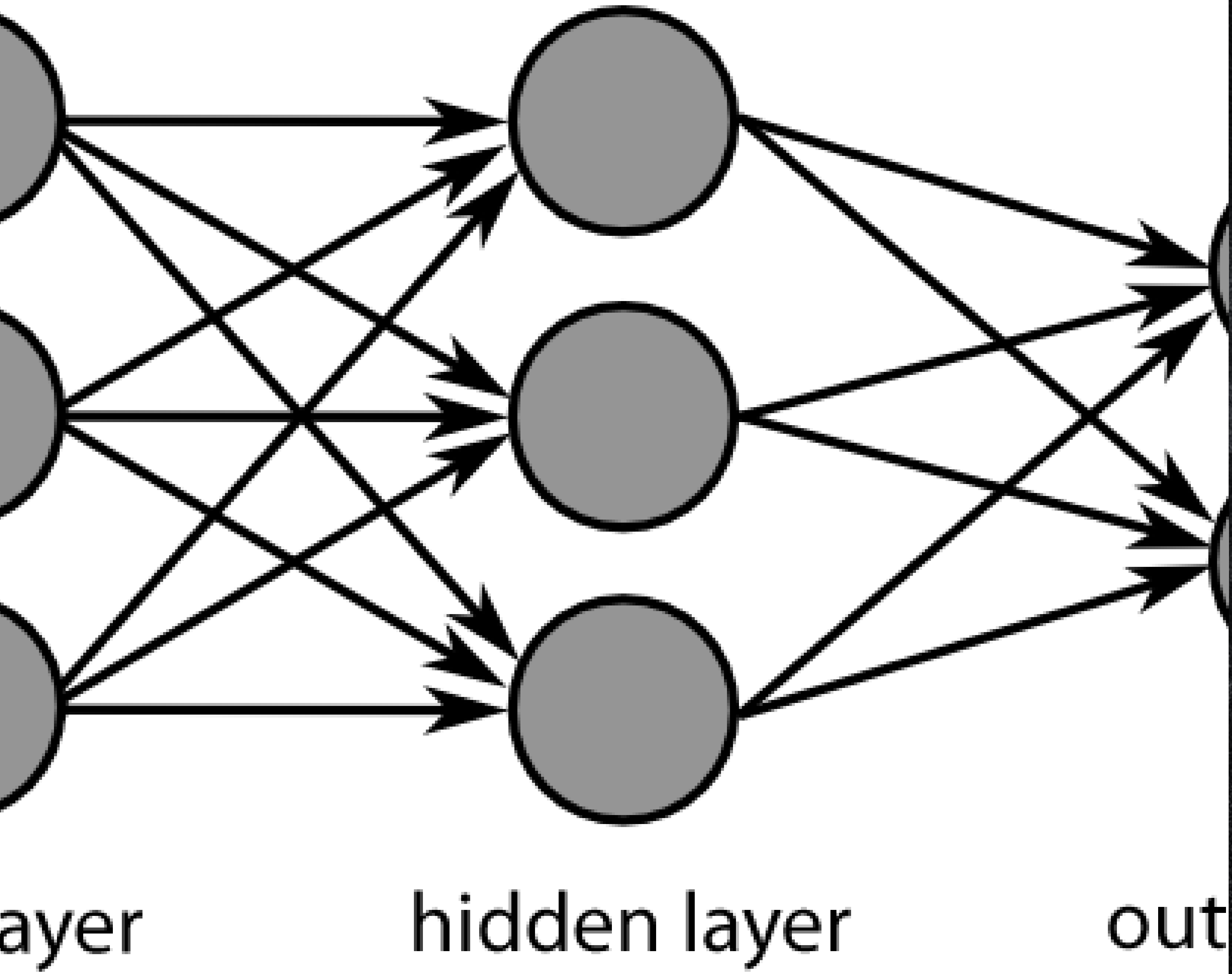
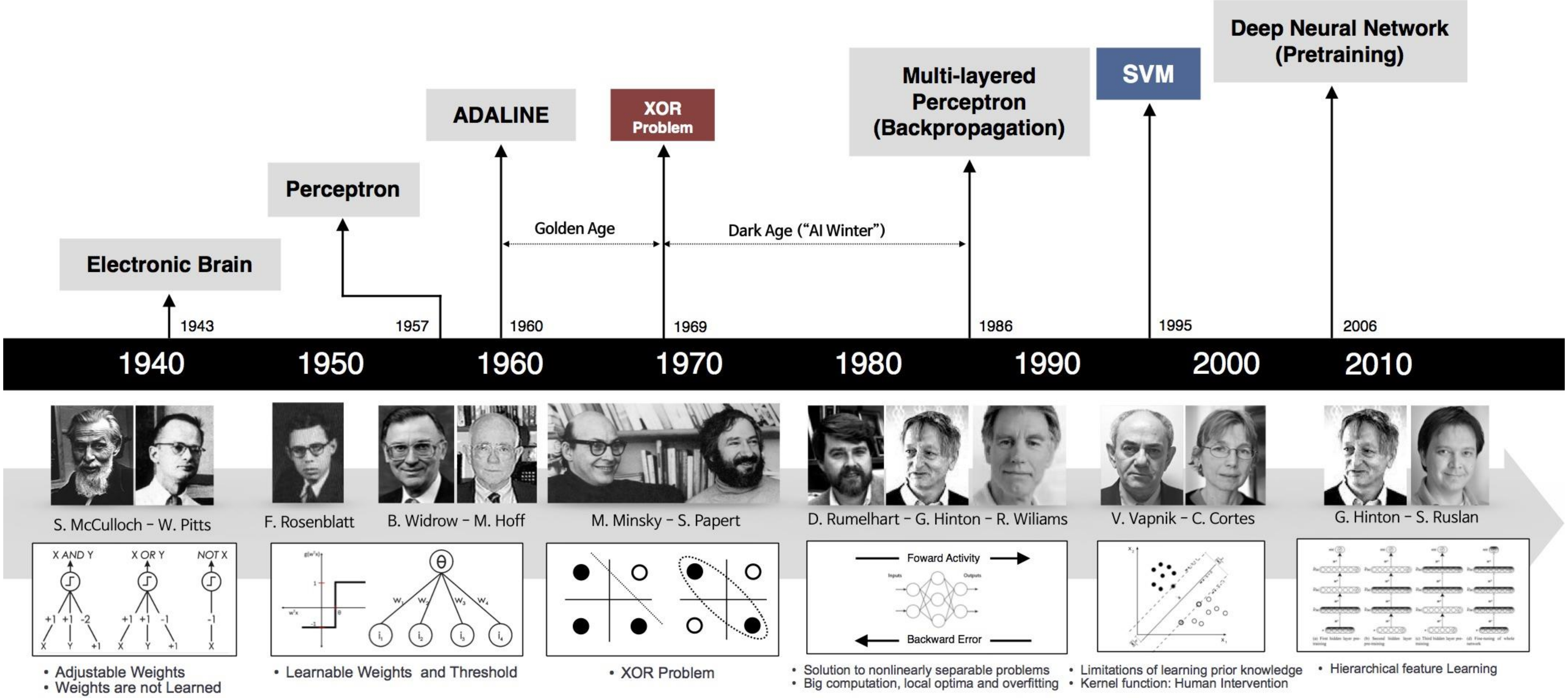


Redes Neuronales

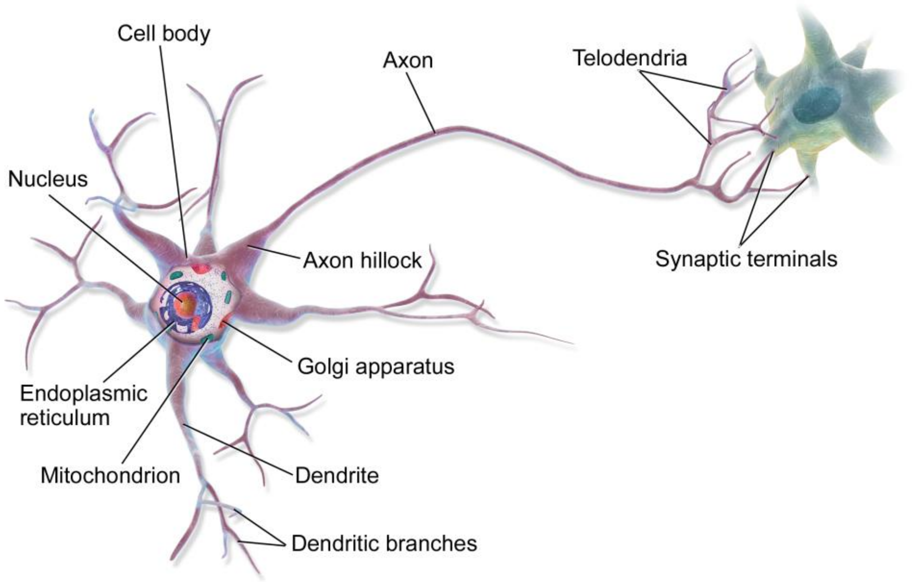
TC3006C



Electronic Brain
Perceptron
ADALINE
XOR Problem
Multi-layered Perceptron (Backpropagation)
SVM
Deep Neural Network (Pretraining)
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch – W. Pitts
F. Rosenblatt
B. Widrow – M. Hoff
M. Minsky – S. Papert
D. Rumelhart – G. Hinton – R. Wiliams
V. Vapnik – C. Cortes
G. Hinton – S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

Introduction

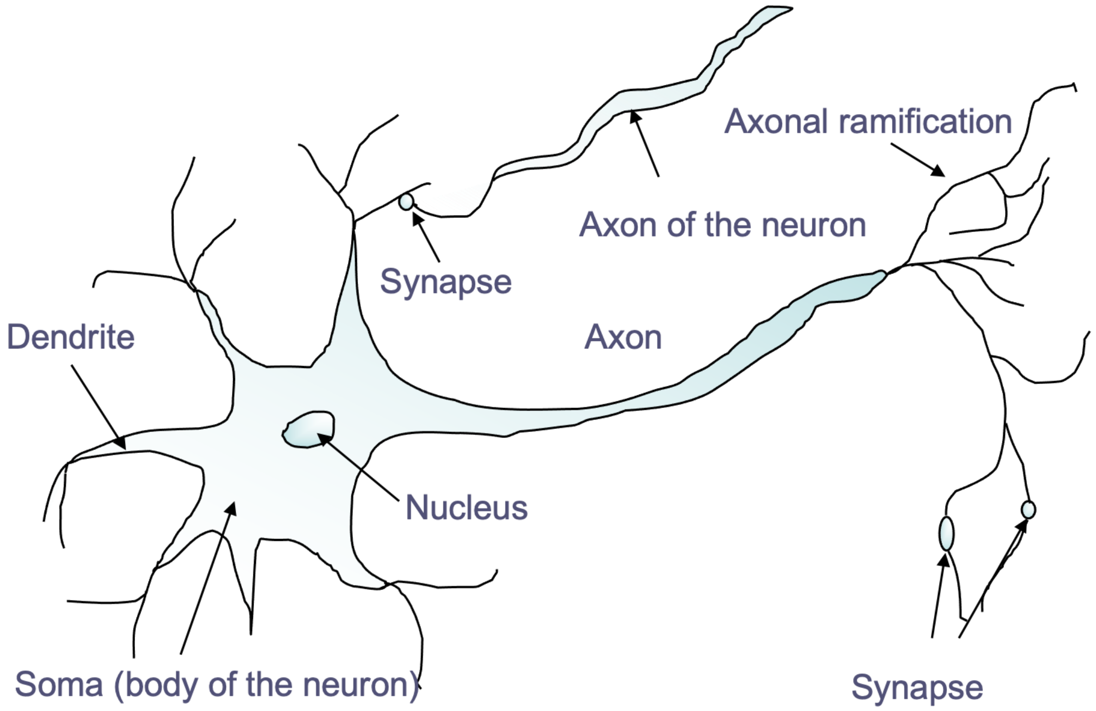
- General, robust, and practical method for learning
  - Real values
  - Discrete values
  - Vector values



- Inspired by biological learning systems
  - Very complex webs of interconnected neurons.
  - Human brain
    - Network 10^{11} neurons
    - Each connected on average to 10^{4} neurons.
    - Neuron switching times 10^{-3} seconds (very slow compared to computers)
    - 10^{-1} second to recognize a familiar face (very fast compared to computers)
      - Sequence of neuron firings cannot be more than a few hundreds due to switching speed.

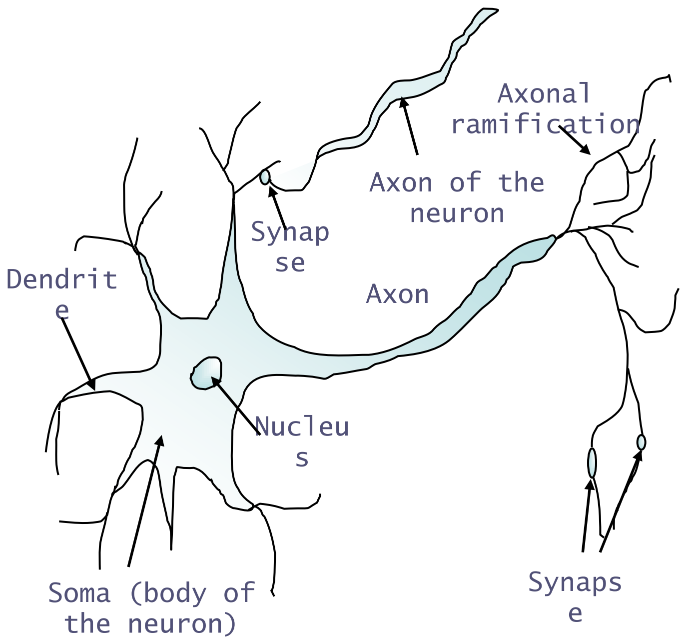
Electrochemical processing

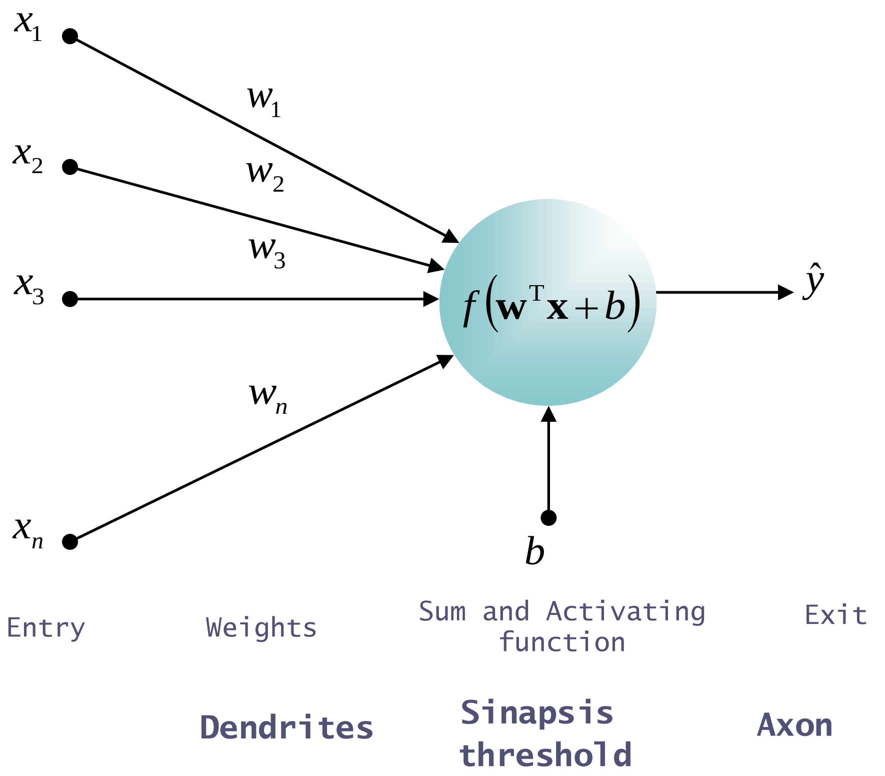
- Electrochemical process 70-100mV, more negative inside the cell.
- Connections generate a difference in electrical potential. This difference jumpstarts the neuron.
- It is really hard to make a computer program based on the functioning of biological neuronal networks. Actually, its functioning is not completely known.
- Sebastian Seung: I am my connectome (http://www.youtube.com/watch?v=HA7GwKXfJB0&feature=colike)
- Brain wiring a no-brainer? Scans reveal astonishingly simple 3D grid structure
  - http://medicalxpress.com/news/2012-03-brain-wiring-no-brainer-scans-reveal.html
  - Video: http://www.youtube.com/watch?v=_oTEhFAAARE



Formal (artificial) neuron

- Parallel signals arrive
- Dendrites are weights: w
- Synapses: Pondered sum
- threshold: activating function



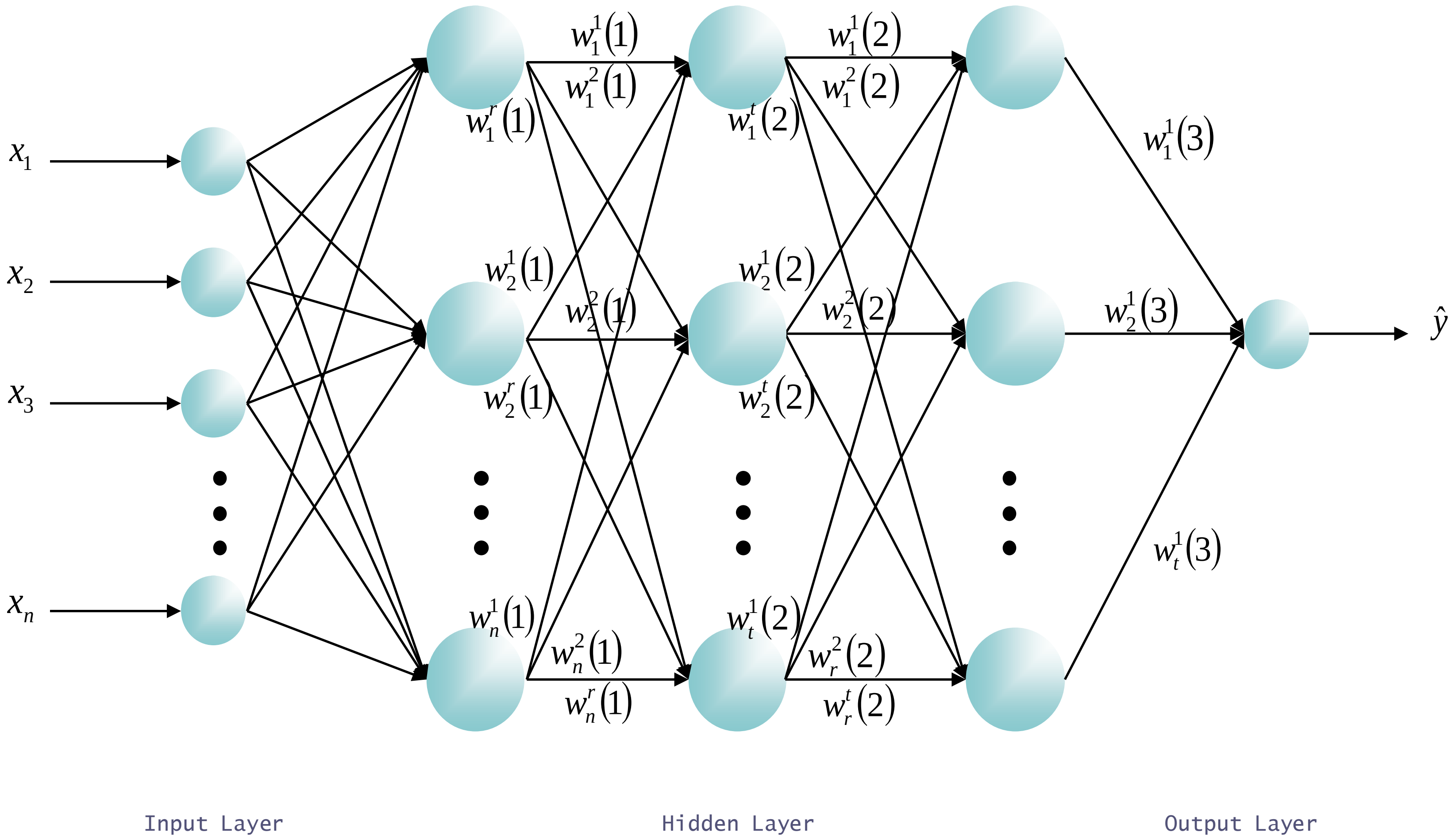


Neurocomputing:
Studies neurocomputing in RNA

1943

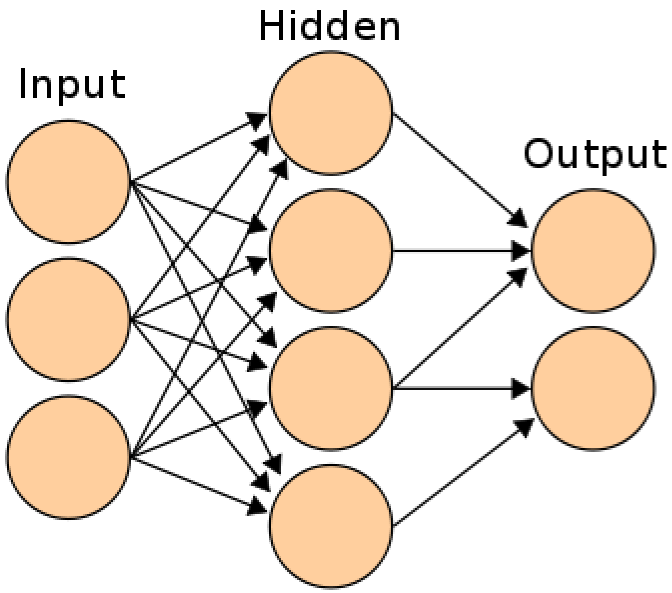
Elements of a Neuronal Network

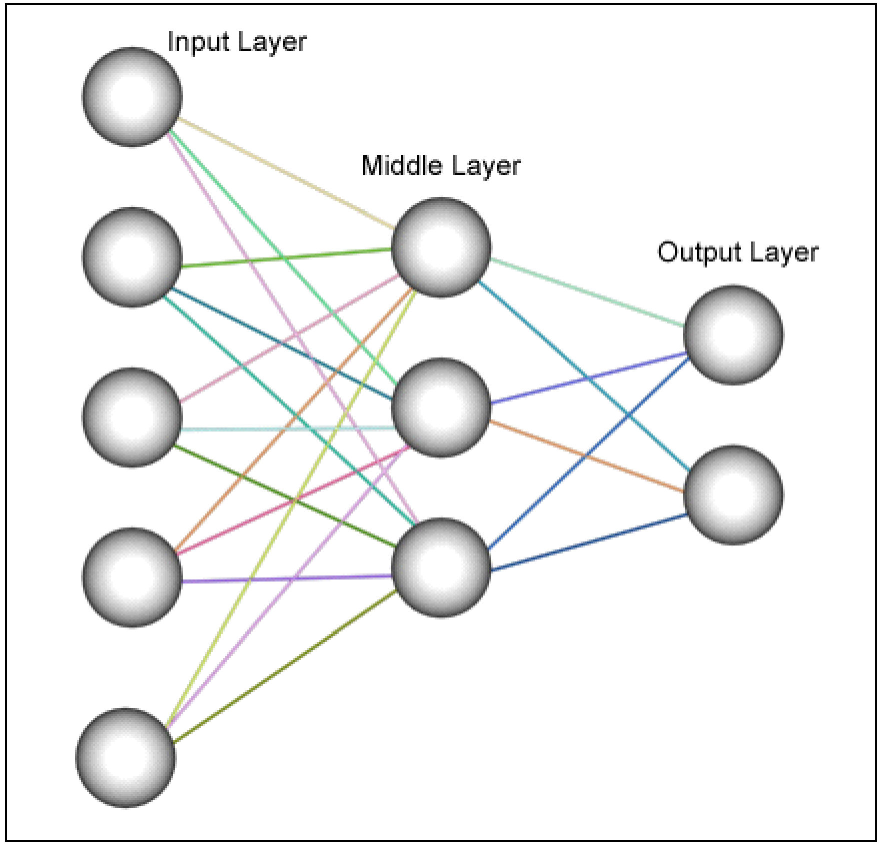
- Network architecture
- Processing units
  - Activation State
  - Activating Function
- Learning Algorithm
- Patterns or Individuals
- Generalization



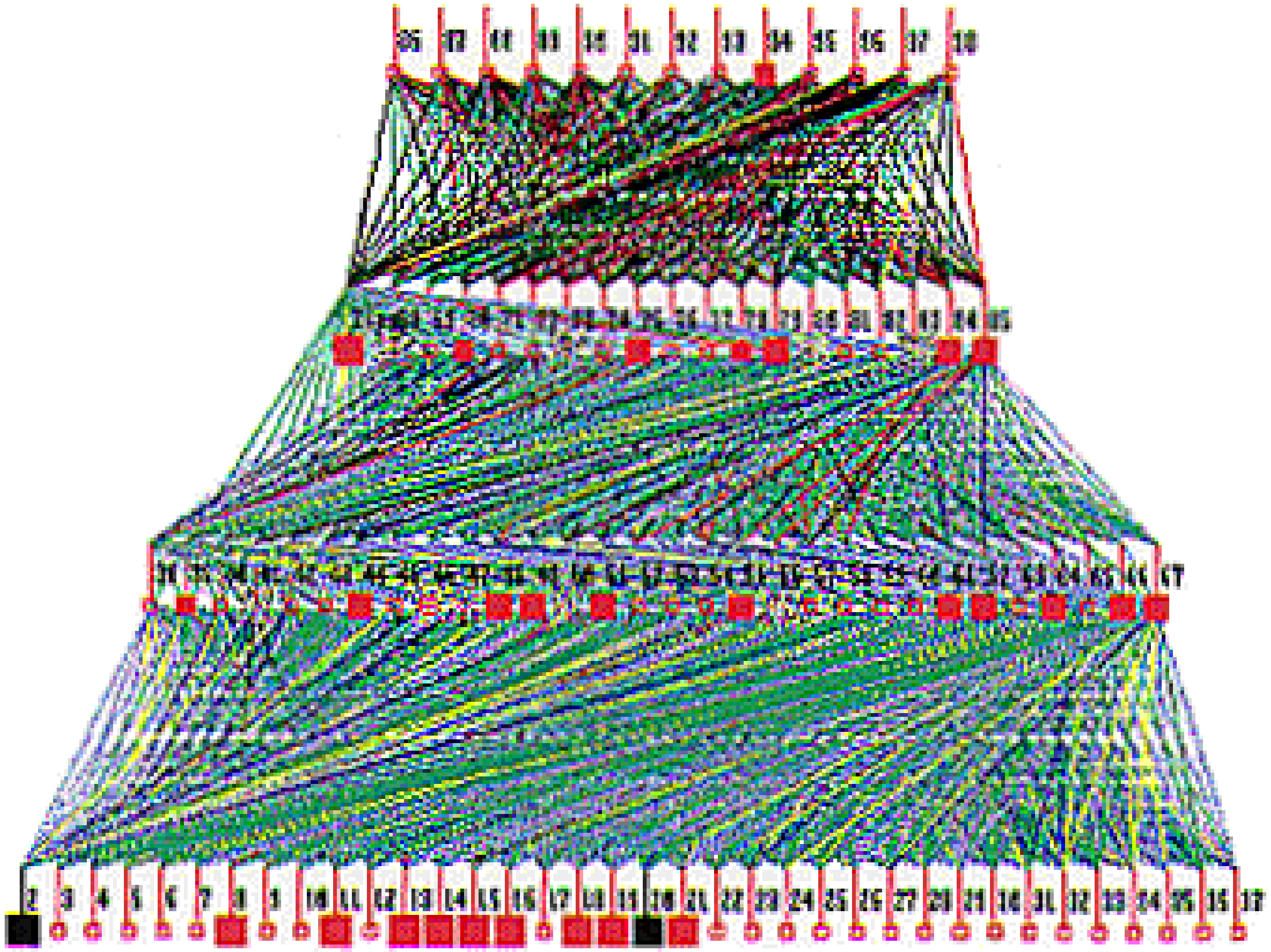
Examples of neuronal networks

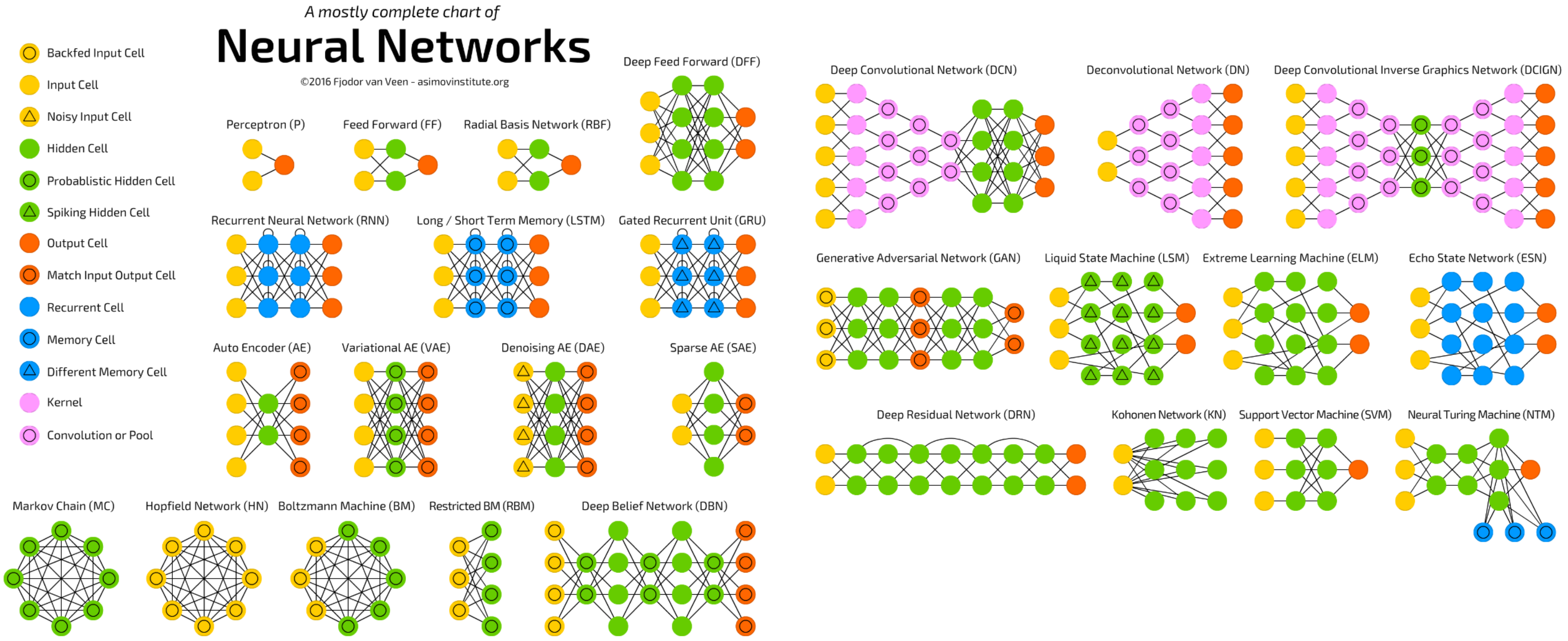
Architectures





Example of a neuronal network for missiles





http://www.asimovinstitute.org/neural-network-zoo/

Training or learning

- Training phase is the interactive **actualization of the weights** in each processing unit.

- Traditional optimization methods: Newton, Square Minimal, Gradient Descents,...

- To each interaction, a learning pattern is presented in the network.

- Each cell calculates its exit and the weights are modified.

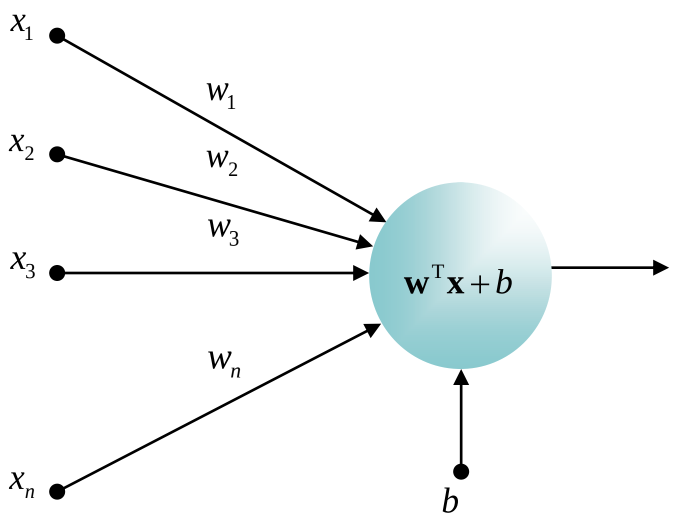
Appropriate problems for ANN learning

- Artificial neural networks are suitable for noisy and complex sensor data.

- Characteristics:
  - Instances are represented by many attribute-value pairs.
  - Target function output may be discrete-valued, real-valued, or vector of several real or discrete valued attributes.
  - Training examples may contain errors.
  - Long training times are acceptable.
  - Fast evaluation of learned target function may be required.
  - Ability of humans to understand learned target function is not important.

Linear Separation

- Formula given decision border



$$\sum w_i x_i + b = 0$$

- *"If all patterns are well classified, with an hyperplane, we say it is a linearly separable problem"*. Minsky y Papert [1988]

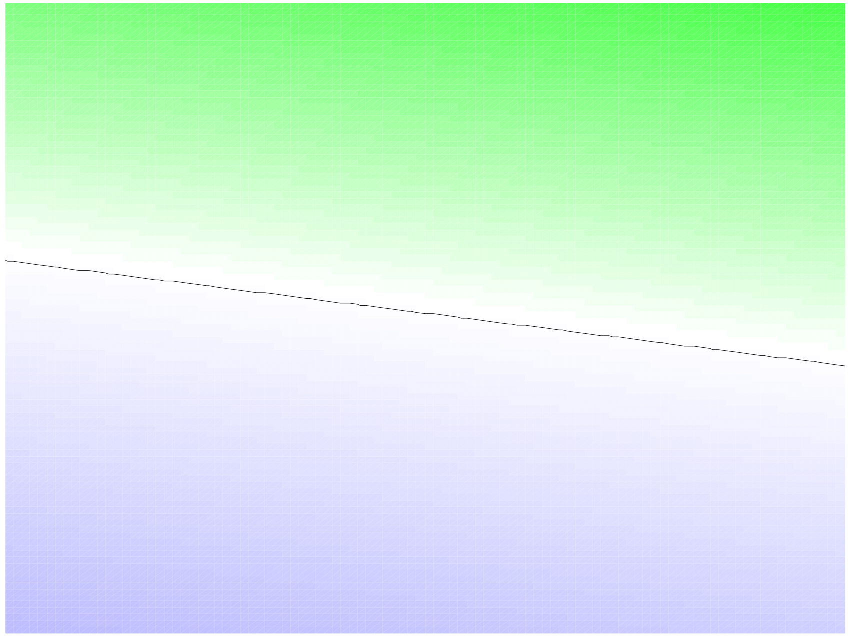
- **One neuron can only separate LS problems**

Linear Separation

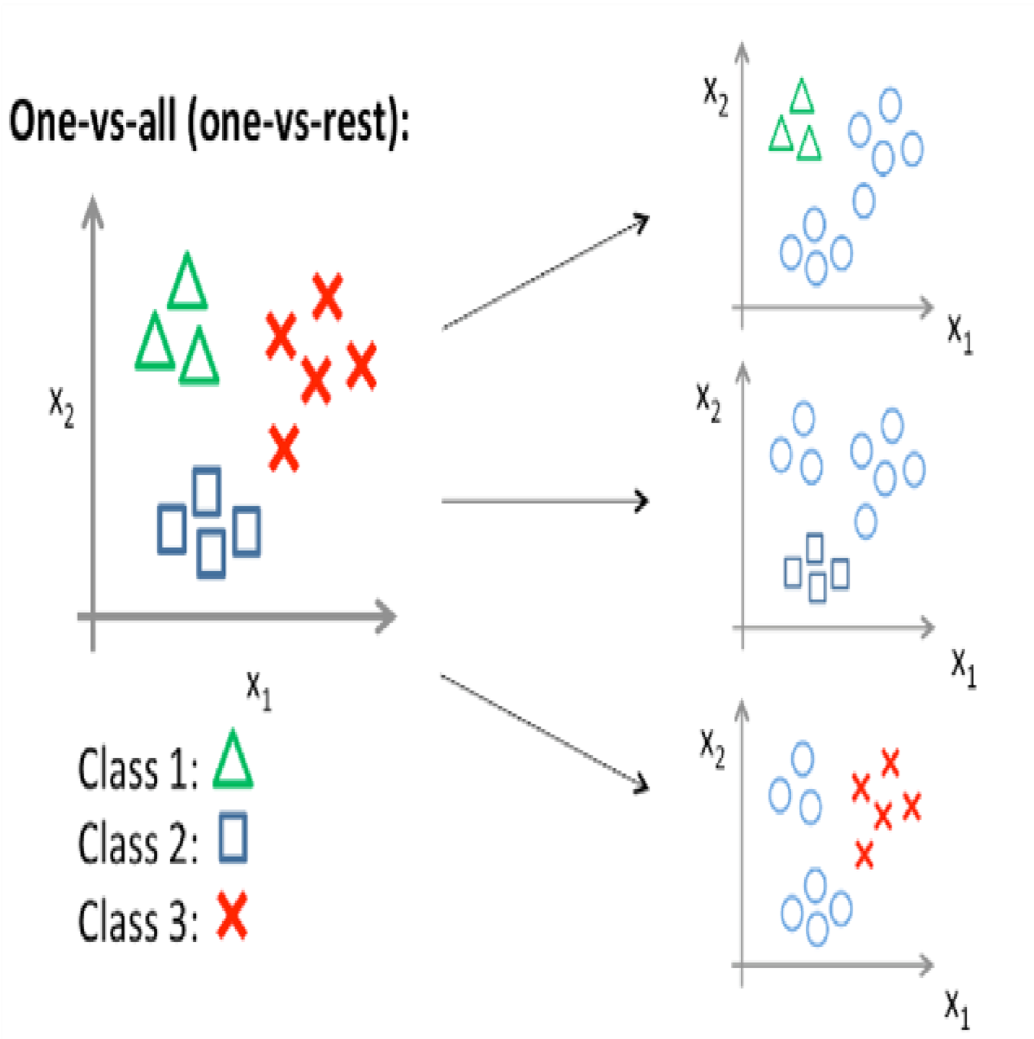
- The regions that are separated, are called separation regions (or decision regions).
- Only one neuron for classification, we can only make linearly separable regions.

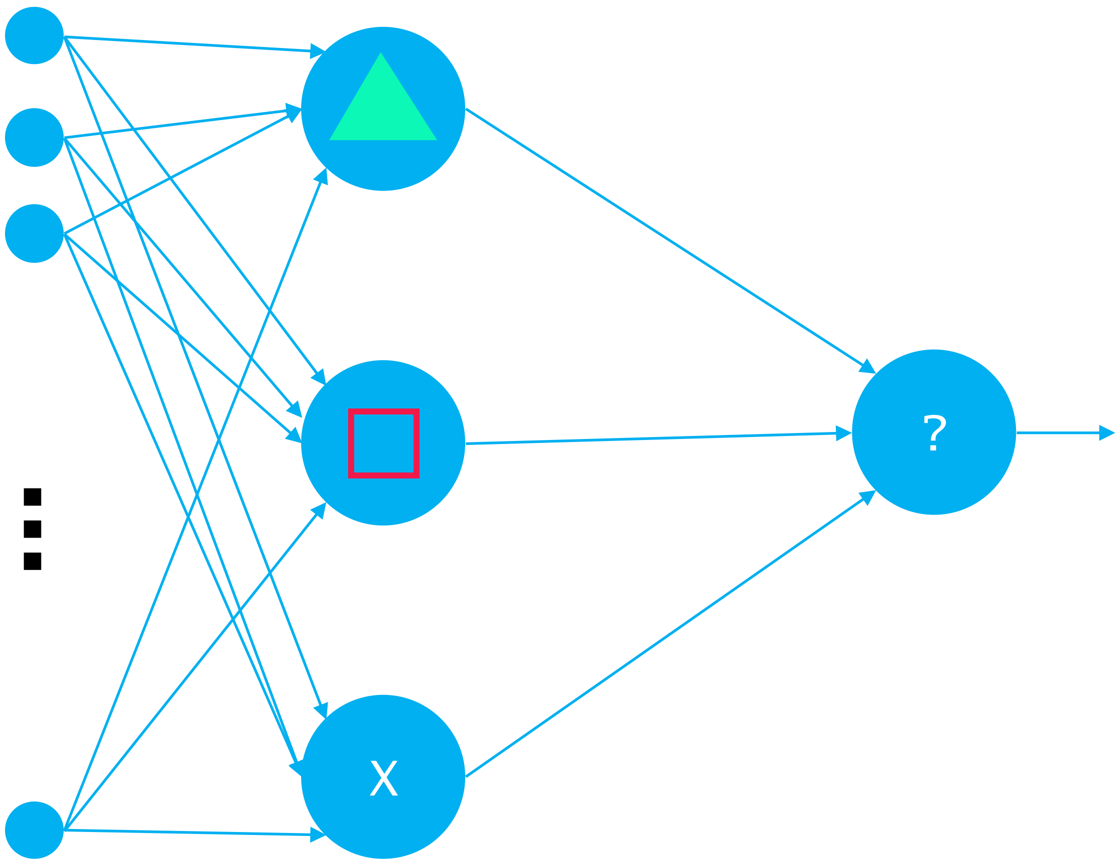
$$x_1 w_1 + x_2 w_2 + b = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}, w_2 \neq 0.$$



Multi-class classification





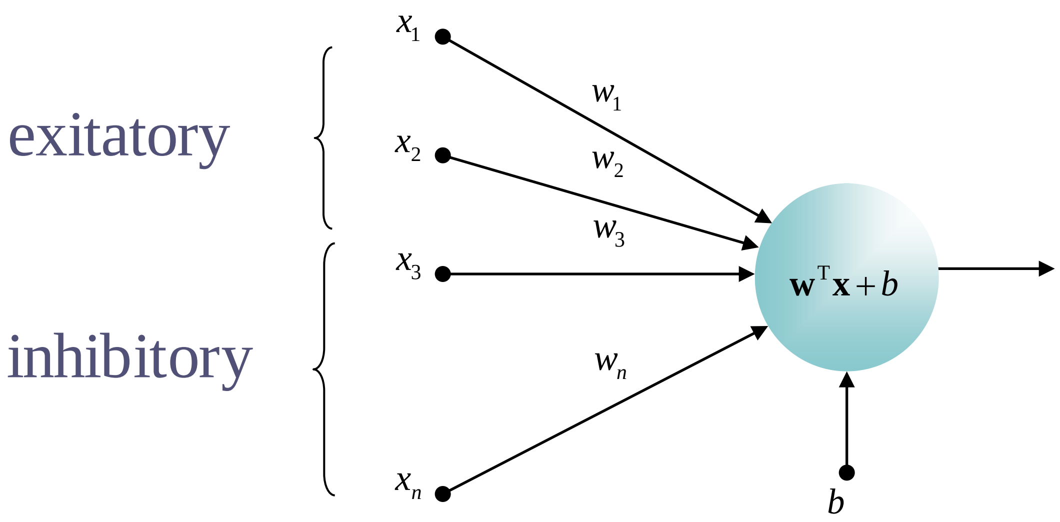
McCulloch – Pitts Model

Artificial Neural Networks

McCulloch – Pitts, NMcCP Characteristics.

- 1943
- Binary Activation.
- Neurons connected through weights.
- Connection is excitatory, positive weight
- Connection is inhibitory, negative weight
- Neuron turns on if pondered sum is higher than the threshold.

McCulloch – Pitts, NMcCP Characteristics.



$$a=\sum w_i x_i+\sum w_i x_i-b$$

$$f(a)=\begin{cases}1\ldots if\ldots a\geq 0\\ 0\ldots if\ldots a<0\end{cases}$$

McCulloch – Pitts, Algorithm

- There is not a learning algorithm, properly said.
- Weights (*w*) and threshold (*b*) are set at the same time.
- Parameterization through visual analysis.
- Executes logical functions.

McCulloch – Pitts, AND Example

X1	X2	w	AND
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

$$\sum w_i x_i - b \geq 0 \qquad 1$$

$$\sum w_i x_i - b < 0 \qquad 0$$

- Objective: find w_1, w_2 and b in order for the function to ensure compliance

$$w_1{=}1,\ w_2{=}1,\ b{=}2$$

McCulloch – Pitts, OR Example

X1	X2	w	OR
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	1

$$\sum w_i x_i - b \geq 0 \qquad 1$$

$$\sum w_i x_i - b < 0 \qquad 0$$

Objetive: find w_1, w_2 and b in order for the function to ensure compliance

w_1=1, w_2 =1, b=1

McCulloch – Pitts, XOR Example

X1	X2	1	XOR
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

$$\sum w_i x_i - b \geq 0 \qquad 1$$

$$\sum w_i x_i - b < 0 \qquad 0$$

Objetive: find w_1, w_2 and b in order for the function to ensure compliance

McCulloch-Pitts Applications

- Once we know how to decompose a logic function in the *sum of products* or *products of sums*, it is possible to implement any type of function as a 3-layer neuronal network.

$$f = \overline{x_1} x_2 x_3 + x_1 x_2 \overline{x_3}, \ldots$$

- Advantages: speed
- Distributed calculus

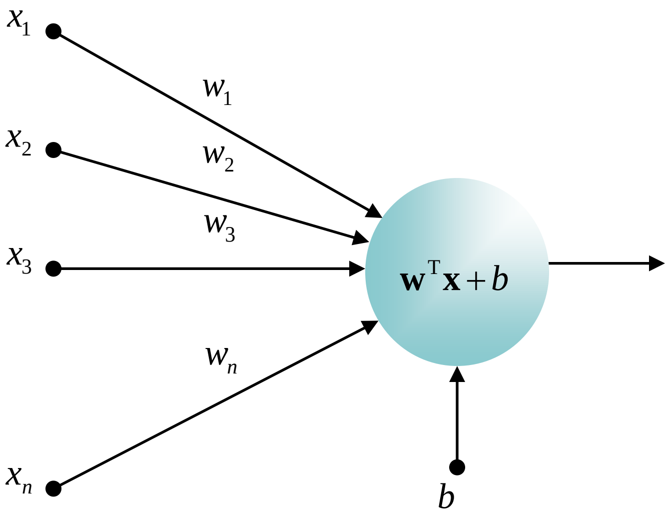
Hebb Learning

Artificial Neural Networks

Hebb Learning, Characteristics

- First known learning algorithm (1949)
- Learning, modification Δw of the weights
  - Modification if two neurons are interconnected and on at the same time
  - Algorithm says nothing about neurons turned off.
- Hebb modified, more robust if they are taken on or off.
- Used for other types of learning algorithms.
- Bipolar inputs and outputs.
  - Inputs and outputs in: 1, –1.

Hebb Learning, Learning Algorithm



$$\sum w_i x_i - b = 0$$
$$w_i(new) = w_i(old) + \Delta w_i$$
$$\Delta w_i = x_i y_i$$

- Initialization, $w_i = 0,\ i = 0,\ldots,n$
- For every l elements of the database.

$$w_i(new) = w_i(old) + \Delta w_i$$
$$\Delta w_i = x_i y_i$$

$$\sum w_i x_i + b \geq 0 \qquad 1$$
$$\sum w_i x_i + b < 0 \qquad 0$$

Hebb Learning, Examples of Learning AND 1, -1

			Δb			b		
x1	x2	y	Δw0	Δw1	Δw2	w0	w1	w2
						0	0	0
-1	-1	-1	-1	1	1	-1	1	1
-1	1	-1	-1	1	-1	-2	2	0
1	-1	-1	-1	-1	1	-3	1	1
1	1	1	1	1	1	-2	2	2

Hebb Learning, Examples of Learning AND 1, -1

$$2x_1 + 2x_2 - 2 = 0$$

All the elements suffer modifications,

Hebb Learning, Examples of Learning, OR 1,-1

			Δb			b		
x1	x2	y	Δw0	Δw1	Δw2	w0	w1	w2
						0	0	0
-1	-1	-1	-1	1	1	-1	1	1
-1	1	1	1	-1	1	0	0	2
1	-1	1	1	1	-1	1	1	1
1	1	1	1	1	1	2	2	2

Hebb Learning, Examples of Learning, AND, output -1

			Δb			b		
x1	x2	y	Δw0	Δw1	Δw2	w0	w1	w2
						0	0	0
0	0	-1	-1	0	0	0	0	-1
0	1	-1	-1	0	-1	0	-1	-2
1	0	-1	-1	-1	0	-1	-1	-3
1	1	1	1	1	1	0	0	-2

Classical error: mixing 0, -1 y 1

Hebb Learning, Examples of Learning AND Output -1

$$-2x_2 = 0$$

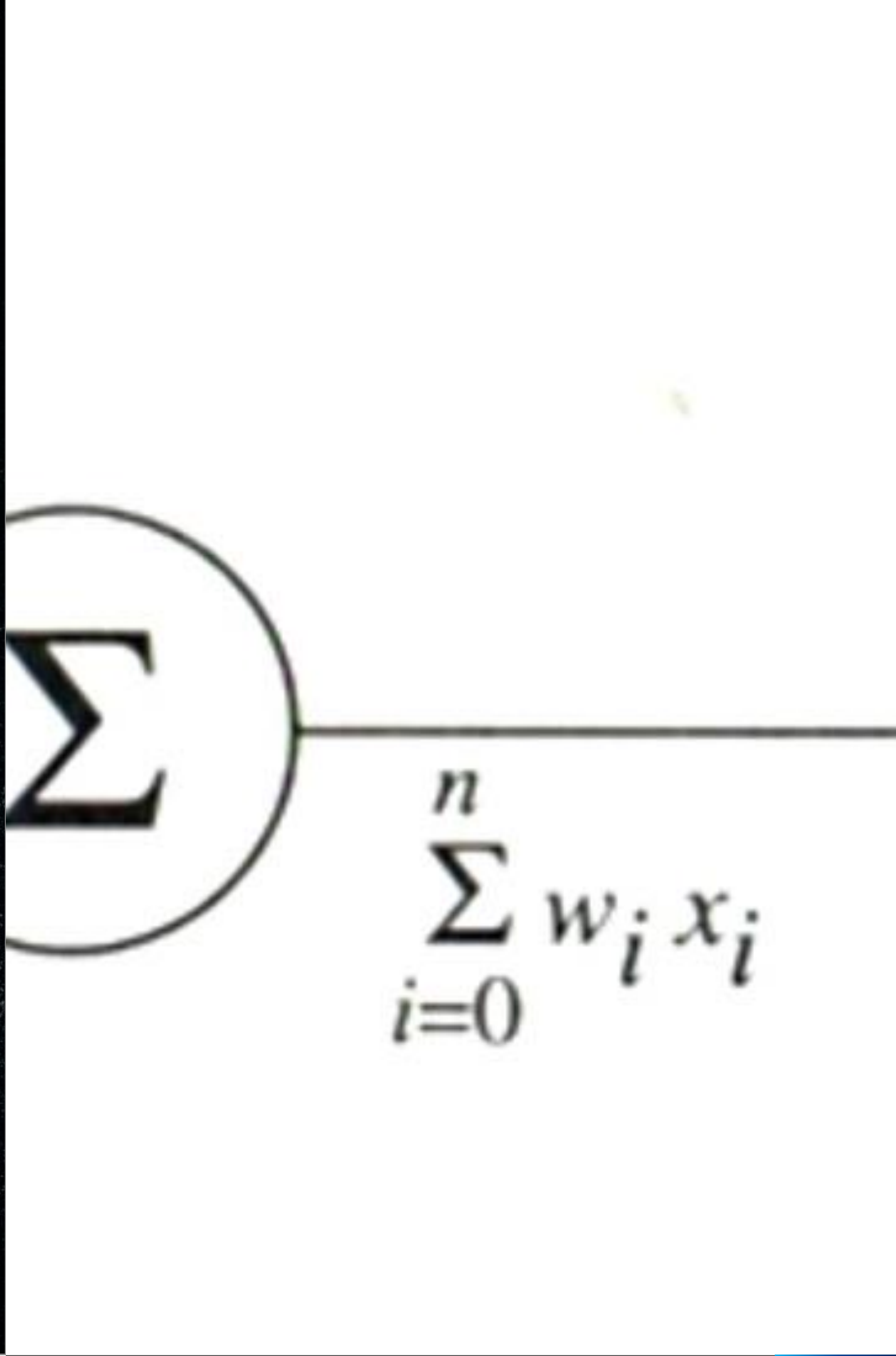
Inputs and error modification play a very important role.

Hebb Learning

- First learning algorithm that follows a defined pattern
- Quite sensitive to the types of inputs and outputs: binary or bipolar.
- The algorithm does not consider non-linearly separable problems.
- Relatively easy problems

Perceptron

Redes Neuronales



Perceptron

Characteristics

- Perceptron : Rossemblat ,1962

- Is guaranteed under certain characteristics, the learning algorithm converges to the optimal solution to the problem.

- The perceptrons were developed for vision.

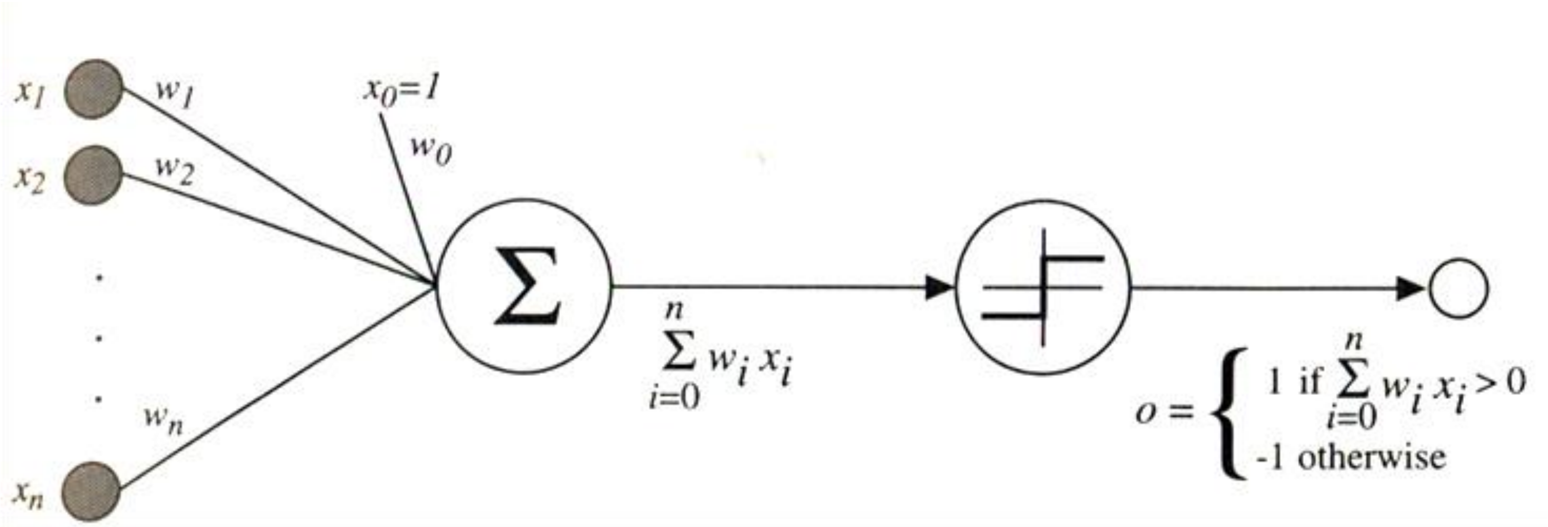
- Inspired by the retina of a fly, which had a captor, a unit of processing and output.

- Inversely performing, about the number of observations.

- Perceptron convergence theorem:

  - For one unit(neuron) and 2 different classes, if the problem, is linearly separable, learning will be completed in a finite number of iterations.

Perceptron



- Input: vector of real-values.
- Calculates a linear combination of these inputs
- Output:
  - 1 if the result is greater than some threshold
  - -1 otherwise
- Equation: $$o(x_1,\ldots,x_n) = \begin{cases} 1 & if \quad w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0 \\ -1 & otherwise \end{cases}$$

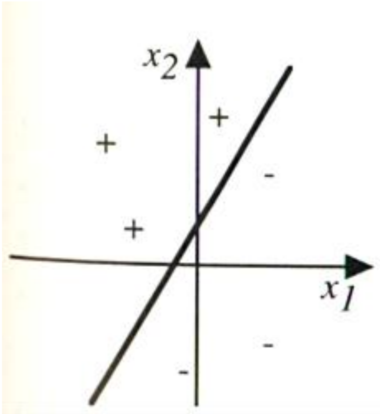
Perceptron …

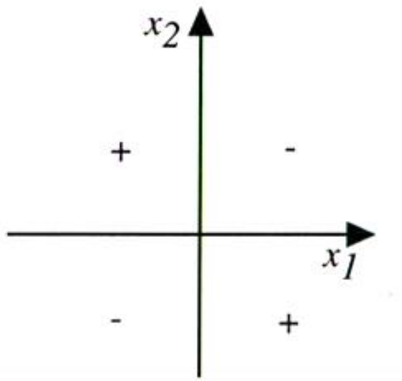
- w_0 is usually negative (- w_0) as it is a **threshold** that must be surpassed to in order for the perceptron to output 1.
- If we add an additional constant X_0 = 1 then we can write: $\sum_{i=0}^{n} w_i x_i > 0$
- Or in vector form: $\vec{w} \cdot \vec{x} > 0$
- So the perceptron function can be written: $o(\vec{x}) = \operatorname{sgn}\left(\vec{w} \cdot \vec{x}\right)$
- where: $\operatorname{sgn}(y) = \begin{cases} 1 & if \quad y > 0 \\ -1 & otherwise \end{cases}$
- So the space of candidate hypothesis considered in perceptron learning are the set of weights:

$$H = \left\{ \vec{w} \,\middle|\, \vec{w} \in \Re^{(n+1)} \right\}$$

Representational power of perceptrons

- Represents hyperplane decision surfaces in n-dimensional spaces.
  - 1 for instances on one side of the hyperplane.
  - -1 for instances on the other side.





- Called *linearly separable* sets of examples.

Representational …

Boolean function

- A single perceptron can be used to represent many boolean functions.
  - Usually 1 if TRUE and -1 if FALSE.
  - All primitive booleans functions: AND, OR, NAND, NOR.
  - Very easy if all input weights all set to the same value (e.g. 1) and setting the threshold w_0 accordingly.
- **Every** boolean function can be represented by some network of interconnected units **based on** primitive boolean functions
  - SOP or POS (two levels deep).
- An input to a boolean function can be negated by **changing the sign** of the corresponding input weight.

Perceptron training rule

Perceptron training rule …

1. Begin with random weights.

2. Apply iteratively the perceptron to each training example (Forward Propagation)
   1. Modify perceptron weights whenever it misclassifies an example.
   2. Use the ***perceptron training rule***: $w_i \leftarrow w_i + \Delta w_i$

   where: $\Delta w_i = \eta (t - o) x_i$

   - t is the target output for current training example
   - o is the output generated by the perceptron
   - η is the learning rate – moderates degree to which weights change
     - usually a small value (0.1) and sometimes is made to decay as iterations increase.

3. Repeat step 2 until perceptron classifies all training examples correctly.

Perceptron training rule …

- Converges toward successful weight values within a finite number of applications of the perceptron training rule.

- Provided:
  - Training examples are linearly separable.
  - A sufficiently small learning rate is used (Minsky and Papert, 1969).

Gradient descent and Delta rule

Gradient descent and Delta rule

- Converges toward a best-fit approximation to the target function when training examples are not linearly separable.
- Uses the ***gradient descent*** to search the hypothesis space of possible weight vectors.
- This rule is the basis of the **Backpropagation** algorithm
  - Can learn networks with many interconnected units.
  - Gradient descent can also serve as the basis for linear algorithms that learn continuously parameterized hypothesis.

Gradient descent and Delta rule …

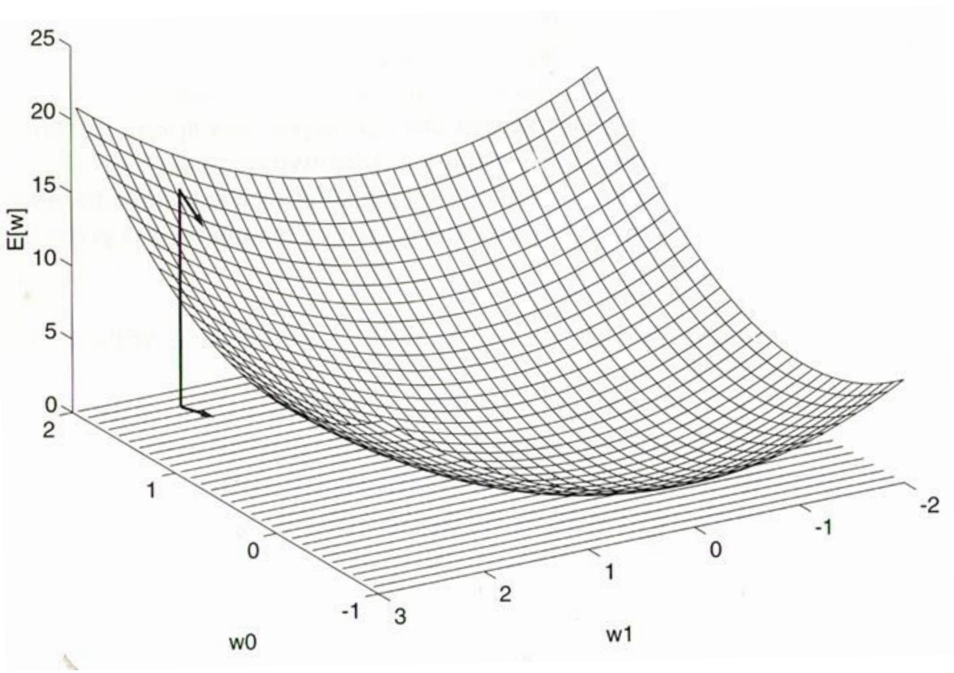
- Unthresholded perceptron $o(\vec{x}) = \vec{w} \cdot \vec{x}$

  - Linear unit: correspond to the first stage of perceptron without the threshold.

- Training error
  - Of a hypothesis relative to the training examples

$$E(w) \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

  - *D*: set of training examples
  - t_d: target output
  - o_d: linear unit output



Gradient descent and Delta rule …

- Gradient descent search determines a weight vector that minimizes *E* by:
  1. Starting with an arbitrary initial weight vector.
  2. Repeatedly modifying it in small steps.
  3. At each step, the weight vector is altered in the direction that produces the steepest descent along the error surface.
  4. Process continues until the global minimum error is found.

Gradient descent and Delta rule …

- Gradient of *E* with respect to $\vec{w}$ is: $\nabla E(\vec{w})$
  - Vector that specifies the direction that produces the *steepest increase* in *E*.
- Training rule for *gradient descent* is: $\vec{w} \leftarrow \vec{w} + \Delta \vec{w}$

  where: $\Delta \vec{w} = -\eta \nabla E(\vec{w})$

- For the component form it can be rewritten as:

$$w_i \leftarrow w_i + \Delta w_i$$

  - where:

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

Gradient descent and Delta rule …

Gradient-Descent (training_examples, η)

Each training example is a pair of the form $\langle \vec{x}, t \rangle$, where $\vec{x}$ is the vector of input values, and t is the target output value, η is the learning rate.

1. Initialize each w_i to some small random value.
2. Until the termination condition is met, Do
    1. Initialize each Δw_i to zero.
    2. For each $\langle \vec{x}, t \rangle$ in training_examples, Do
        1. Input the instance $\vec{x}$ to the unit and compute the output o *(Forward Propagation)*
        2. For each linear unit weight w_i, Do

$$\Delta wi \leftarrow \Delta wi + \eta(t - o)x_i$$

    3. For each linear unit weight w_i, Do

$$w_i \leftarrow w_i + \Delta w_i$$

Gradient descent and Delta rule …

- Advantages
  - Search strategy for large or infinite hypothesis space.
    - Hypothesis space contains continuously parameterized hypothesis (like weights in a linear unit)
    - Error can be differentiated with respect to these hypothesis parameters.

- Disadvantages
  - Converging to a local minimum can be quite slow – thousand of descent steps.
  - No guarantee of finding the global minimum with multiple local minima in the error surface

Class Exercise

- Using and AND or an OR
- Perform a Feed Forward Propagation by hand

Gradient descent and Delta rule …

- Stochastic gradient descent (incremental gradient descent)
  - Alternative to alleviate disadvantages of gradient descent.
  - Instead of updating weights after summing over *all* training examples, weights are updated after *each* training example is presented.
  - **Delta rule** (LMS rule, Adaline rule, Widrow-Hoff rule):

$$\Delta wi \leftarrow \eta(t-o)x_i$$

  - So, the error is defined over each individual training example:

$$E_d(\vec{w}) = \frac{1}{2}(t_d - o_d)^2$$

Gradient descent and Delta rule …

Stochastic-Gradient-Descent (training_examples, η)

Each training example is a pair of the form $\langle \vec{x}, t \rangle$, where $\vec{x}$ is the vector of input values, and t is the target output value, η is the learning rate.

1. Initialize each w_i to some small random value.
2. Until the termination condition is met, Do
    1. Initialize each Δw_i to zero.
    2. For each $\langle \vec{x}, t \rangle$ in training_examples, Do
        1. Input the instance $\vec{x}$ to the unit and compute the output *o*. *(Forward Propagation)*
        2. For each linear unit weight w_i, Do

$$wi \leftarrow wi + \eta(t - o)x_i$$