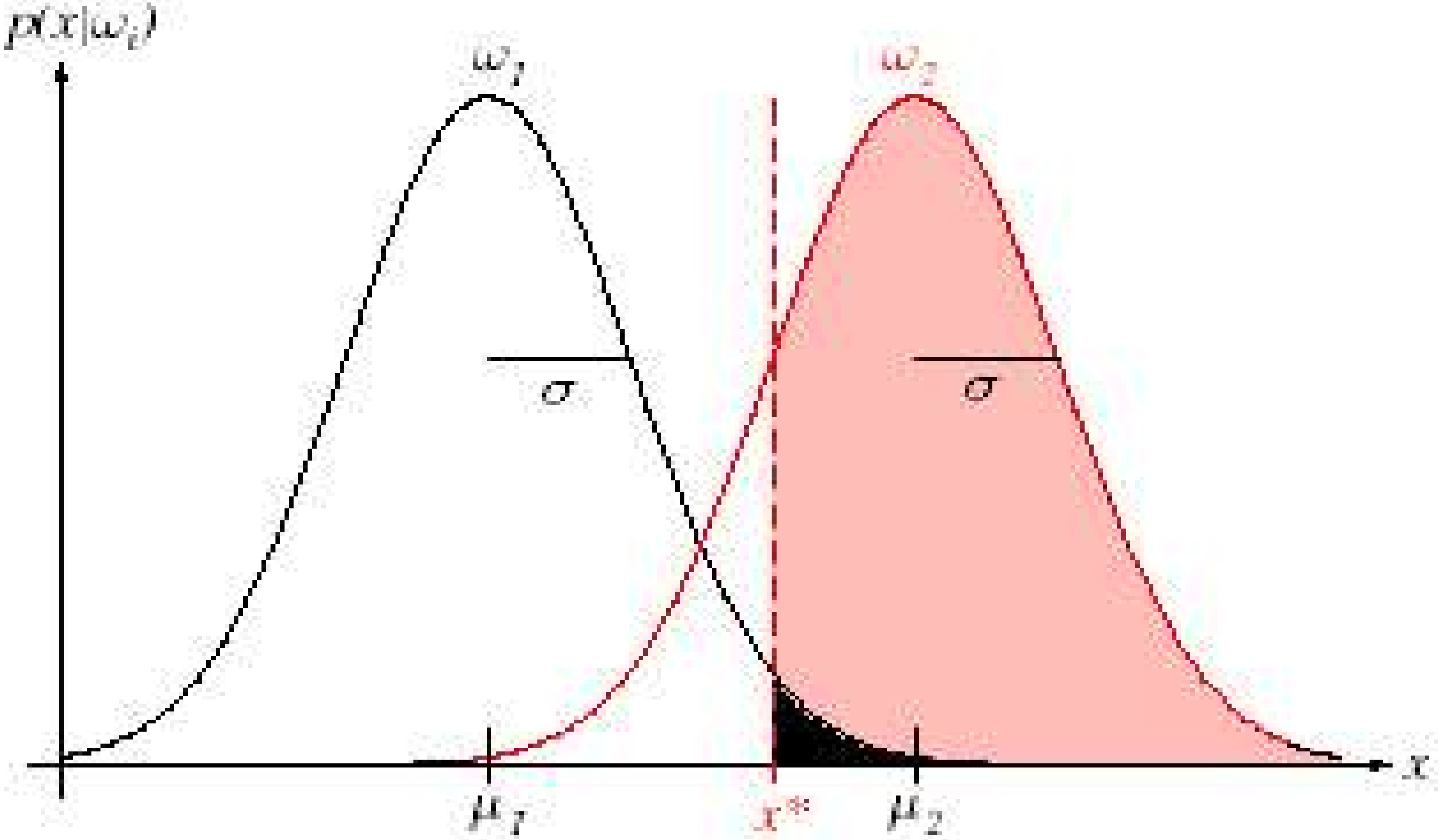
CSC421 Intro to Artificial Intelligence

UNIT 32: Instance-based Learning and Neural Networks

Outline

- Nearest-Neighbhor models
- Kernel Models
- Neural Networks
- Machine Learning using Weka

Classification using single Gaussian



Nearest-Neighbor

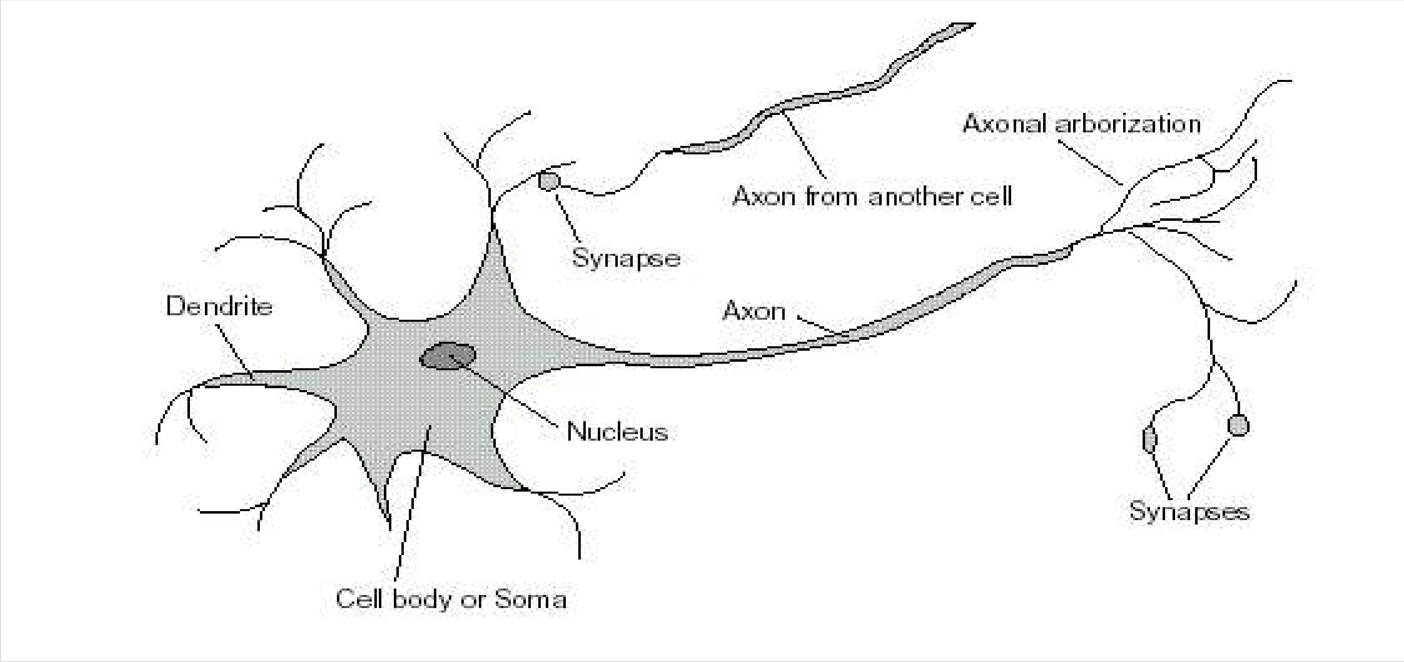
- Key idea: properties of any particular input point x are likely to be similar to the points in the neighborhood of x
- A form of local density estimation
  - Just enough to fit k points (typically 3-5)
  - Distacnes: Euclidean, Standarize + Euclidean, Mahalanobis, Hamming (discrete features) = #features in which points differ
- Simple to implement, good performance but doesn't scale well

Kernel Models

- Each training distance generates a little density function – a kernel function
- Density estimate = normalized sum of all the little kernel functions
- $P(x) = 1/N \sum K(x, x_i)$
- Kernel function depends only on distance
- Typical choice Gaussian (Radial-Basis Functions)
- Uses all instances

Neural Networks

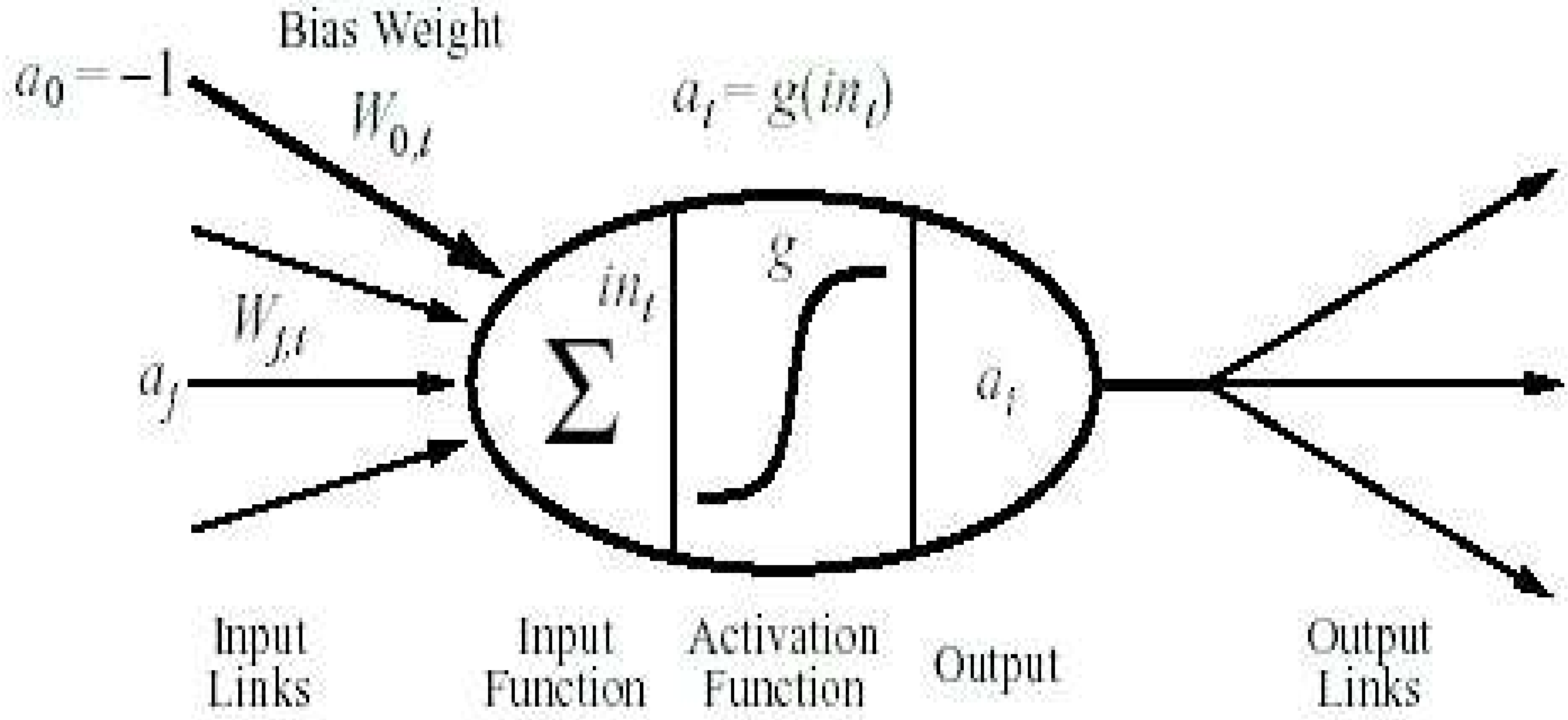
- Biological inspiration but more of a simplification than a real model
- Distributed computation, noisy inputs, learning, regression



McCulloch-Pitts Unit

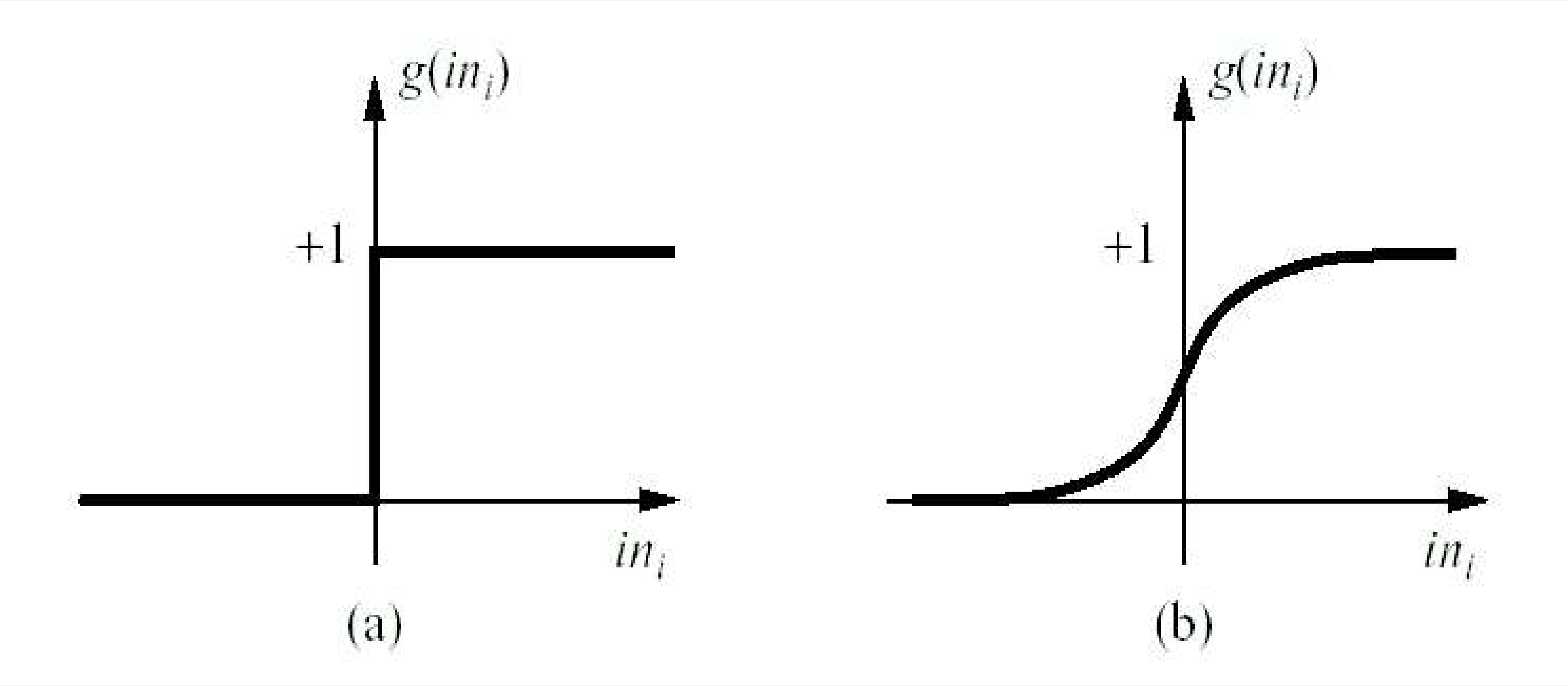
- Output is a "squashed" weighted sum of input

$$a_i \leftarrow g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$



Activation Functions

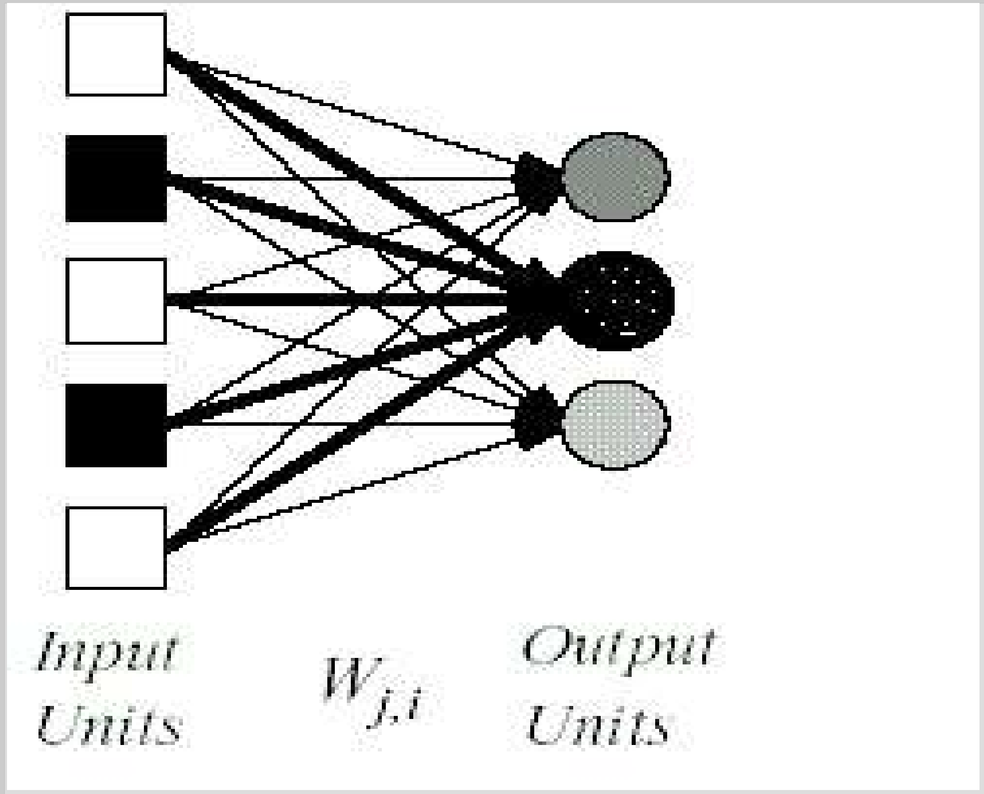
- Step function
- Sigmoid $1 / (1 + e^{-x})$



Network Structures

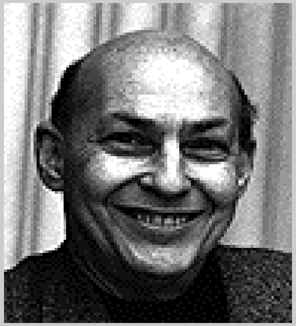
- Feed-forward networks
  - Single-layer perceptrons
  - Multiple-layer perceptrons
- Feedforward networks implement functions, have no internal state
- Recurrent Networks
  - Hopfield networks (holographic associative memory)
  - Boltzmann machines
  - Have internal states can oscillate

Single Layer Perceptron



Output units operate separately
(no shared weights)

Learning by adjusting weights to reduce error

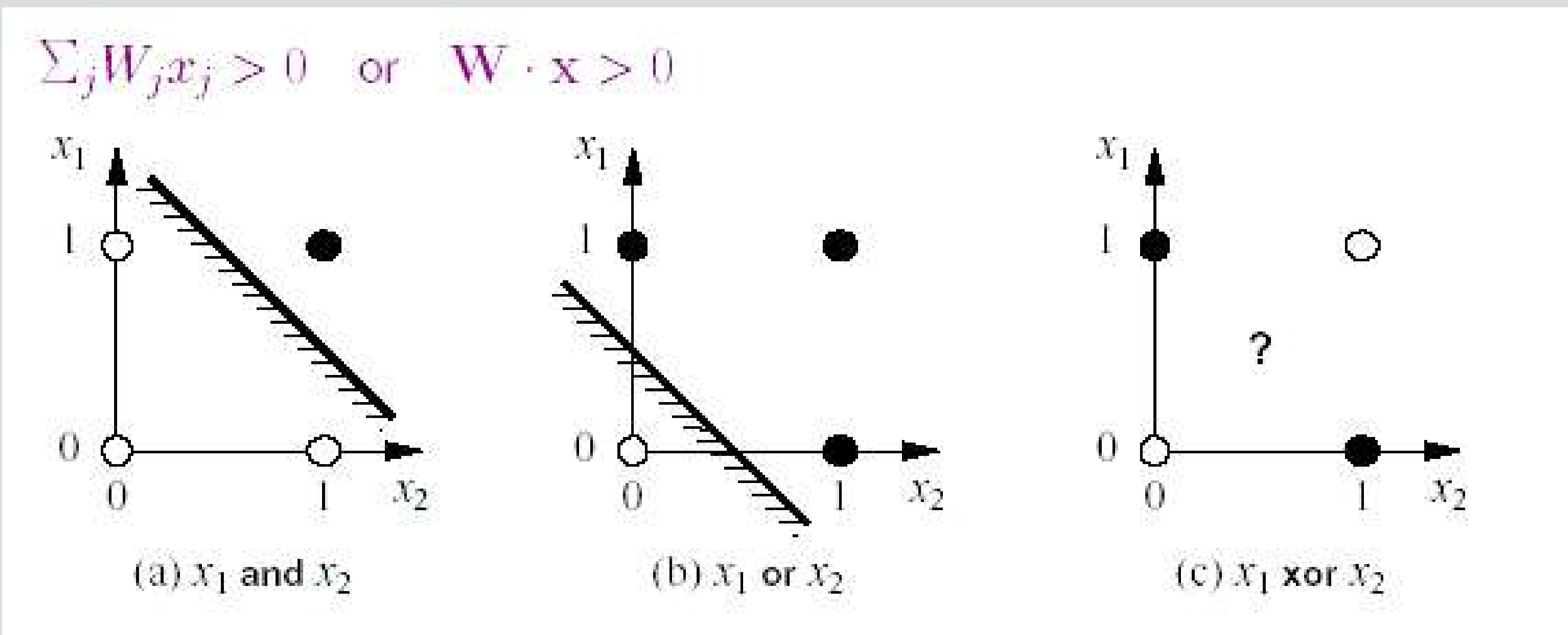


A bit of history

- Rosenblatt (1957-1960) – Cornell first computer that could “learn” by trial & error
- Perceptrons – brain, learning, lots of hype
- Nemesis: Marvin Minsky (MIT)
  - 1969 Perceptrons – proof that simple 3-layer perceptron can not learn the XOR function – postulation that multi-layer can not (turned out not to be true)
  - 10 years of no funding for ANN
  - Later retracted his position

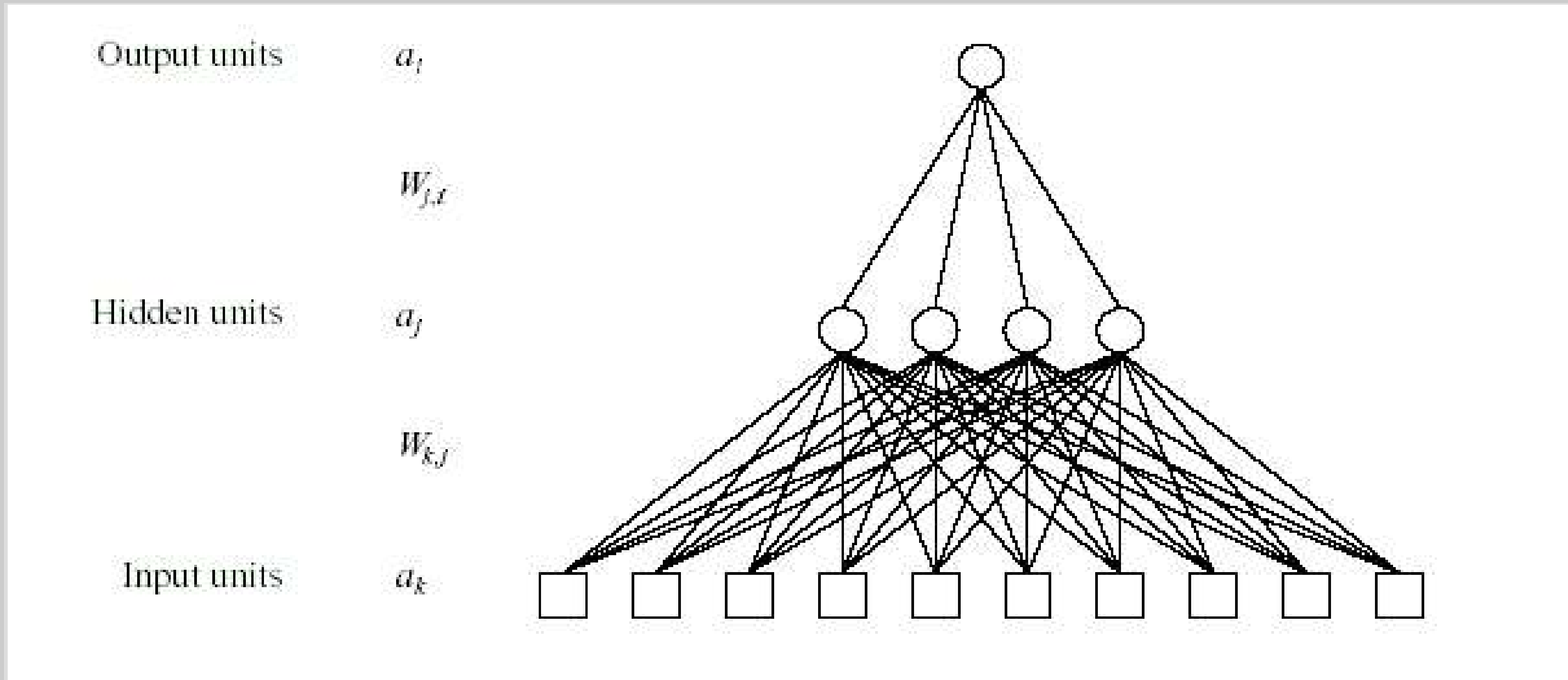
Expressiveness of perceptrons

- Perceptrons can represent:
  - AND, OR, NOT, majority but not XOR
- Represents a linear separator in input space



(a) x_1 and x_2 (b) x_1 or x_2 (c) x_1 xor x_2

MultiLayer Perceptrons



Layers are usually fully connected
Number of hidden units choosen empirically
Can represent any function