Lecture 7:

Artificial Neural Networks

Week of February 6, 2023

University California, Berkeley

Machine Learning Algorithms

MSSE 277B, 3 Units

Spring 2023

Prof. Teresa Head-Gordon

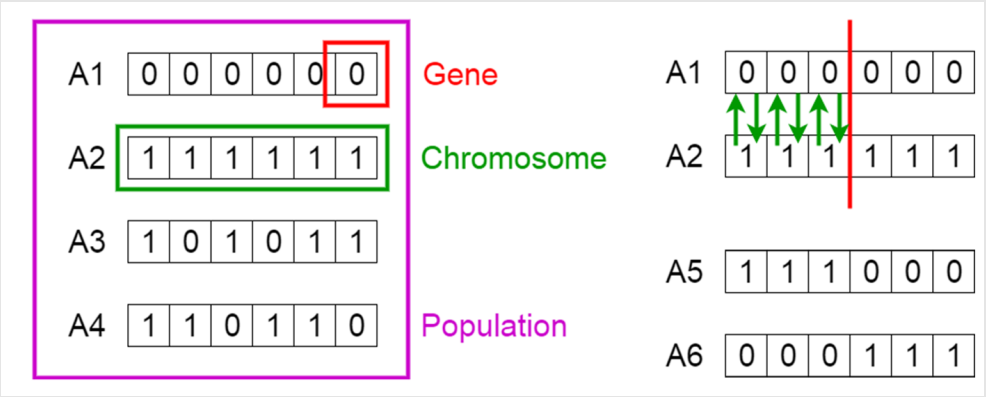
Departments of Chemistry, Bioengineering, Chemical and Biomolecular Engineering

Review Previous Lecture: Genetic Algorithms

GA: a biology inspired meta-heuristic optimization algorithm based on the principles of natural selection

(1) Encoding function: In optimization applications, how to best encode your problem: variables and objective function values

(2) Population undergoes changes by operations of reproduction, cross-over, mutation, maybe others



(3) Interbreeding population

(4) Survival of the fittest: as measured by some objective function; scoring function; free energy or energy function

with proper encoding and low mutation rates, the final population will be dominated by the fittest members.

Today's Lecture: No Free Lunch Theorem

Lecture Purpose: Many existing metaheuristic approaches to optimization are quite successful, and GA is a prime example! But according to the "no free lunch" theorem, all metaheuristics which search for extrema are exactly the same in performance, statistically, when averaged over all possible cost functions and applications.

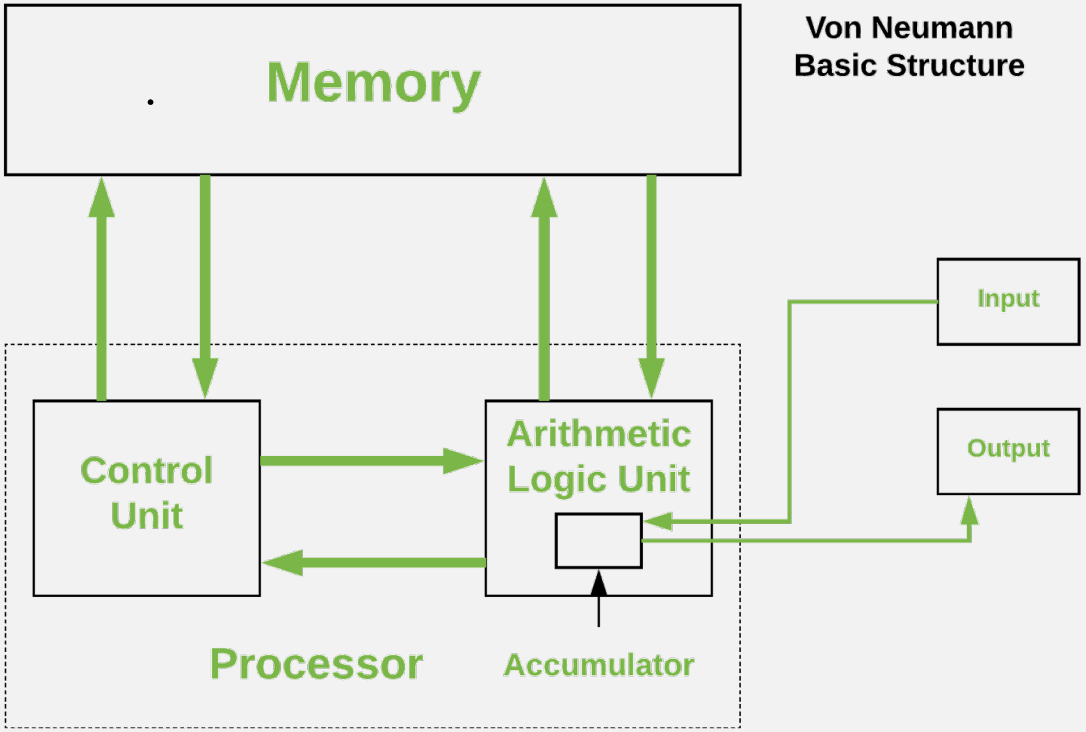
Since it is theoretically impossible to have a best universal optimization strategy, the only way for one strategy to be superior to the others is when we focus on a particular class of problem. With prior knowledge of the problem under consideration, one metaheuristic can be modified and, thus, may become more suited to the problem relative to the others.

An important goal in this class is getting a good sense of what are the "massageable" approaches of a given ML method applied to a given chemical/physical problem.

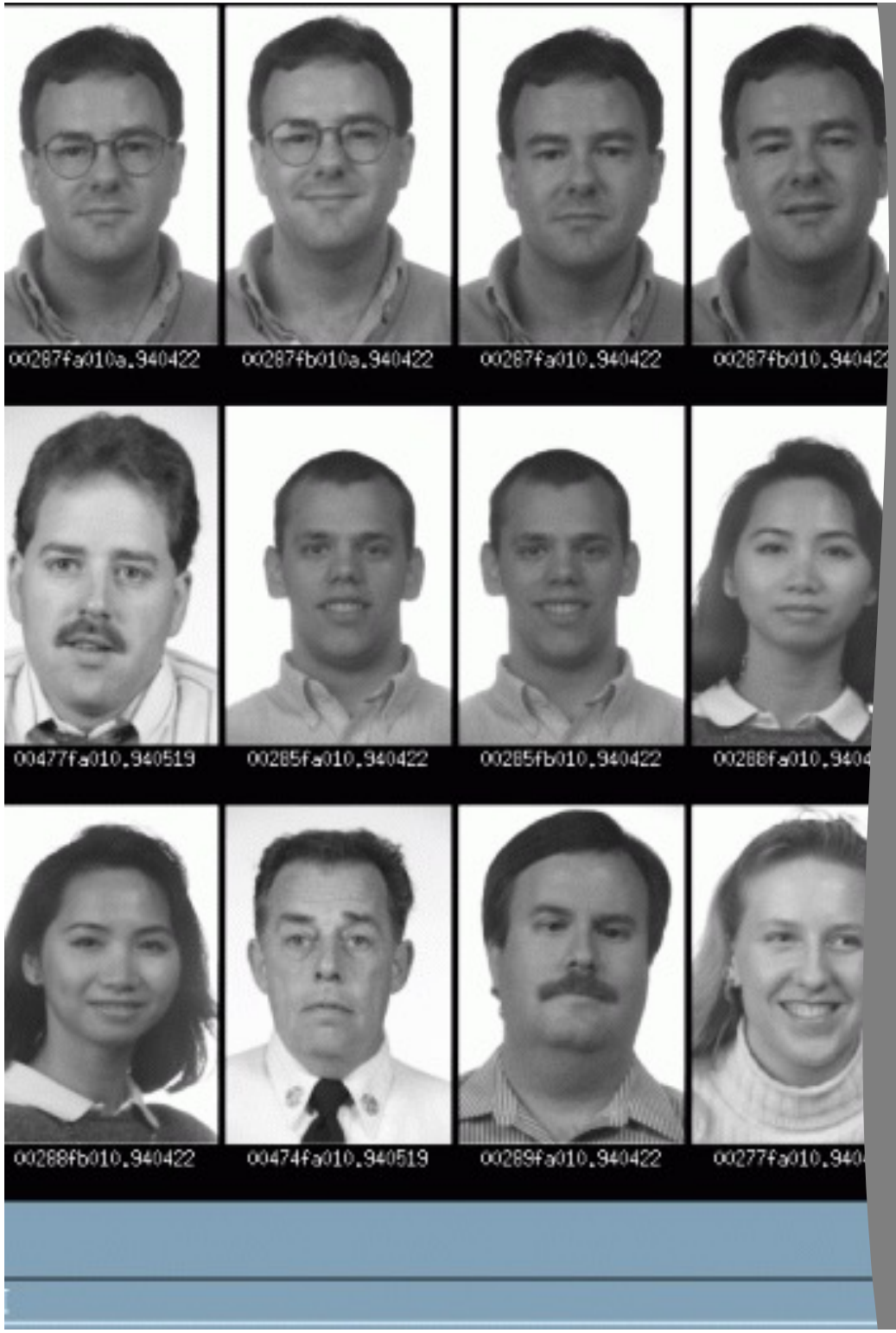
Today we focus on artificial neural networks as a final biology-inspired metaheuristic algorithm for finding "better" and/or (hopefully) best solution for classification and regression problems.

Metaheuristic Algorithms for Non-algorithmic Computing

Most of this semester we have focused on algorithms and methods that can be solved on a von Neumann machine; i.e. we process symbols based on an underlying formal model manifested as a computer program



- However, there are many problems that we can't formalize through symbols of a mathematical model that is then solved on a computer – i.e. such as a potential energy cost function.
- Instead we have to learn the relationship or the mapping between some input space and an inferred outcome.



Introduction to Neural Networks

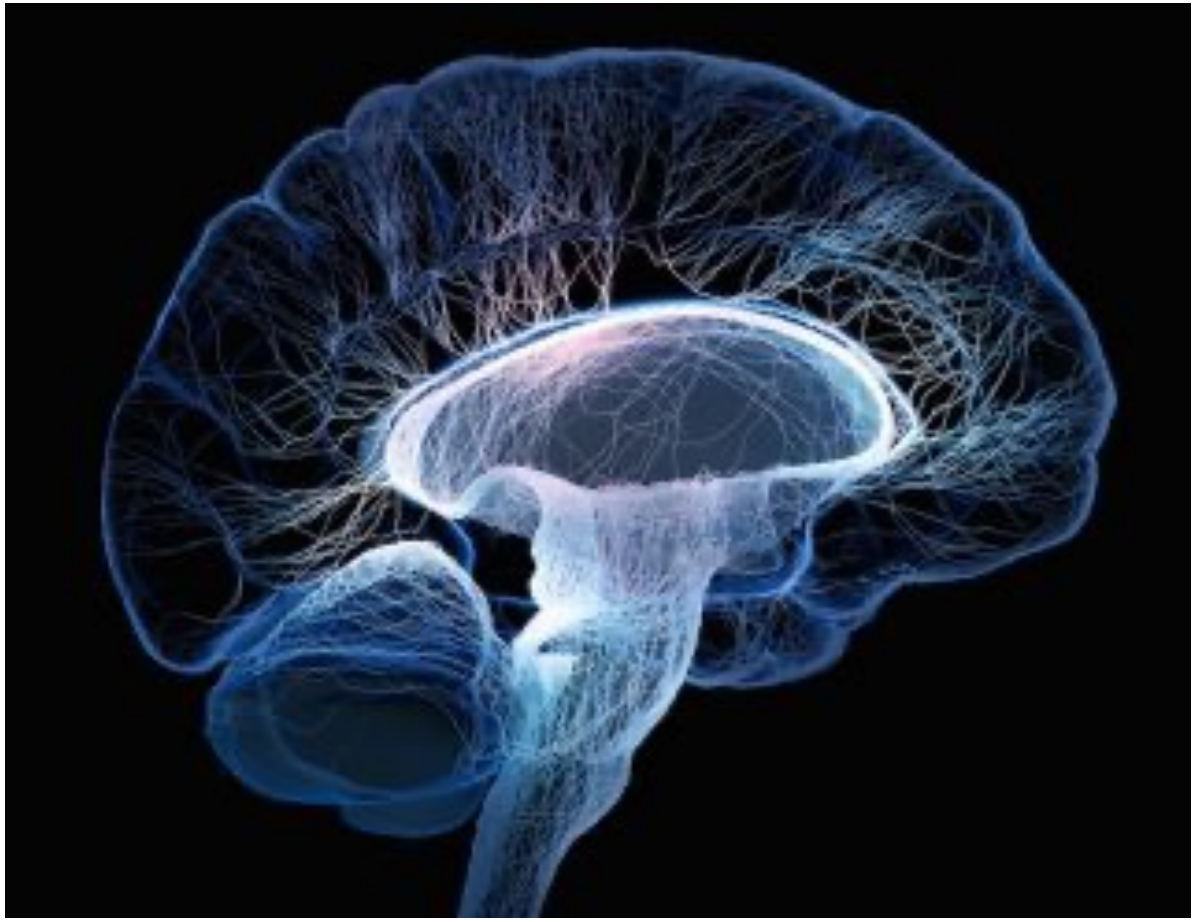
One good example, is pattern recognition. For example, how does an infant recognize faces of Mom/Dad from everyone else.

- Algorithmically we can define eyes, nose, hairline and a formal mathematical model that describes their spatial arrangement with respect to each other

- But that would likely not be precise enough to pick out Mom and Dad among these folks.

However, as we know, the human brain is quite good at figuring out that problem, and does it quite speedily!

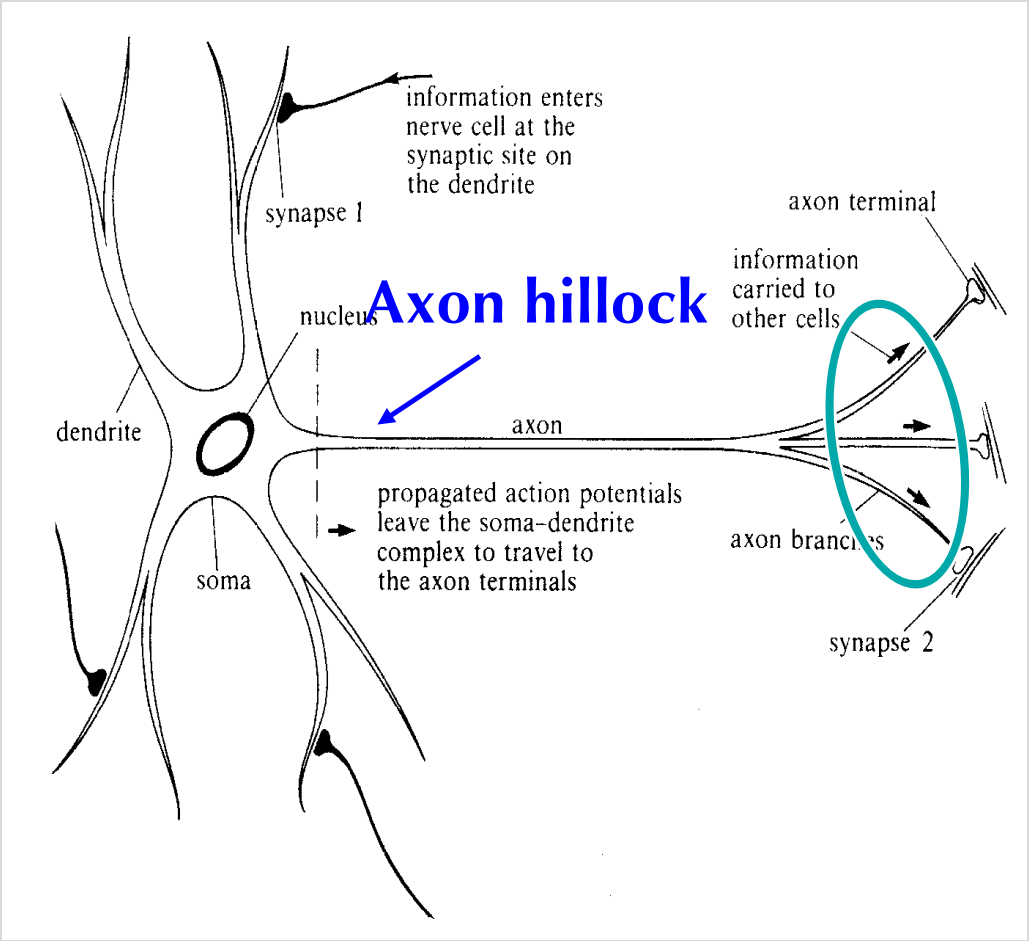
Introduction to Neural Networks



Artificial neural networks (ANN) seeks to understand how the brain processes information. By analogy, it is used in optimization and classification in the sciences.

- The human brain contains ~10 billion neurons, and each neuron is connected to other neurons through about ~10^4 synapses.
- We can view brain architecture as a massively parallel computer that can perform parallel computations extremely efficiently.
- For example, complex visual perception occurs within ~10^{-1}s. It is assumed that each neuron processor speed is ~10^{-2}s, so that is 10 processing steps!

http://www.willamette.edu/~gorr/classes/cs449/brain.html



Biological Neuron

(1) Incoming action potential travels along axon to synapse

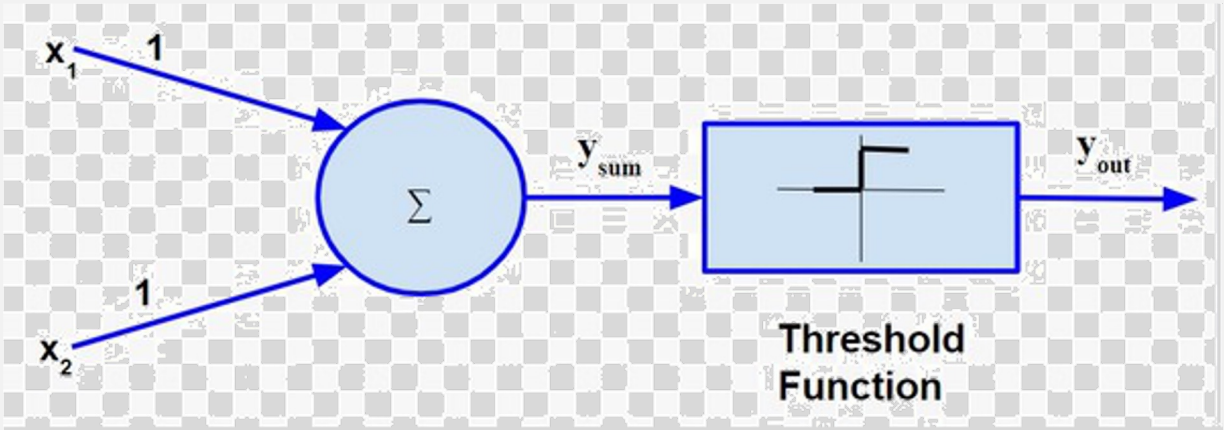
(2) Neurotransmitter released at synapse and binds to membrane of neuronal cell

(3) Cell is induced into a state of hyperpolarized or depolarized by the incoming signals

(4) All such signals are integrated at the axon hillock

(5) If integrated signal reaches a certain threshold, then neuron fires and action potential is sent along outgoing axon to other neurons

Artificial Neuron: McCulloch-Pitts Neuron



Threshold Logic Unit

$$y = \begin{cases} 1 & \Sigma \geq 0 \\ 0 & \Sigma < 0 \end{cases}$$

We have various possibilities:

$$y = \frac{1}{1+\exp(-\beta a)}$$

$$y = \tanh(\beta a)$$

$$y = \sum_{i=1}^{L} x_i - \theta \rightarrow fire \,/\, no\ fire$$

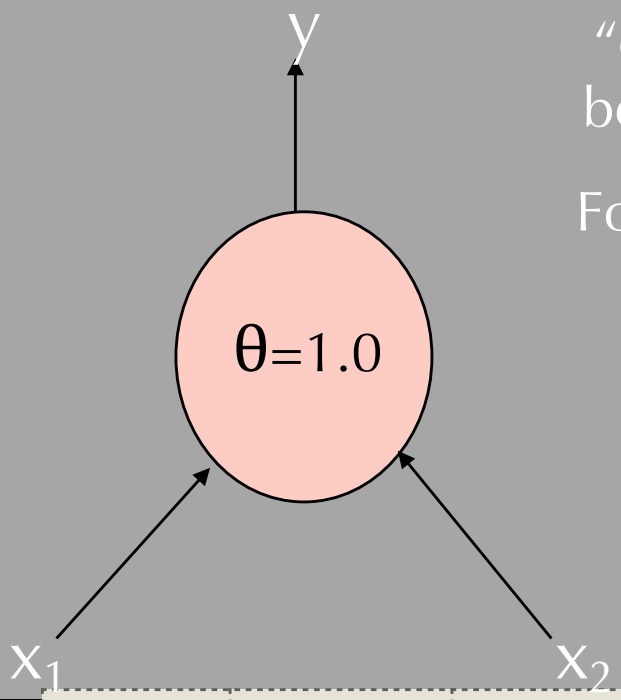
(1) {x} correspond to incoming action potentials from other axons

(2) Σ is the integrating unit corresponding to axon hillock

(3) If integrated signal reaches a threshold, $\boldsymbol{\theta}$, then it fires to give y. Otherwise no fire

The McCulloch-Pitts Neuron is a neuronal computing element as it can be used to classify two outcomes: on or off.

Simple "OR" Function



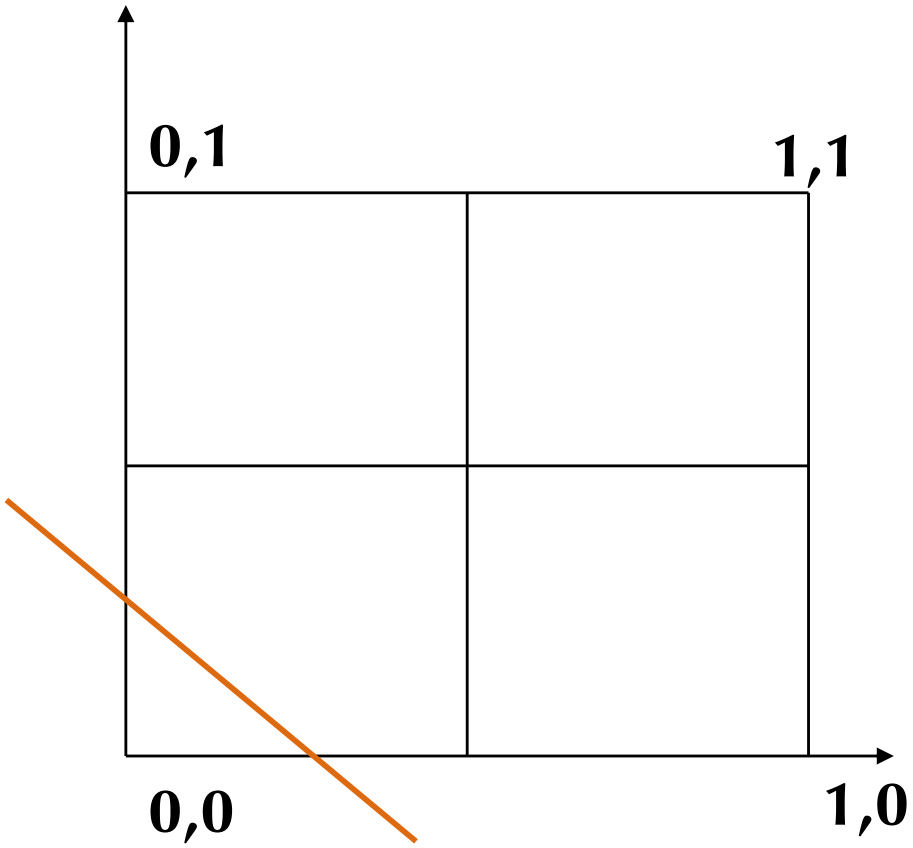
Lets consider some Boolean operations such as simple "or" defined by truth table below: an output action will become TRUE if either one "OR" more events are TRUE

For a McCulloch-Pitts neuron we just need to optimize a threshold θ to determine the truth table

$$y = 1 \quad if \sum_{i=1}^{L} x_i - \theta \geq 0$$

$$y = 0 \quad if \sum_{i=1}^{L} x_i - \theta < 0$$

x_1	x_2	OR outcome	$\sum_{i=1}^{L} x_i - \theta$	y=1 if ≥ 0
0	0	No fire	0+0-1	0
0	1	Fire	0+1-1	1
1	0	Fire	1+0-1	1
1	1	Fire	1+1-1	1



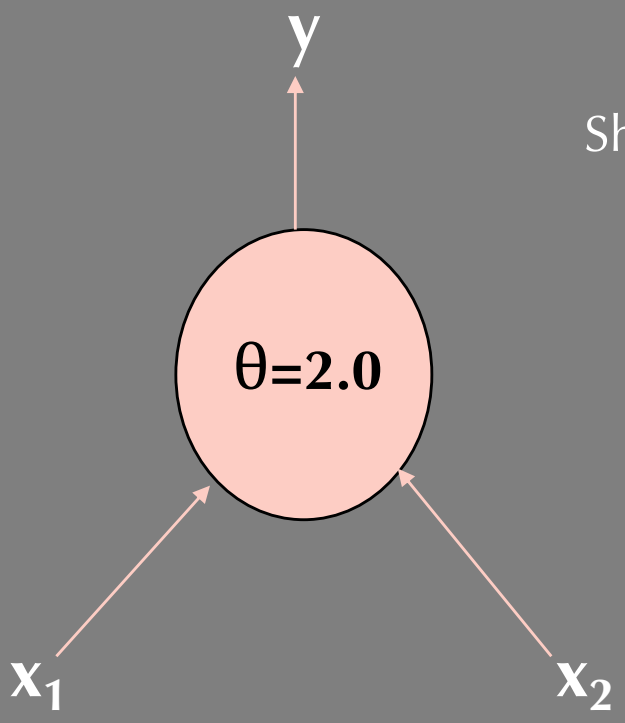
Geometric Interpretation of Neuronal Computing

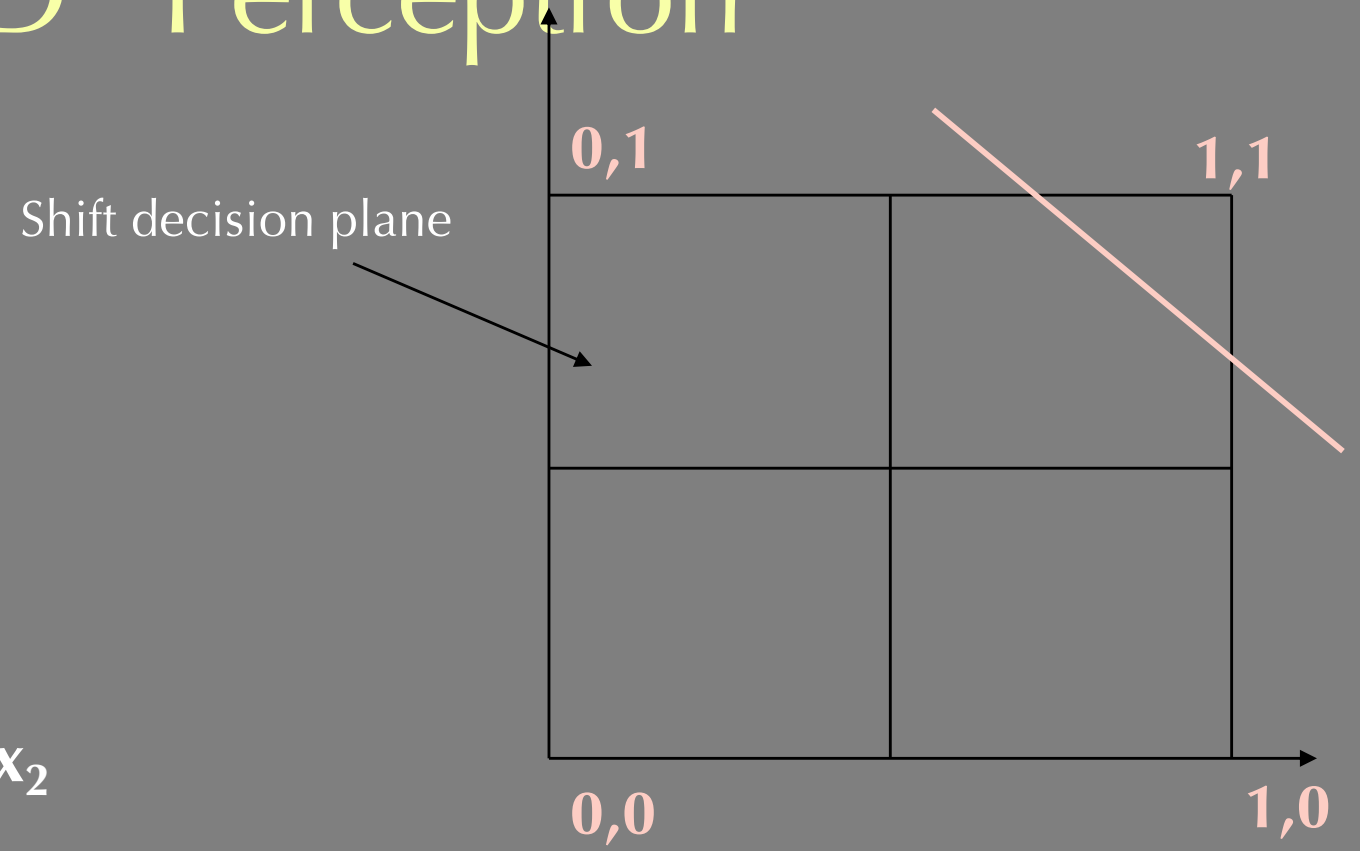
Can define a decision plane where solution space can be divided or classified into on or off. (Typically it is a decision surface in N-dimensional space)

The red line is the decision plane we determined for the simple OR function

This is controlled through the threshold variable θ and definitions of our input encoding: many solutions possible

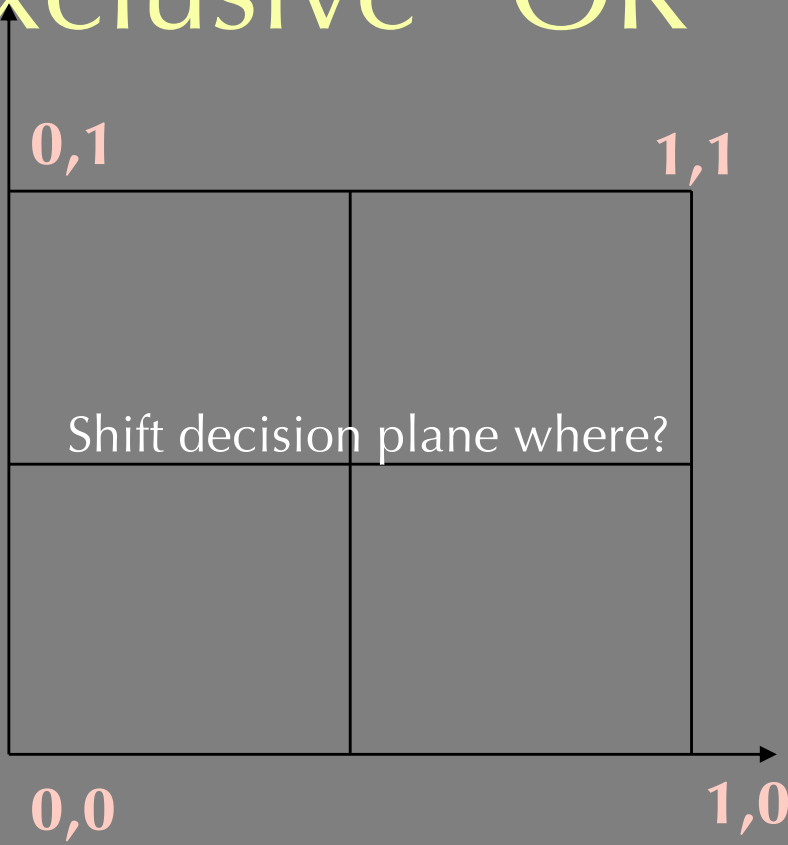
Simple "AND" Perceptron





x_1	x_2	AND outcome	$\sum_{i=1}^{L} x_i - \theta$	y=1 if ≥ 0?
0	0	No fire	0+0-2	0
0	1	No fire	0+1-2	0
1	0	No fire	1+0-2	0
1	1	Fire	1+1-2	1

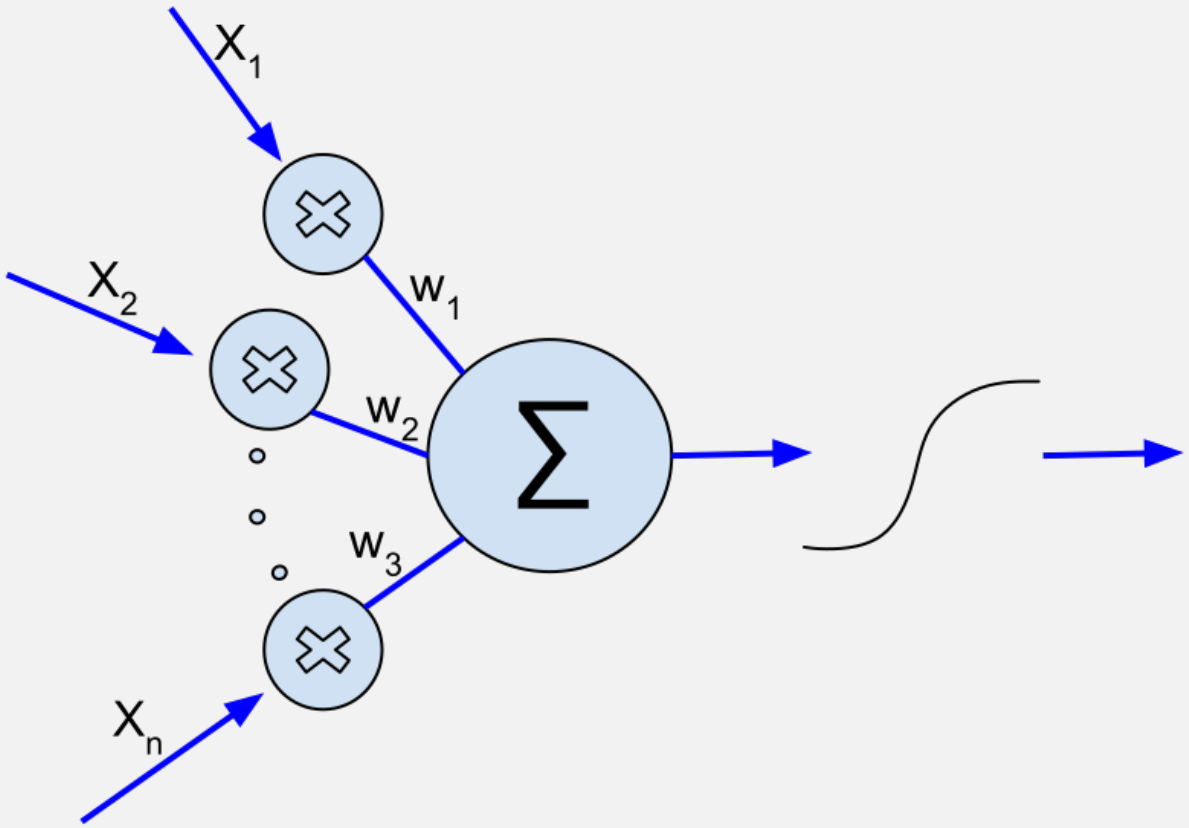
Exclusive "OR" Perceptron



There is no set of thresholds that can define a decision plane that classifies the outcomes for the XOR function using a McCulloch-Pitts neuron

x_1	x_2	XOR outcome	$\sum_{i=1}^{I} x_i - \theta$	y=1 if ≥ 0?
1	1	No fire	(1) $2-\theta < 0$	From (2) and (3) $\theta \leq 1$
1	0	Fire	(2) $1-\theta \geq 0$	
0	1	Fire	(3) $1-\theta \geq 0$	Always incompatible with (1)
0	0	No fire	(4) $0-\theta < 0$	

Artificial Neuron: Simple Perceptron



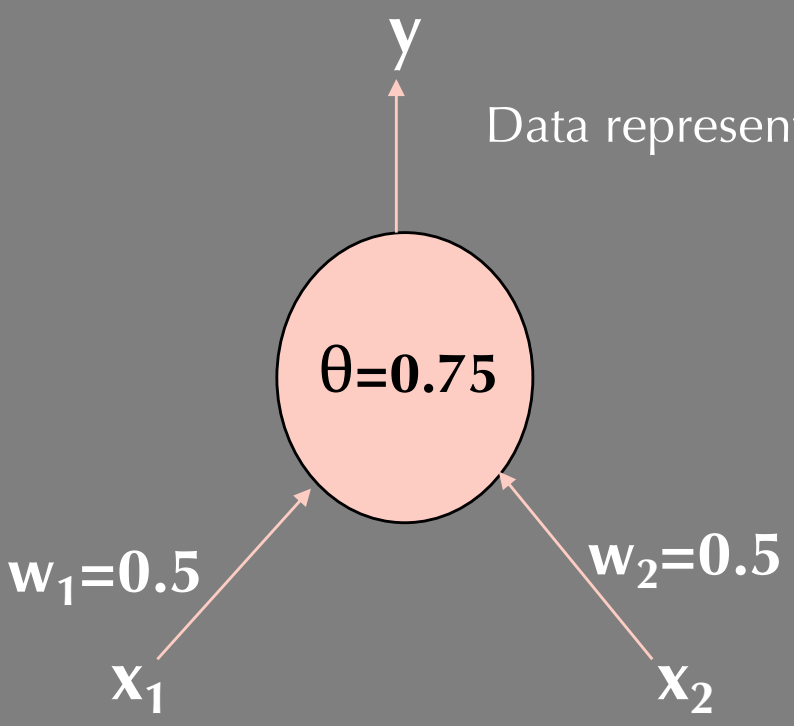
(1) {x} correspond to incoming action potentials from other axons

(2) {w} are set of "neurotransmitter strengths" and Σ is the integrating unit corresponding to axon hillock

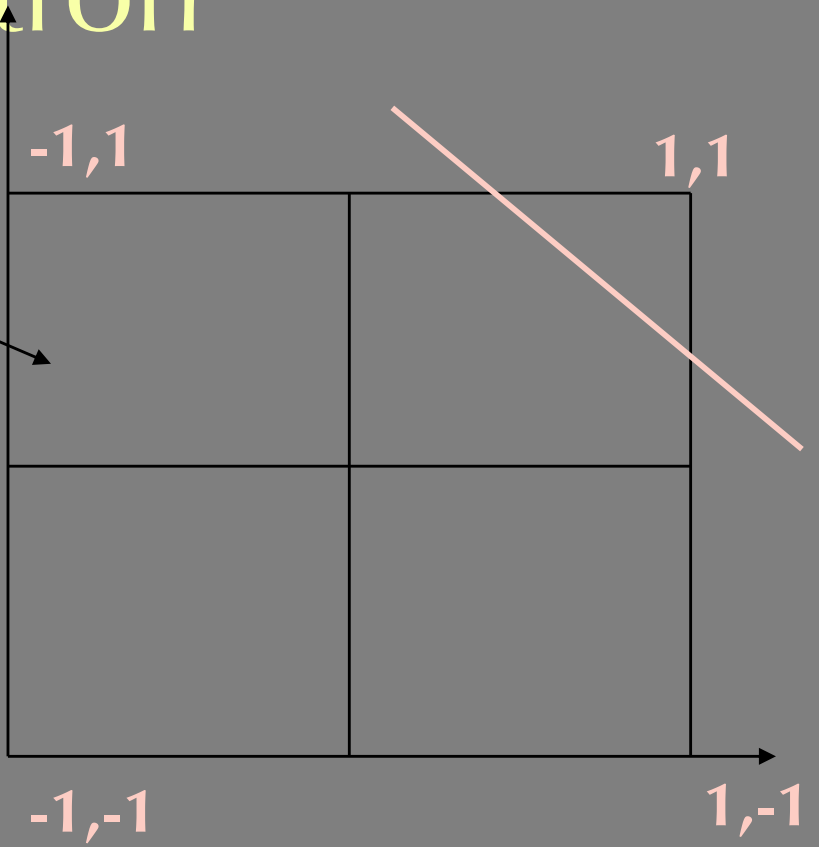
$$\Sigma = \sum_{j=1}^{L} w_j x_j - \theta$$

(3) If integrated signal reaches a threshold, $\boldsymbol{\theta}$, then it fires to give y.

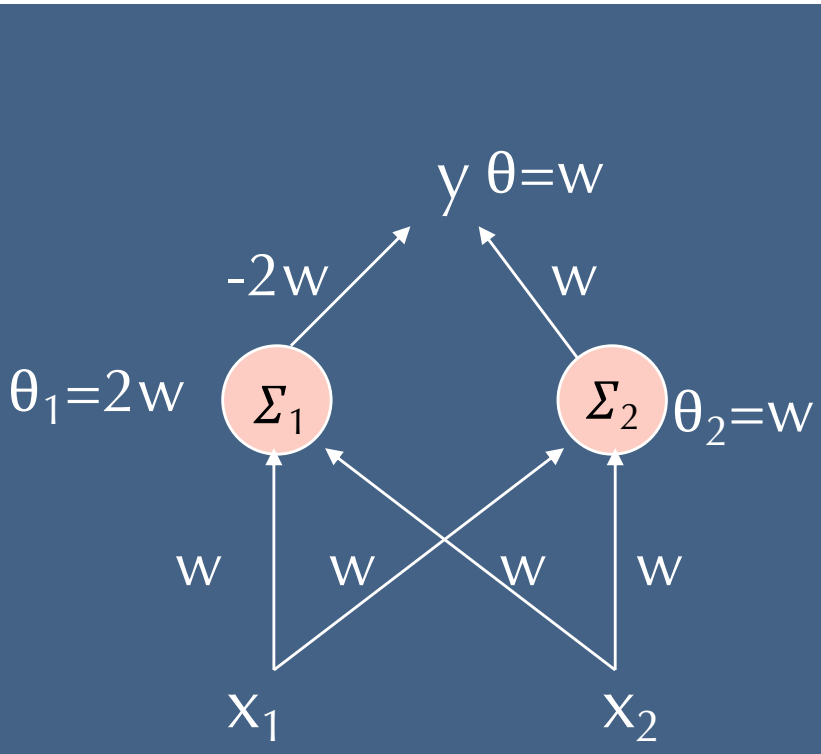
Simple "AND" Perceptron

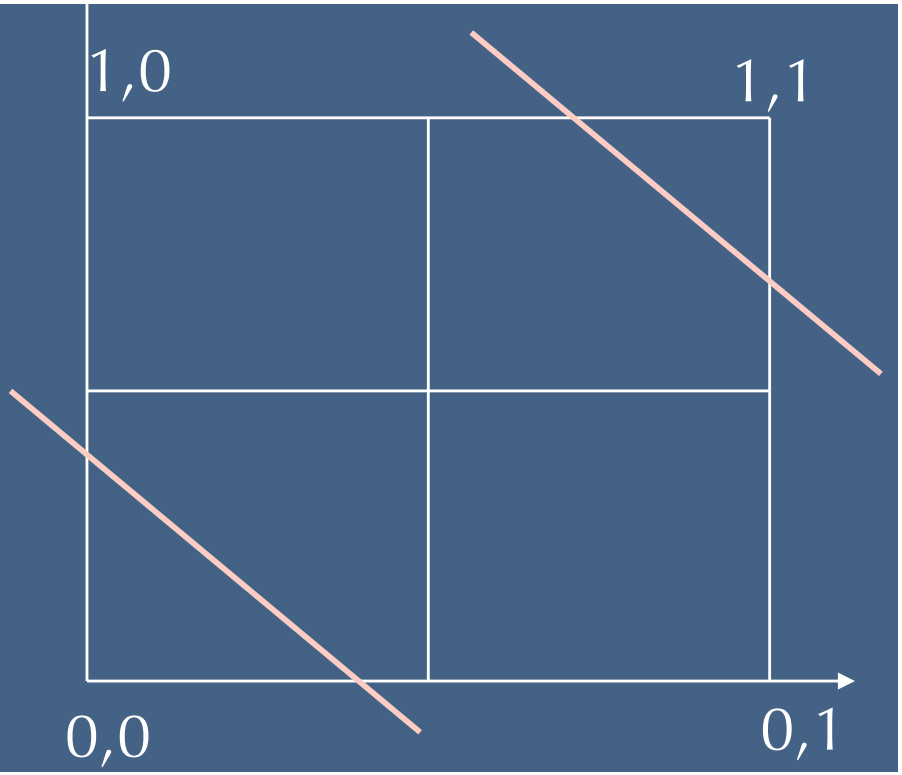


Data representation is changed



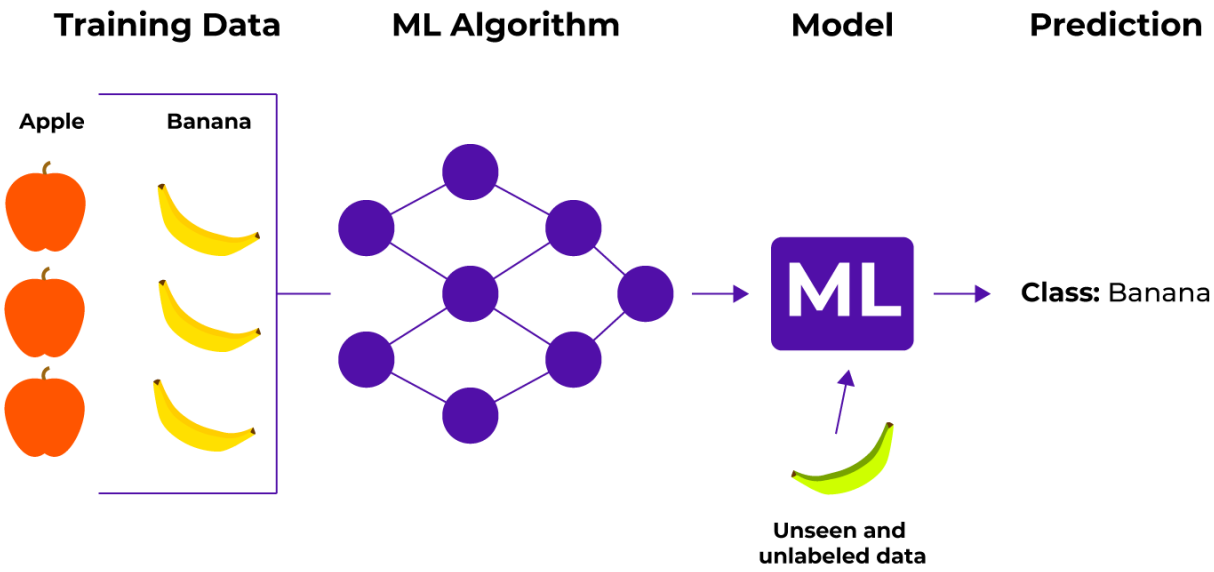
x_1	x_2	AND outcome	$\sum_{j=1}^{L} w_j x_j - \theta$	y=Heaviside
-1	-1	No fire	-1.75	0
-1	1	No fire	-0.75	0
1	-1	No fire	-0.75	0
1	1	Fire	0.25	1





Multi-layered Networks: Solution to XOR

x_1	x_2	$\Sigma_1 = \sum_{i=1}^{L} wx_{i,1} - \theta$	X'_1=1 if ≥ 0?	$\Sigma_2 = \sum_{i=1}^{L} wx_{i,2} - \theta$	X'_2=1 if ≥ 0?	$\sum_{j=1}^{L} wx_j - \theta$	y=1 if ≥ 0
1	1	w+w-2w= 0	1	w+w-w=2w	1	-2w+w-w	-1
1	0	w+0-2w= -w	0	w+0-w=0	1	0+w-w	1
0	1	0+w-2w= -w	0	0+w-w=0	1	0+w-w	1
0	0	0+0-2w= -2w	0	0+0-w=-w	0	0+0-w	-1



$$C = \frac{1}{2}\sum_{\mu}^{N}\sum_{j}^{J}\left[O_j^{\mu} - y_j^{\mu}\right]^2$$

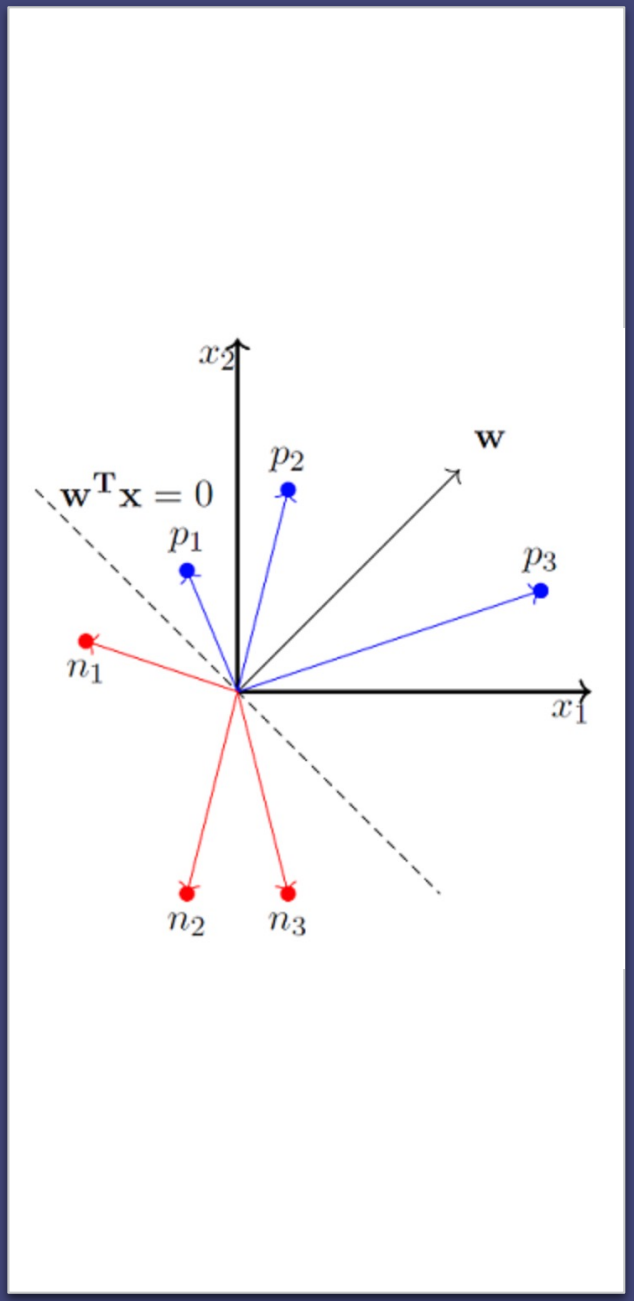
Where μ is a sum over the N examples, and j is the sum over the all J output units.

Supervised Learning

For most real-life problems we can't write down a solution for weights, thresholds that correctly determines the mapping of the input space {x} to output space {y}; i.e. the determination of the decision hyperplane.

We therefore find this decision plane by providing the ANN some examples in which to learn the mapping.

If we are to maximize the fidelity of this mapping, then we propose to minimize the deviation of the predicted output, y, from the observed output, O



Simple Perceptron Learning

We can train the perceptron model to "learn" by adjusting its weights and thresholds with labelled $\vec{x}_\mu, \vec{y}_\mu$ data, where μ is an example of an input,output relationship.

- Given M examples of when $\vec{y}_\mu = 1$, and P examples of when $\vec{y}_\mu = 0$, and $N = M \cup P$ then

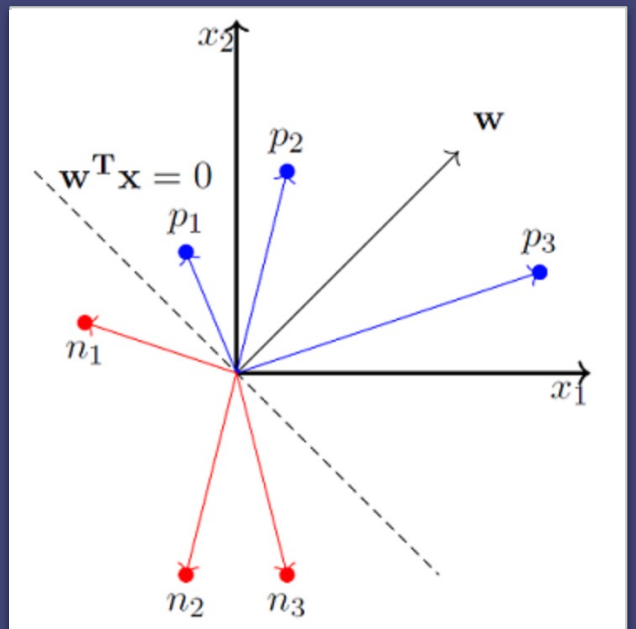
$$\sum_{i=0}^{N} w_i \, x_{i,\mu} = \vec{w}^T \cdot \vec{x}_\mu$$

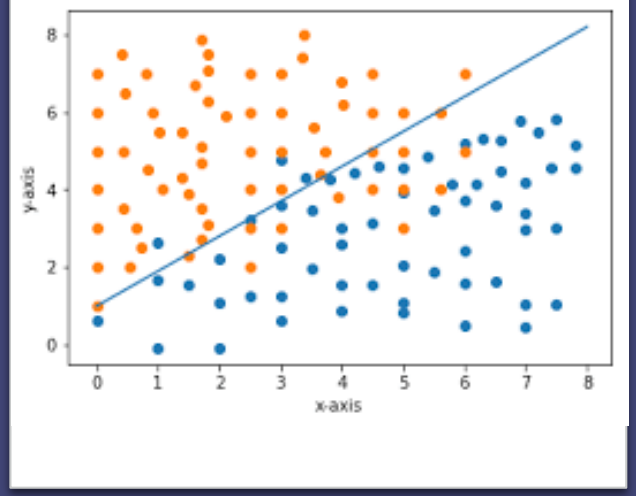
- If $\vec{x}_\mu$ belongs to M: $\vec{w}^T \cdot \vec{x}_\mu > 0$ and if $\vec{x}_\mu$ belongs to P: $\vec{w}^T \cdot \vec{x}_\mu < 0$

Decision plane is $\vec{w}^T \cdot \vec{x}_\mu = 0$ or determined by

$$\frac{\vec{w}^T \cdot \vec{x}_\mu}{\|\vec{w}\| \|\vec{x}_\mu\|} = cos\alpha$$

Weight vector will be defined by an angle $\alpha < 90$ for $(\vec{x}_j \in M)$ and $\alpha > 90$ for $(\vec{x}_j \in P)$





Simple Perceptron Learning

Therefore the Perceptron model "learns" by adjusting its weights and thresholds with labelled $\vec{x}_\mu, \vec{y}_\mu$ data

If $\vec{x}_\mu$ belongs to M:

$$\vec{w}_{new} = \vec{w} + \vec{x}_\mu$$

Else if $\vec{x}_\mu$ belongs to P:

$$\vec{w}_{new} = \vec{w} - \vec{x}_\mu$$

End do

$$cos\alpha_{new} = \vec{w}_{new}^T \cdot \vec{x}_\mu$$
$$= \left(\vec{w} + \vec{x}_\mu\right)^T \cdot \vec{x}_\mu$$
$$= \vec{w}^T \cdot \vec{x}_\mu + \vec{x}_\mu^T \cdot \vec{x}_\mu$$
$$= cos\alpha + \vec{x}_\mu^T \cdot \vec{x}_\mu$$
$$cos\alpha_{new} > cos\alpha$$
$$\alpha_{new} < \alpha$$

$$cos\alpha_{new} = \vec{w}_{new}^T \cdot \vec{x}_\mu$$
$$= \left(\vec{w} - \vec{x}_\mu\right)^T \cdot \vec{x}_\mu$$
$$= \vec{w}^T \cdot \vec{x}_\mu - \vec{x}_\mu^T \cdot \vec{x}_\mu$$
$$= cos\alpha - \vec{x}_\mu^T \cdot \vec{x}_\mu$$
$$cos\alpha_{new} < cos\alpha$$
$$\alpha_{new} > \alpha$$

Does well for linearly separable data!
Will see this in HW