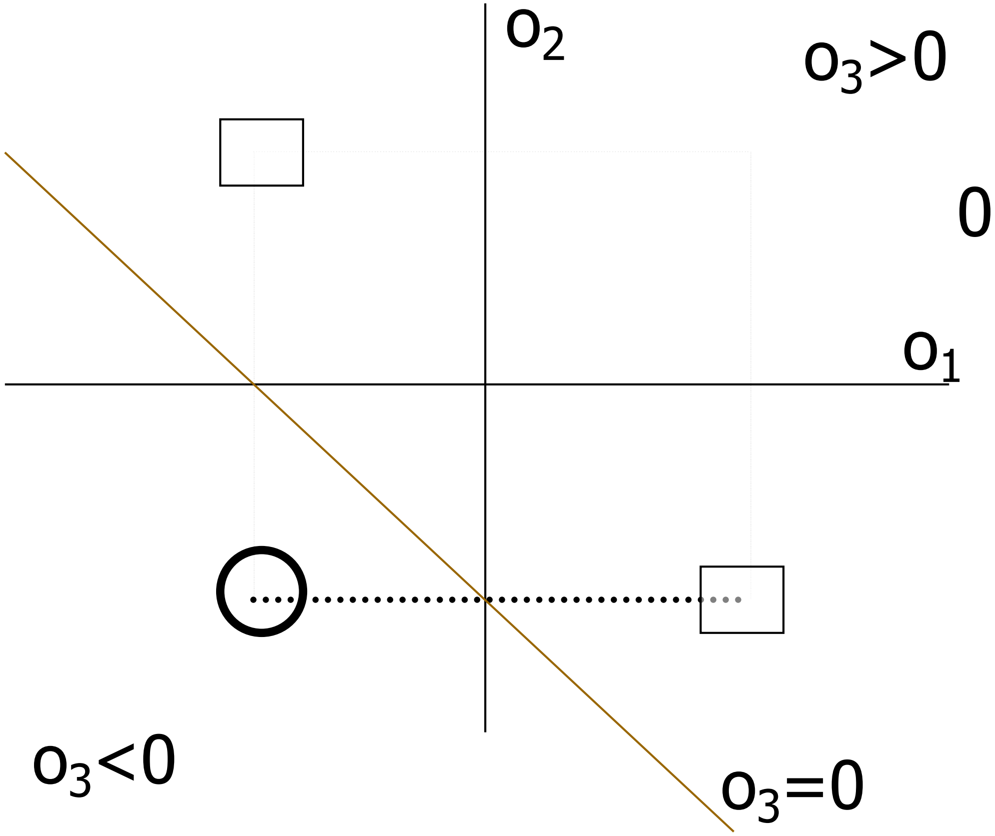
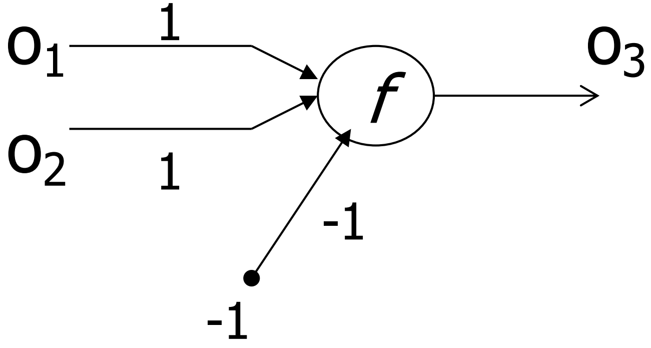


Image Space



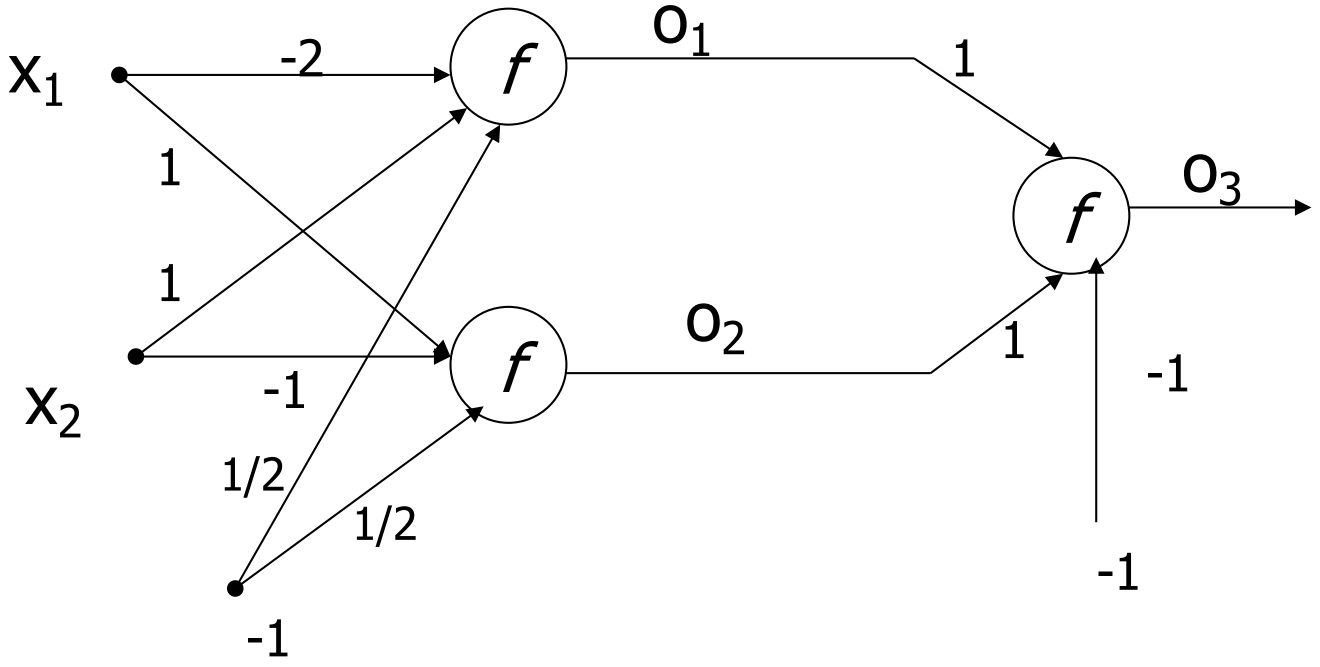
$0_3=\mathrm{sgn}(o_1+o_2+1)$



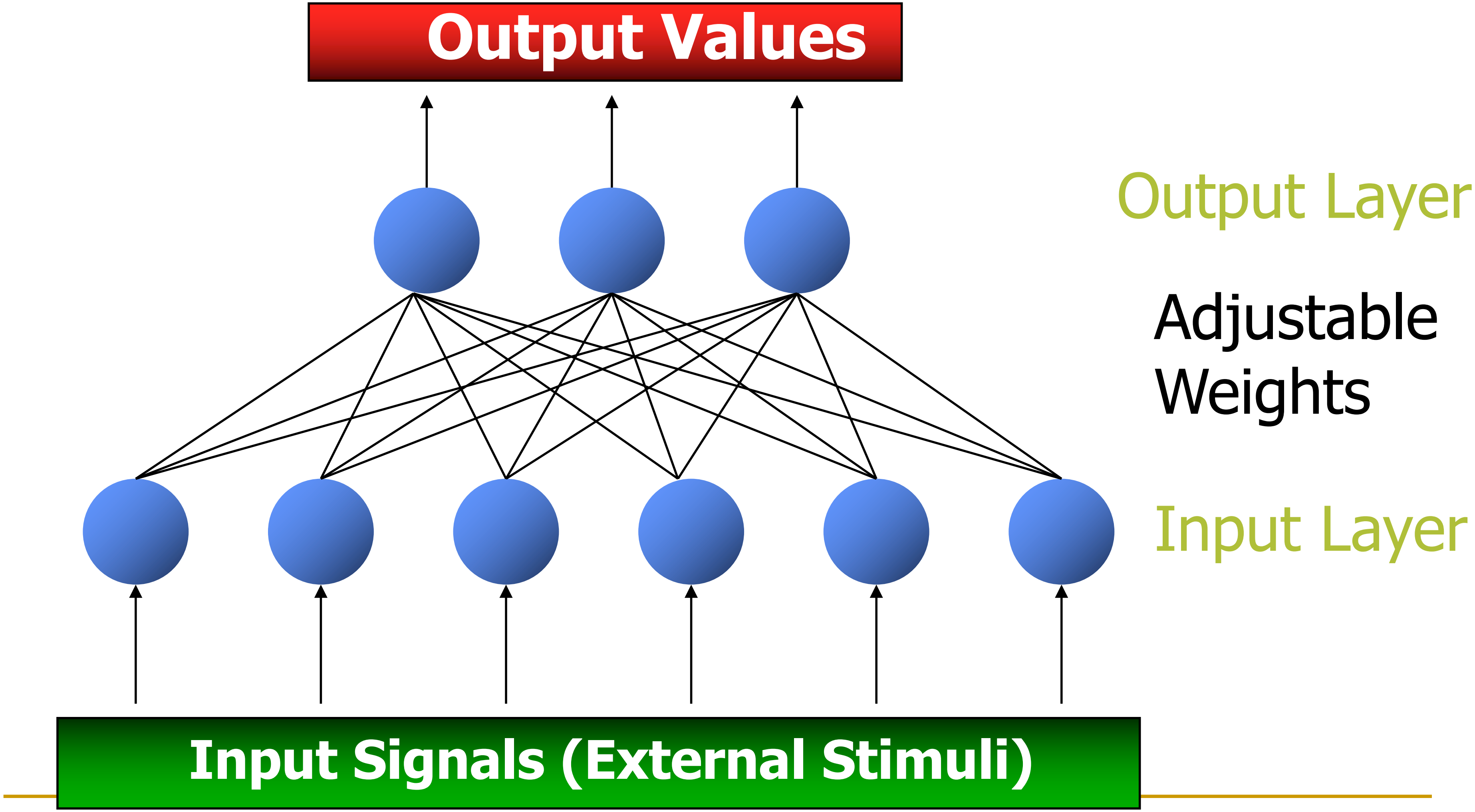
Finally…

Pattern Space		Image Space		o_1+o_2+1	0_3	Class
x1	x2	o_1	o_2	-	-	-
0	0	-1	-1	-ve	-1	2
0	1	1	-1	+ve	+1	1
1	0	-1	1	+ve	+1	1
1	1	-1	-1	-ve	-1	2

Two Layer Network



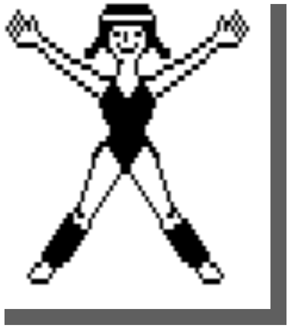
Multilayer Perceptron (MLP)



Over-fitting

- With sufficient nodes can classify any training set exactly
- May have poor generalisation ability
- Cross-validation with some patterns
    - Typically 30% of training patterns
    - Validation set error is checked each epoch
    - Stop training if validation error goes up

Training time



- How many epochs of training?
  - Stop if the error fails to improve (has reached a minimum)
  - Stop if the rate of improvement drops below a certain level
  - Stop if the error reaches an acceptable level
  - Stop when a certain number of epochs have passed

Design Issues

- Training errors can be taken cumulative (sum of all pattern presented in one batch).

$$E = 0.5\sum_{p=1}^{P}\sum_{k=1}^{K}(d_{pk} - o_{pk})^2$$

OR

$$E_{rms} = \frac{1}{PK}\sqrt{\sum_{p=1}^{P}\sum_{k=1}^{K}(d_{pk} - o_{pk})^2}$$

- Learning constant.
- Weight initialization.
- Steepness of activation function.
- Momentum method: Increase the speed of convergence by taking in to account last change of weights.
- Number of hidden layer neurons:
- Network Pruning.
- Different activation functions(e.g Radially symmetric function): Radial Basis function networks – f(x)=sinx/x.

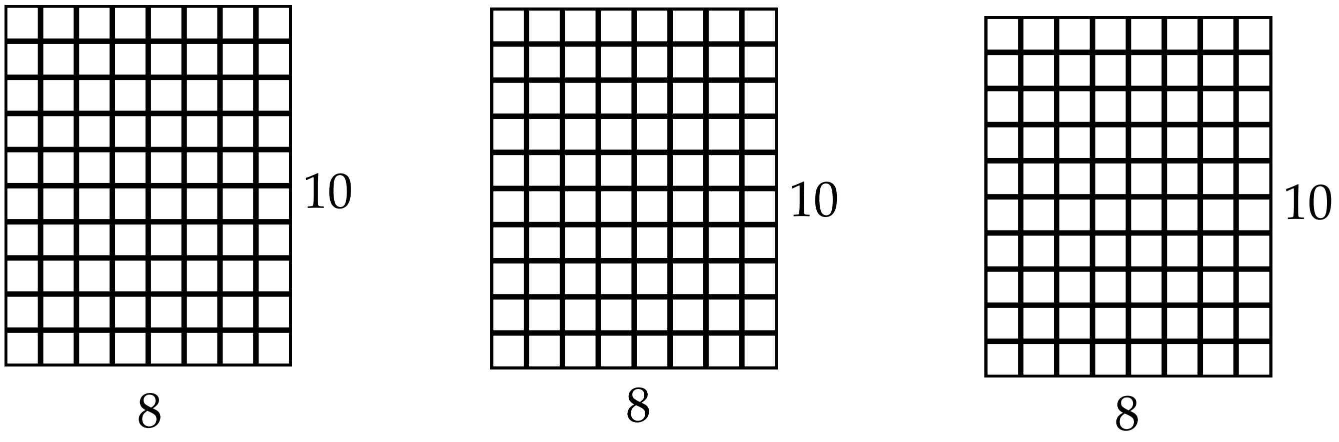
$$\sqrt{I * K}$$

- Perceptron convergence theorem.

Recurrent Networks

- **Have feedback connections while training.**
- **No feedback while recall.**
- **Fundamental problem this network solves is: Associative memory problem: store p patterns X in such a way that when presented with a new pattern y the network responds by producing whichever of the stored pattern closely matches y.**
- **Can be very well used for solving TSP.**

OCR for 8x10 characters



- NN are able to generalise
- learning involves generating a partitioning of the input space
- for single layer network input space must be linearly separable
- what is the dimension of this input space?
- how many points in the input space?
- this network is binary(uses binary values)
- networks may also be continuous

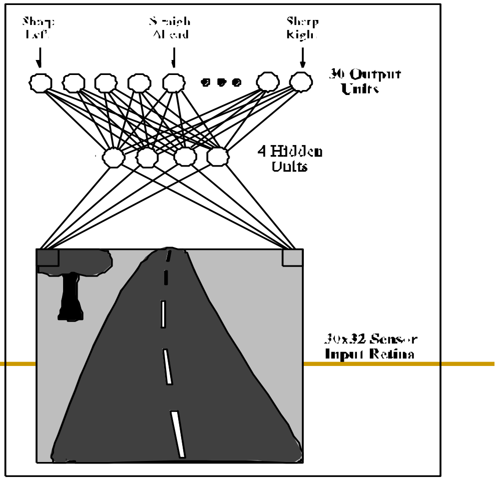
ALVINN

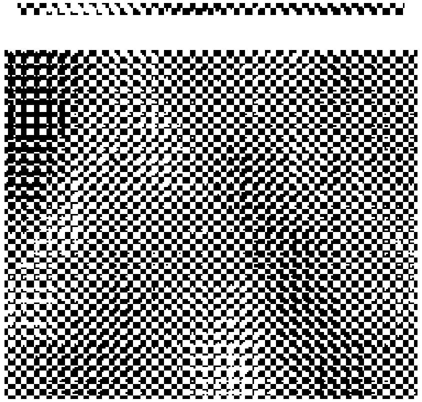
Drives 70 mph on a public highway

30 outputs for steering

4 hidden units

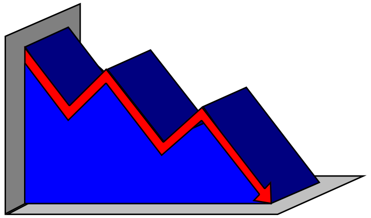
30x32 pixels as inputs



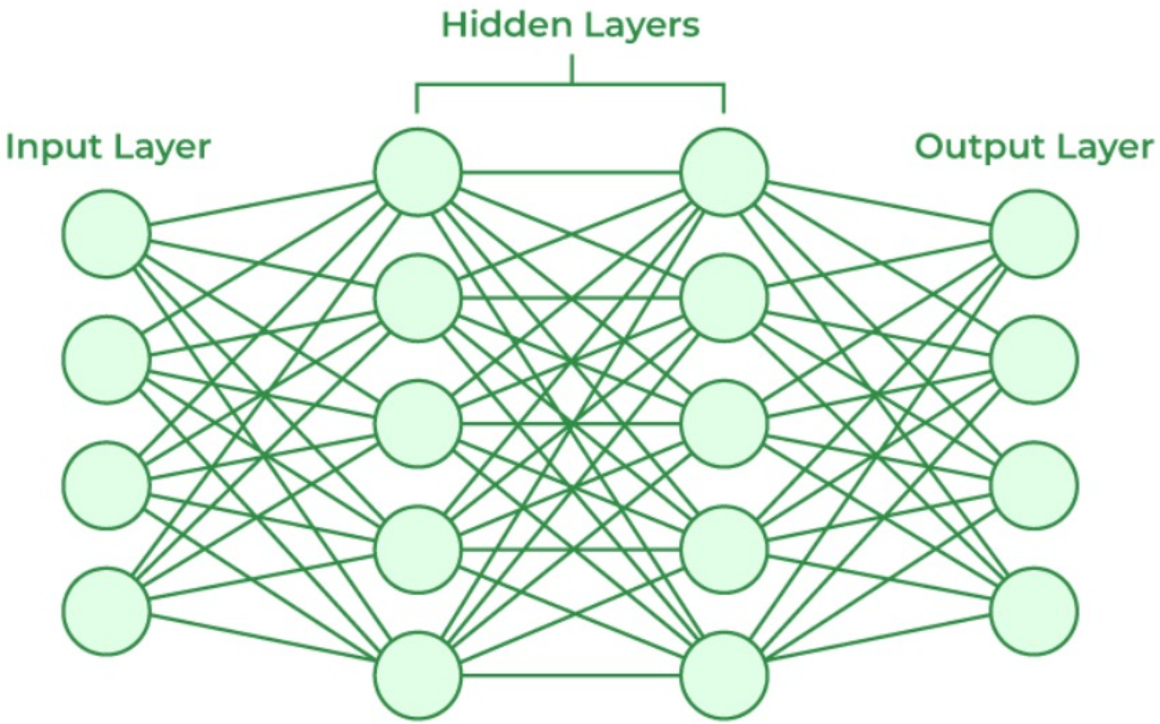


30x32 weights into one out of four hidden unit

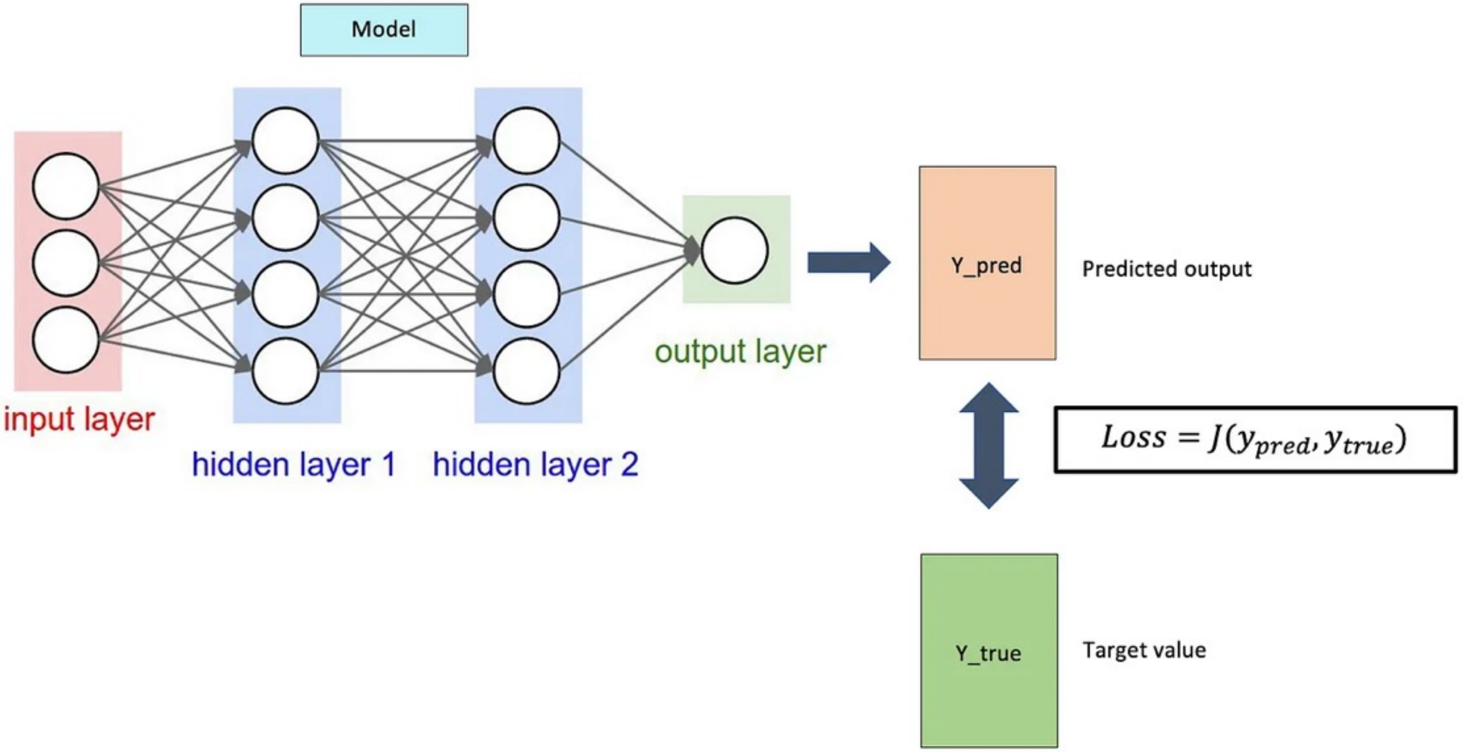
Stock market prediction



- **"Technical trading" refers to trading based solely on known statistical parameters; e.g. previous price**
- **Neural networks have been used to attempt to predict changes in prices**
- **Difficult to assess success since companies using these techniques are reluctant to disclose information**



Neural Networks Architecture


Model
input layer
hidden layer 1
hidden layer 2
output layer
Y_pred
Predicted output
$Loss = J(y_{pred}, y_{true})$
Y_true
Target value

Best References

- Introduction to Artificial Neural Networks: Jacek M Zurada
- Introduction to Artificial Neural Networks : S Haykin
- Matlab ANN Toolbox
- Studgard university ANN simulator