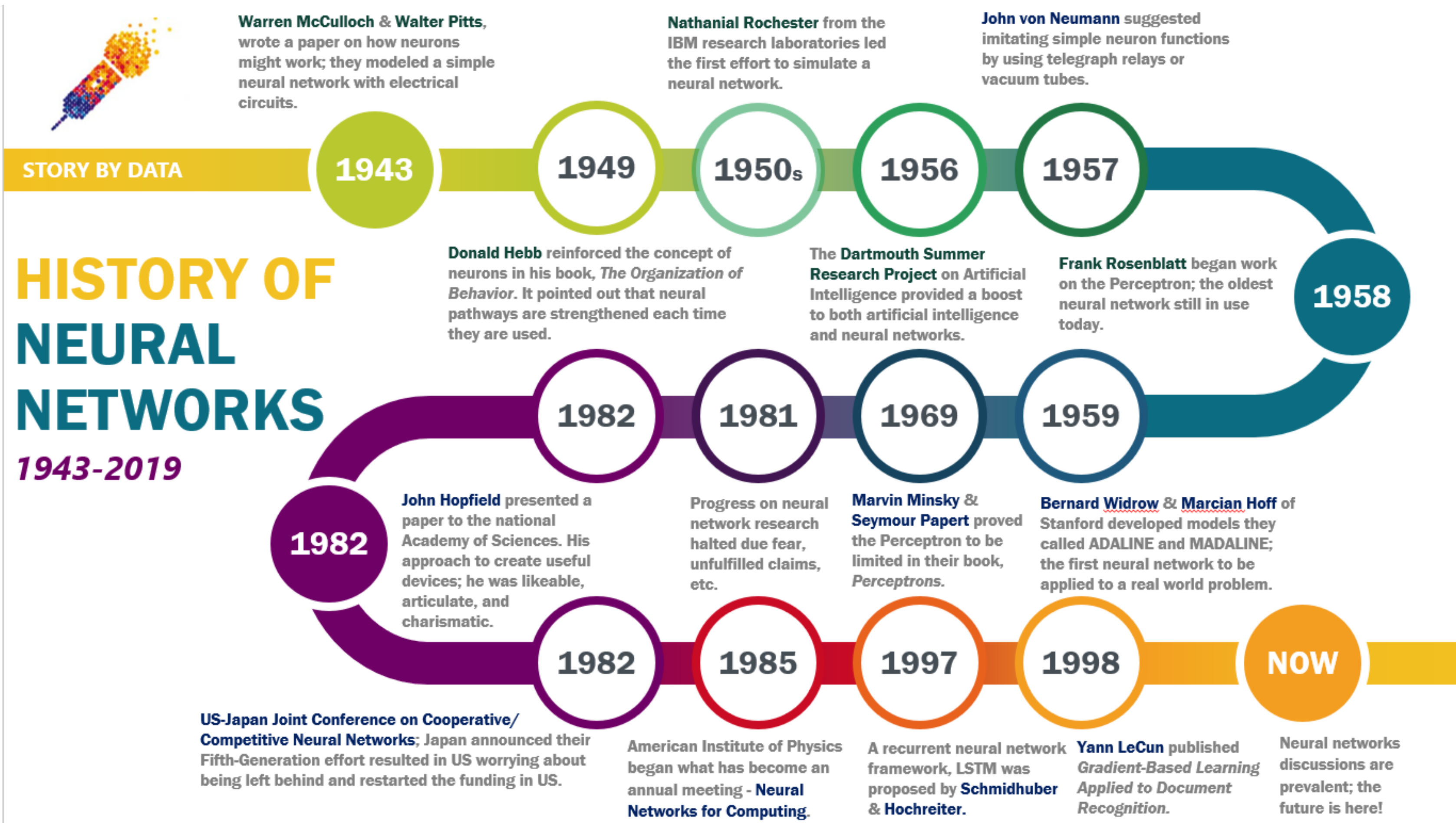
Neural networks


HISTORY OF NEURAL NETWORKS
1943-2019
STORY BY DATA
1943
Warren McCulloch & Walter Pitts, wrote a paper on how neurons might work; they modeled a simple neural network with electrical circuits.
1949
Donald Hebb reinforced the concept of neurons in his book, The Organization of Behavior. It pointed out that neural pathways are strengthened each time they are used.
1950s
Nathanial Rochester from the IBM research laboratories led the first effort to simulate a neural network.
1956
The Dartmouth Summer Research Project on Artificial Intelligence provided a boost to both artificial intelligence and neural networks.
1957
John von Neumann suggested imitating simple neuron functions by using telegraph relays or vacuum tubes.
1958
Frank Rosenblatt began work on the Perceptron; the oldest neural network still in use today.
1959
Bernard Widrow & Marcian Hoff of Stanford developed models they called ADALINE and MADALINE; the first neural network to be applied to a real world problem.
1969
Marvin Minsky & Seymour Papert proved the Perceptron to be limited in their book, Perceptrons.
1981
Progress on neural network research halted due fear, unfulfilled claims, etc.
1982
John Hopfield presented a paper to the national Academy of Sciences. His approach to create useful devices; he was likeable, articulate, and charismatic.
1982
US-Japan Joint Conference on Cooperative/ Competitive Neural Networks; Japan announced their Fifth-Generation effort resulted in US worrying about being left behind and restarted the funding in US.
1985
American Institute of Physics began what has become an annual meeting - Neural Networks for Computing.
1997
A recurrent neural network framework, LSTM was proposed by Schmidhuber & Hochreiter.
1998
Yann LeCun published Gradient-Based Learning Applied to Document Recognition.
NOW
Neural networks discussions are prevalent; the future is here!

Intuition: visual hierarchy in the brian

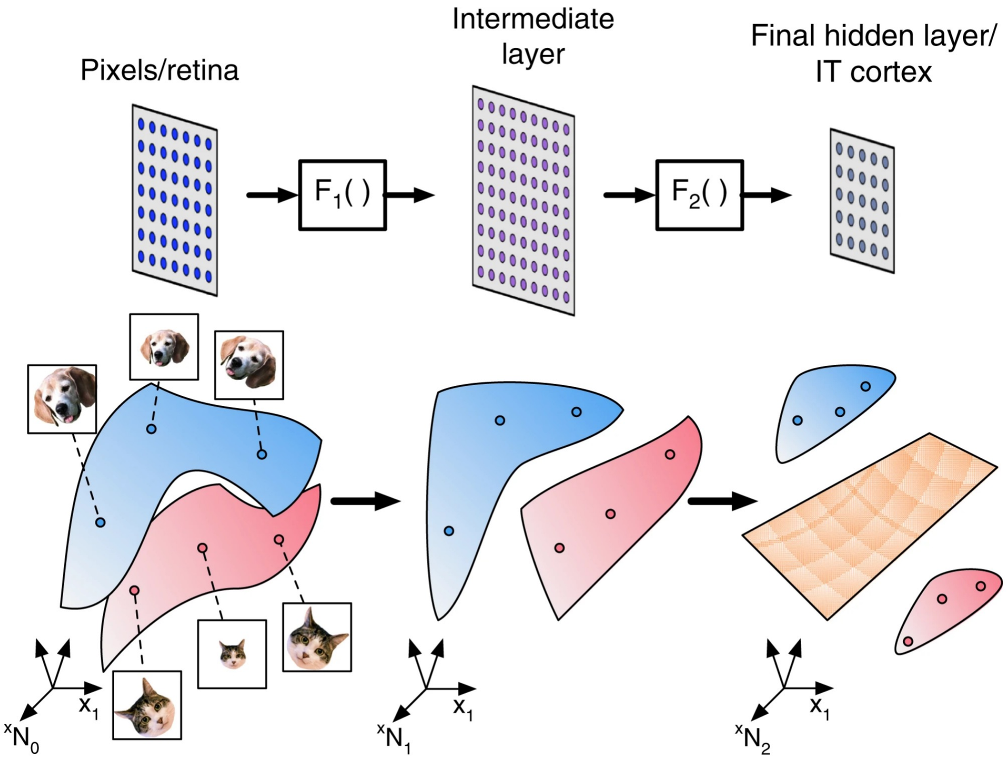
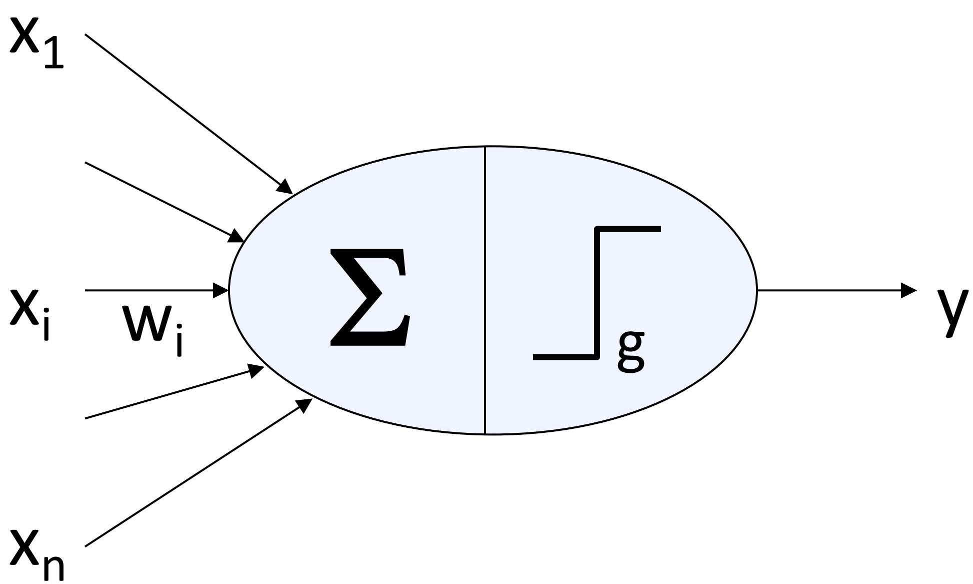


Illustration of three layers in a visual hierarchy where the population response of the first layer is mapped into intermediate layer by F_1 and into the last layer by F_2 (top). The transformation of per-stimuli responses is associated with changes in the geometry of the object manifold, the collection of responses to stimuli of the same object (colored blue for a 'dog' manifold and pink for a 'cat' manifold). Changes in geometry may result in transforming object manifolds which are not linearly separable (in the first and intermediate layers) into separable ones in the last layer (separating hyperplane, colored orange).

Cohen, Uri, et al. "Separability and geometry of object manifolds in deep neural networks." *Nature communications* 11.1 (2020): 1-13

Perceptrons can model different functions



The function $x_1 \wedge x_2 \wedge \neg x_3$?

Majority function ?

Adapted from K. Hauser's slide

Learning perceptrons

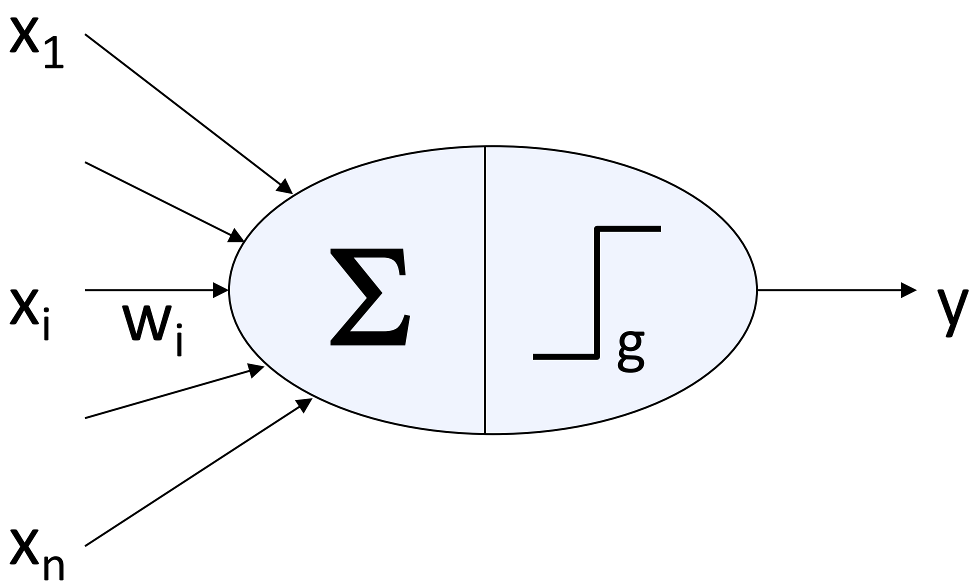
- How do we learn w?
  - Take derivative, set equal to 0, solve for w?
- Simple update rule:
  - Start with initial guess of w.
  - For each exemplar (x,y), and each weight w_i, update:

    $w_i \leftarrow w_i + \alpha\, x_i\, (y - g(w^T x))$

    (where y is either 0 or 1 and g() is either 0 or 1)
- Converges if data is linearly separable, but oscillates otherwise

Adapted from K. Hauser's slide

Perceptrons can model different functions



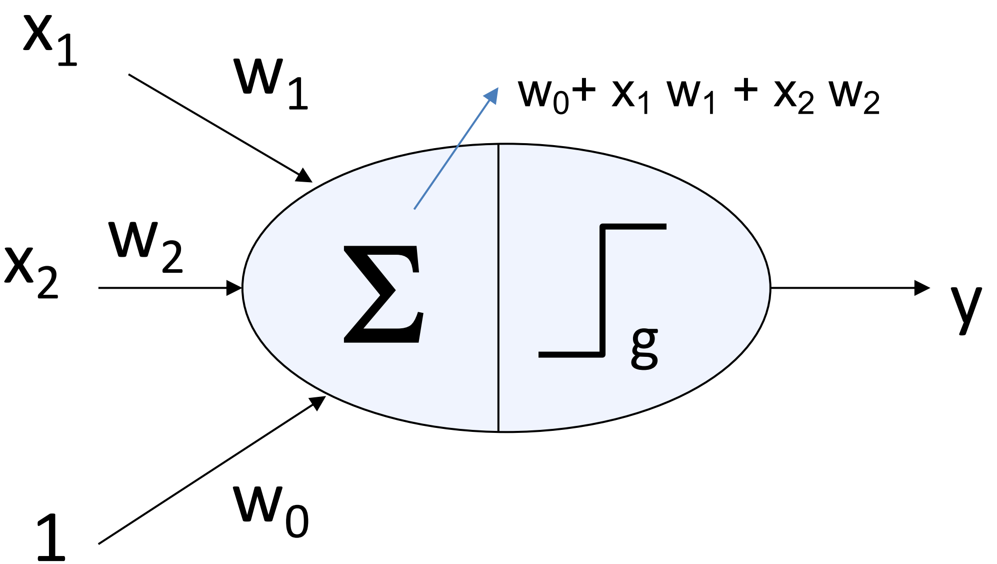
The function $x_1 \wedge x_2 \wedge \neg x_3$?

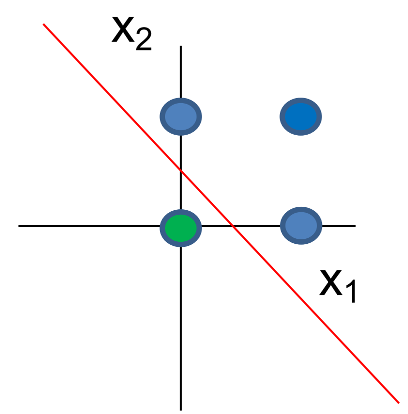
Majority function ?

XOR ?

Adapted from K. Hauser's slide

Perceptrons can model different functions





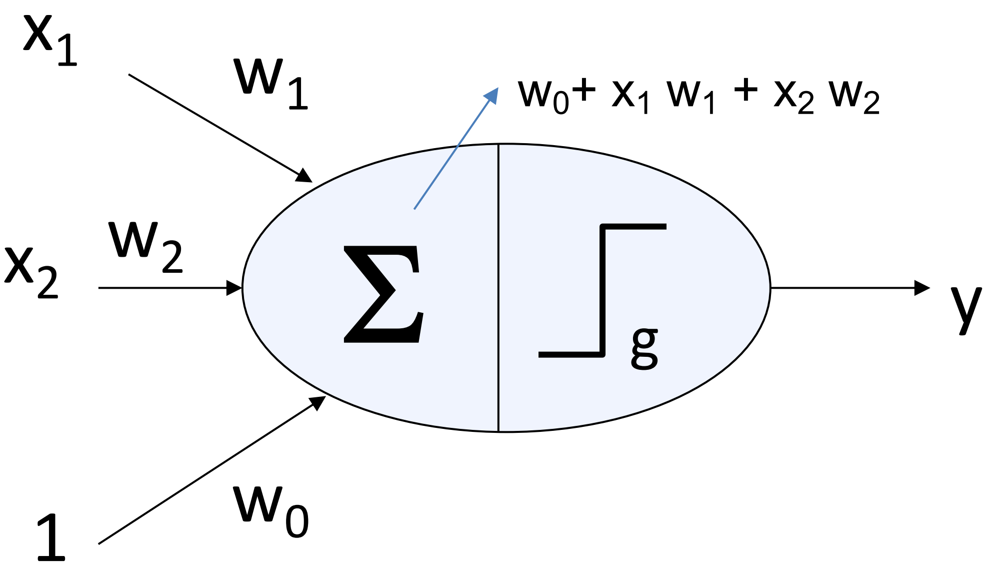
OR function

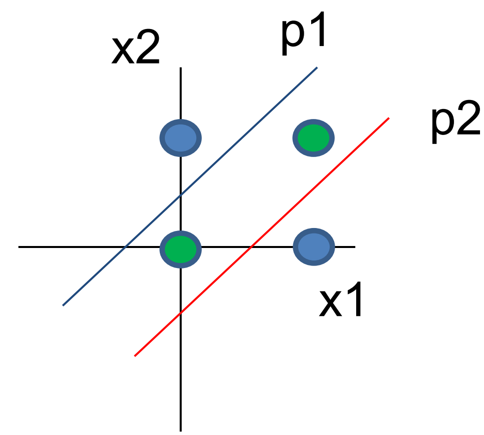
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

$w_0 = -0.5$
$w_1 = 1$
$w_2 = 1$

Adapted from K. Hauser's slide

Perceptrons can model different functions



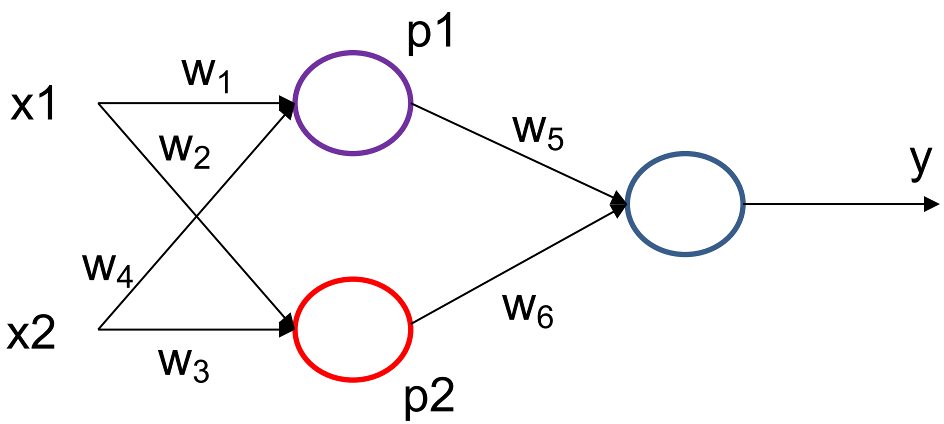


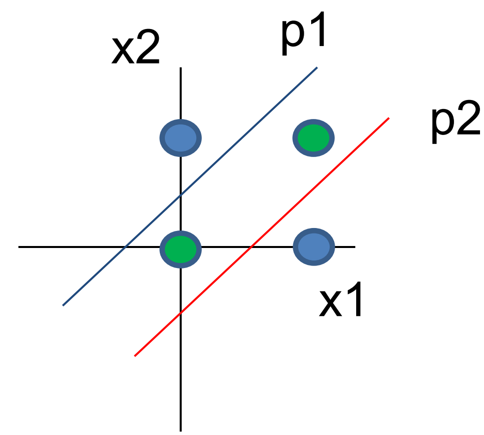
XOR ?

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Adapted from K. Hauser's slide

XOR with perceptrons





p1

x1	x2	p1
0	0	0
0	1	0
1	0	1
1	1	0

$w_1 = 1$
$w_2 = -1$

p2

x1	x2	p1
0	0	0
0	1	1
1	0	0
1	1	0

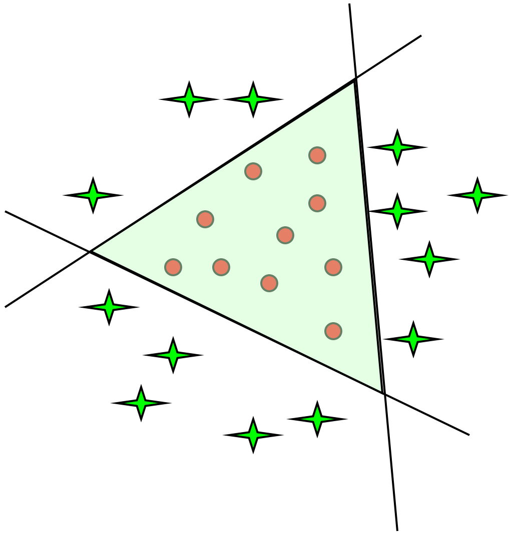
$w_3 = -1$
$w_4 = 1$

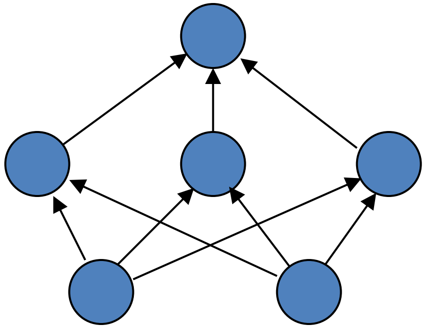
p3

x1	x2	p1	p2	y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

$w_5 = 1$
$w_6 = 1$

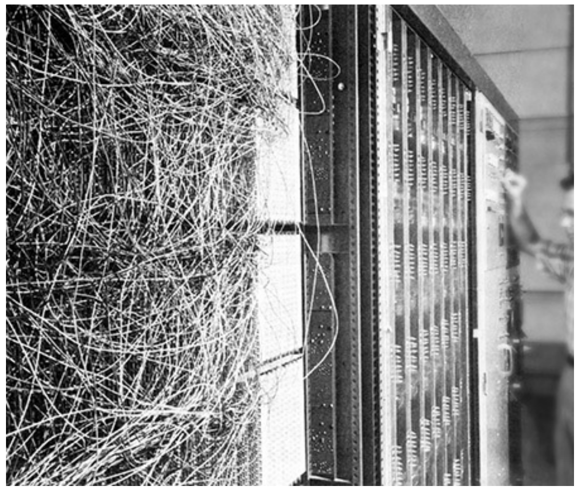
Multi-Layer Generalization

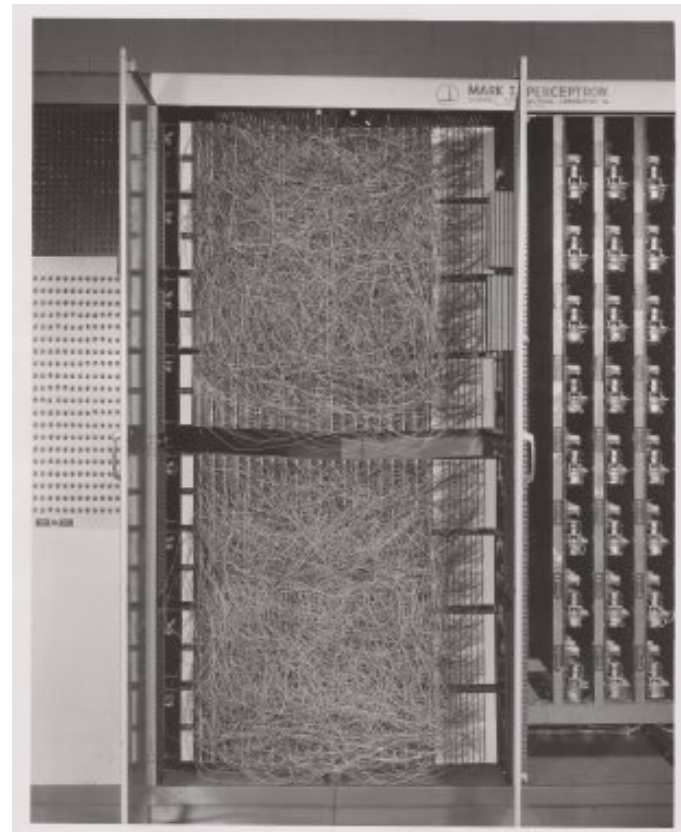




Slide credit: T. Martinez

Perceptrons

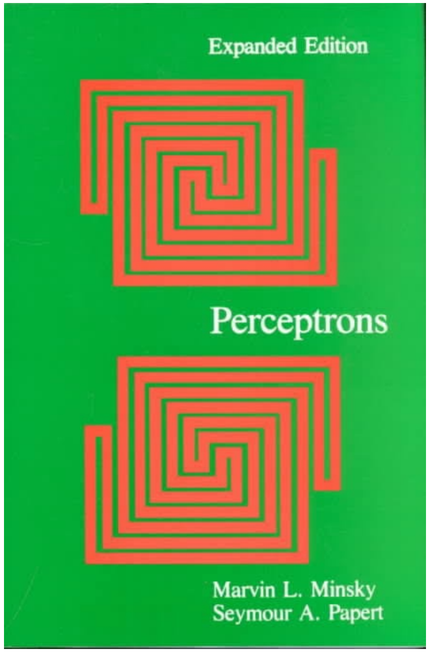




"the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

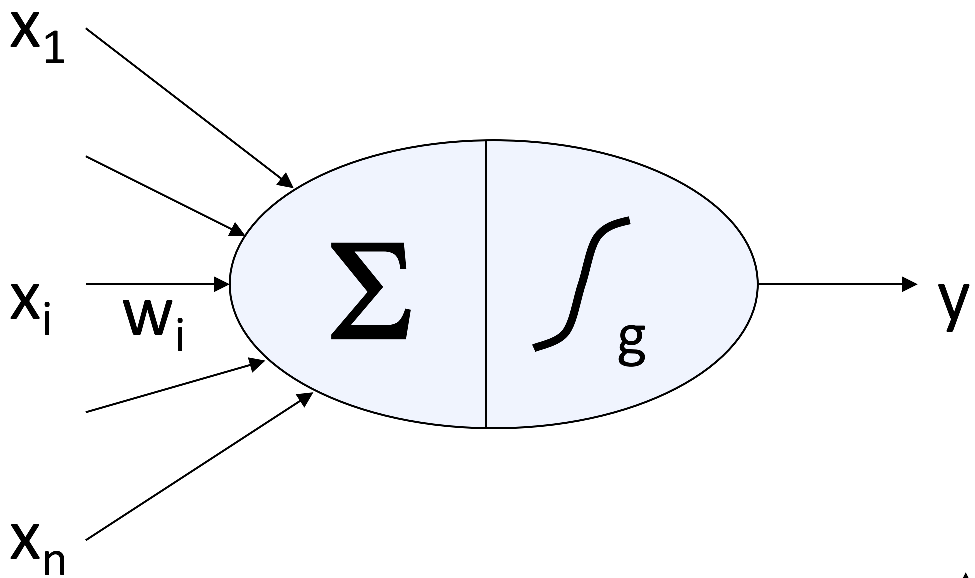
Frank Rosenblatt, 1958

Perceptrons



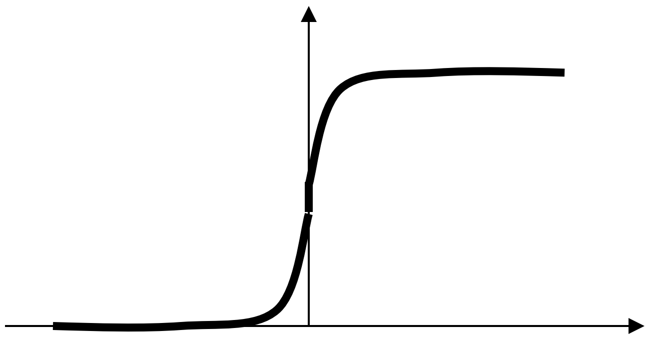
...until Minky & Papert showed they couldn't even learn XOR. (1969)

Unit (Neuron)



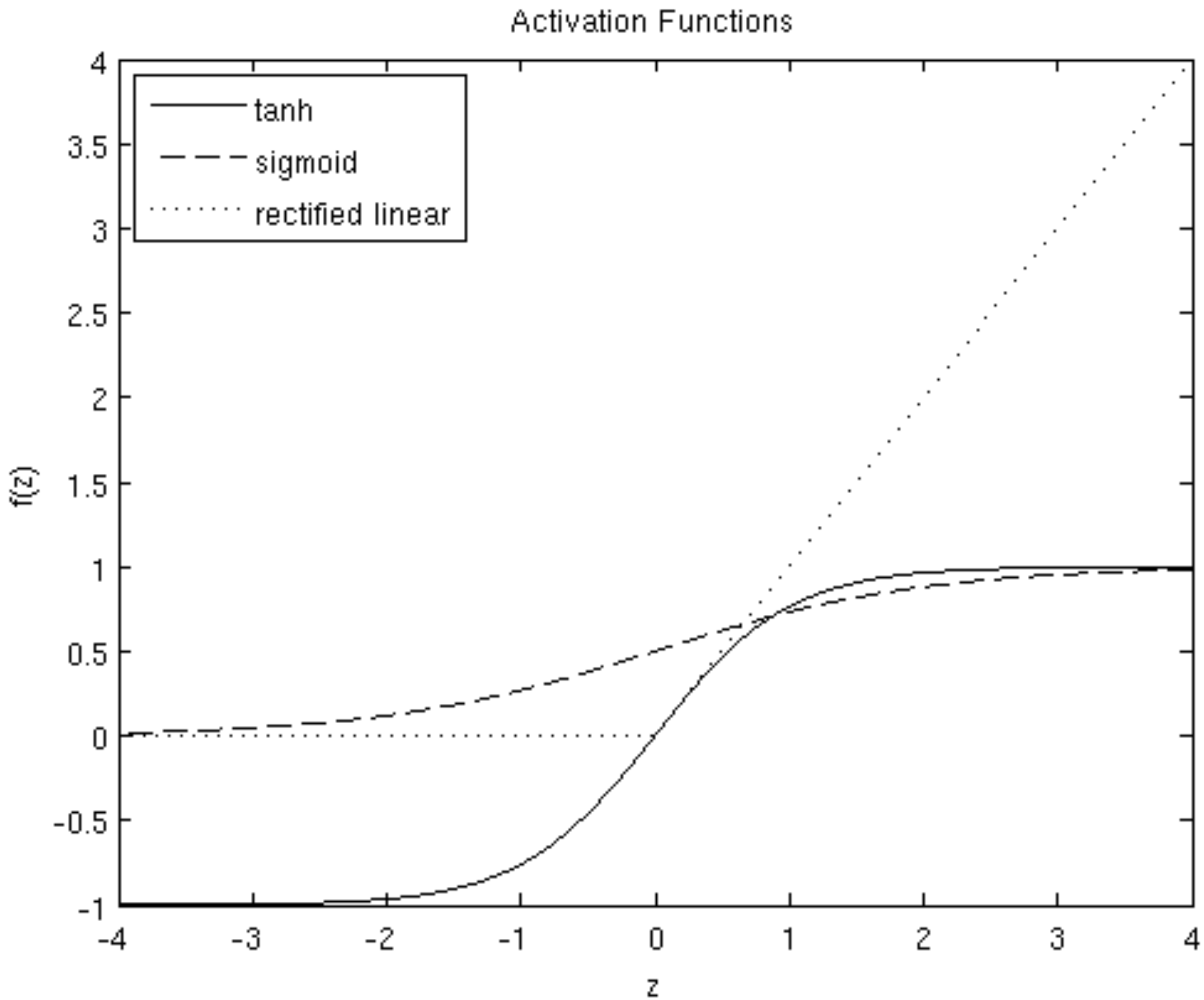
$y = g(\Sigma_{i=1,\ldots,n} w_i x_i)$

$g(u) = 1/[1 + \exp(-au)]$



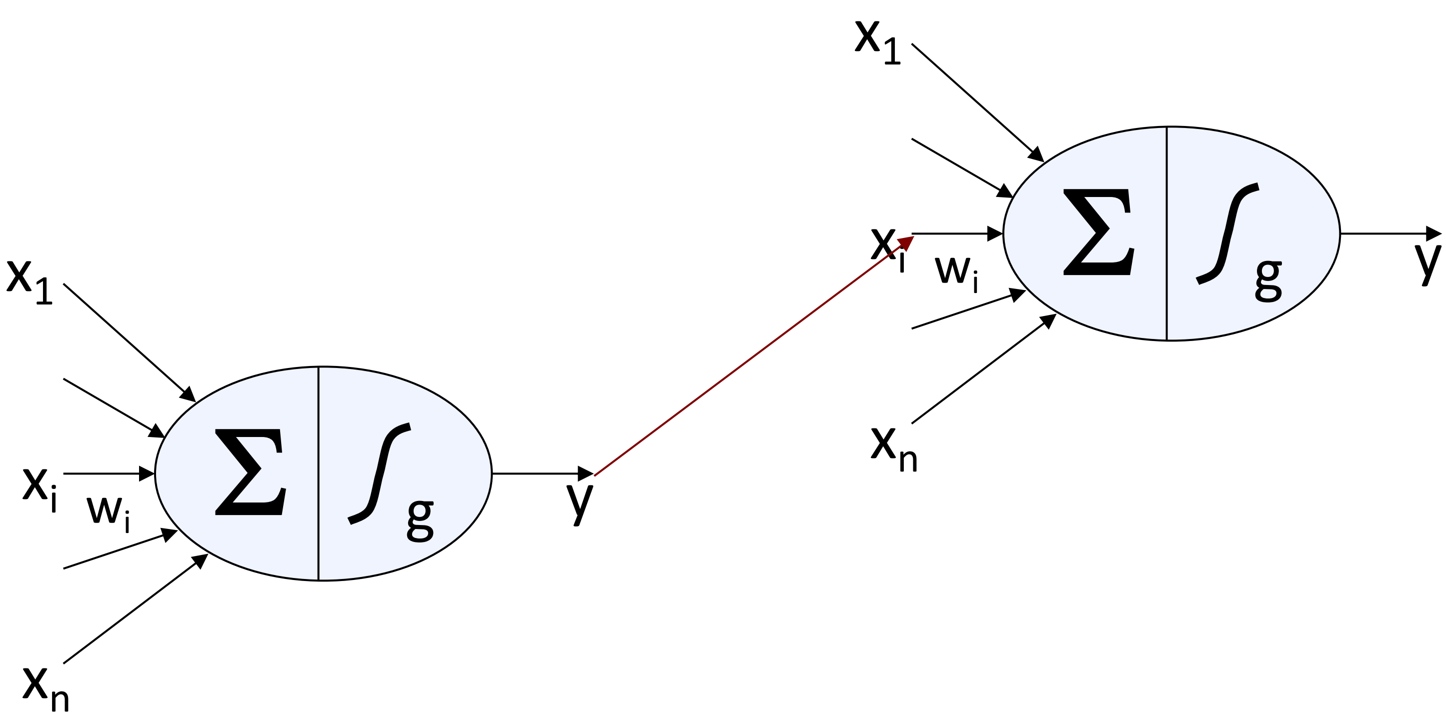
Adapted from K. Hauser's slide

Common activation functions



Neural Network

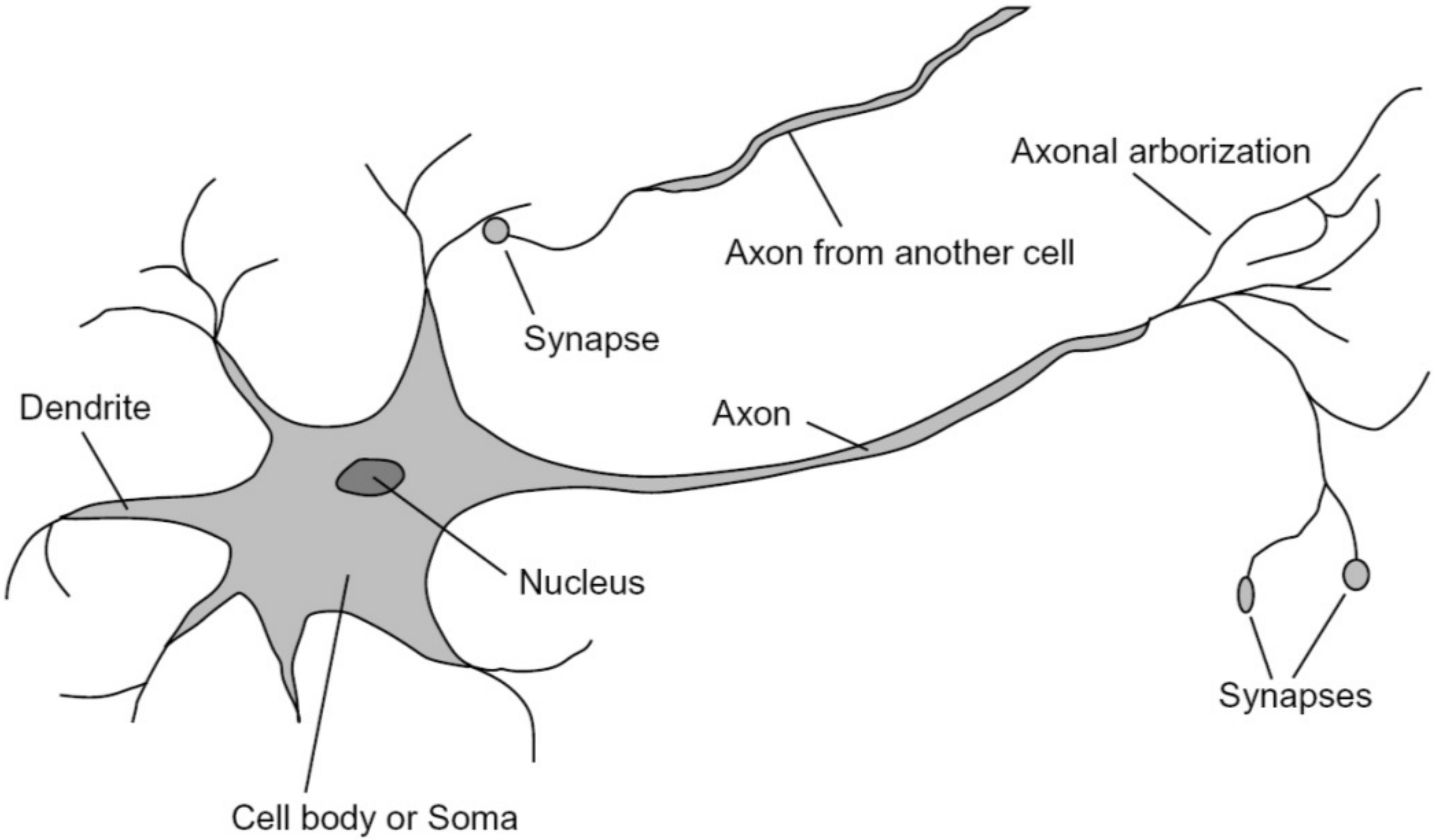
- Network of interconnected neurons



Acyclic (feed-forward) vs. recurrent networks

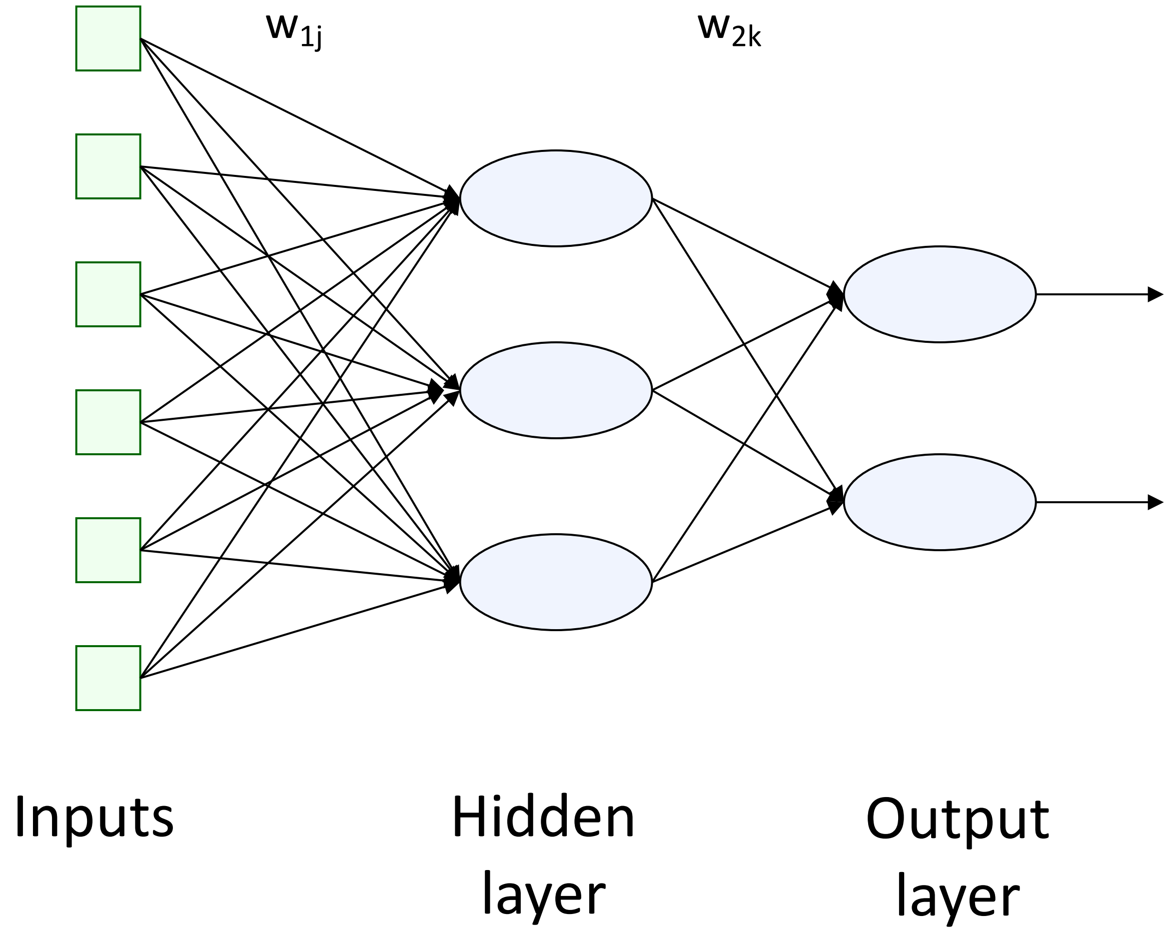
Adapted from K. Hauser's slide

Inspiration: Neuron cells



Slide credit: Rob Fergus

Two-Layer Feed-Forward Neural Network



Adapted from K. Hauser's slide

Networks with hidden layers

- Can represent XORs, other nonlinear functions
- Many, many variants:
  - Different network structures
  - Different activation functions
  - Etc…
- As the number of hidden units increases, the network's capacity to learn more complicated functions also increases

- *How to train hidden layers?*

Adapted from K. Hauser's slide

Next Class

- Training Neural Networks