# Neural Networks

# The Basic Idea

"The strategy has been to develop simplified mathematical models of brain-like systems and then to study these models to understand how various computational problems can be solved by such devices."

D. E. Rumelhart, B. Widrow, M. A. Lehr,
The Basic Ideas in Neural Networks
*Communications of the ACM*, March, 1994

# Neural Computing

- von Neumann architectures
  - serial
  - powerful processors (measured in MIPS)
- Architecture of the brain
  - massively parallel
  - slow processors (10 “instructions” per second)

# Number of processors (neurons)

| Organism | Neurons | Comments |
|---|---|---|
| trees | | |
| c. elegans (worm) | | |
| aphysia (snail) | | |
| mouse | | |
| human | | |

# Arguments for the approach

- Some tasks have resisted automation with traditional architectures
  - vision, speech, and other tasks with high volumes of information
- Fault tolerance, graceful degradation
  - every day a few neurons die

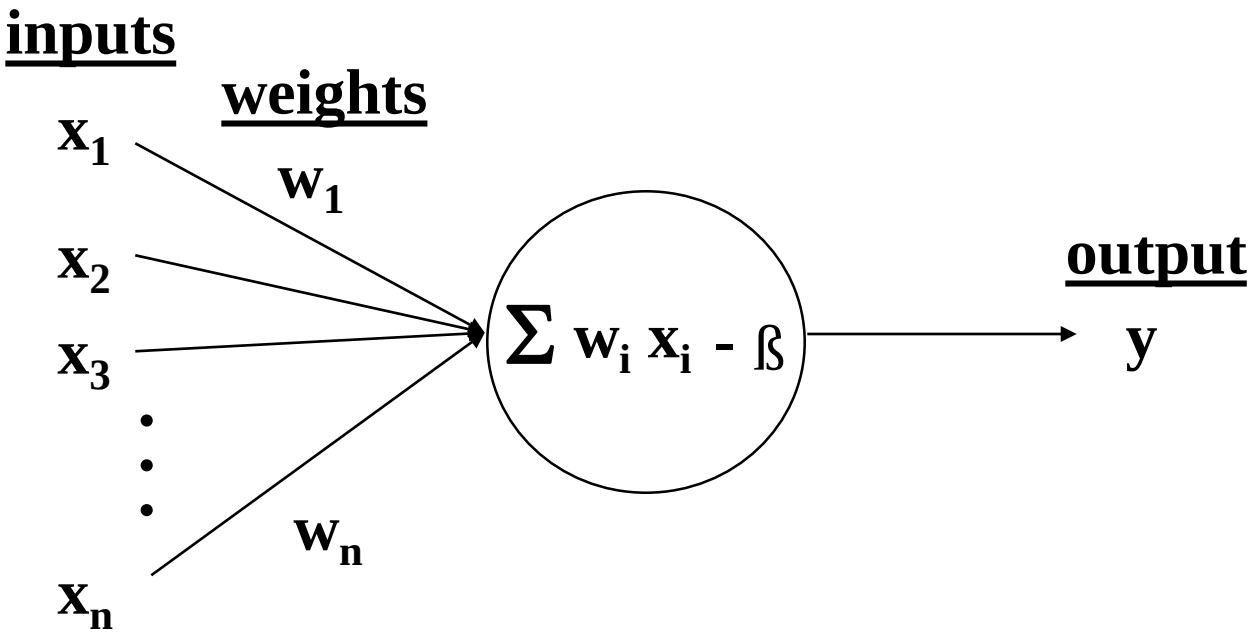
# A Brief History

- McCulloch & Pitts, 1943
  - brain as a *computational* organism
- Rosenblatt, 1950s
  - model of neurons called perceptrons
- Minsky & Papert, 1969
  - limitations of perceptrons
  - intelligence is symbol processing
  - result: funding dries up
- Exploding interest, 1980s
  - backpropagation learning rule
  - fast computers for simulating networks
- Hinton, Osindero, and Teh, 2006
  - a fast learning algorithm for deep belief nets

# Modeling a neuron

- Important features:
  - output of a neuron is either on (“fires”) or off
  - output of a neuron depends only on its inputs
  - inputs come in different strengths, depending on the efficiency of the synapses
  - learning changes the efficiency of the synaptic junctions

# A typical unit

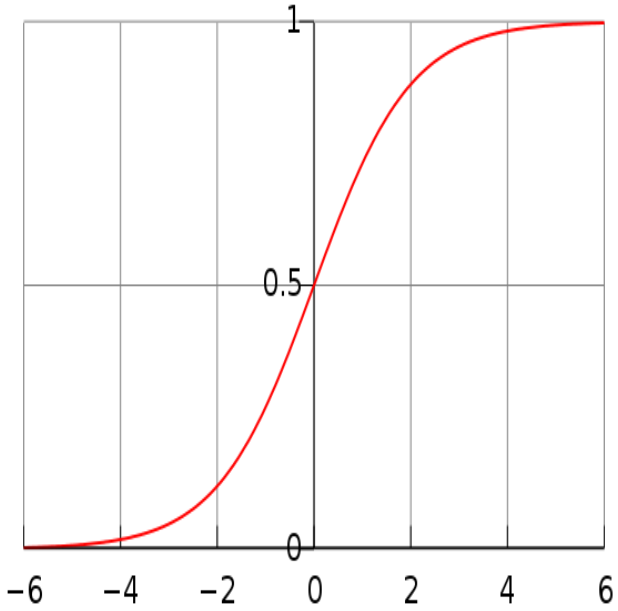



$$y = f\left(\sum w_i x_i - ß\right)$$

# Thresholding functions

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

**step function**



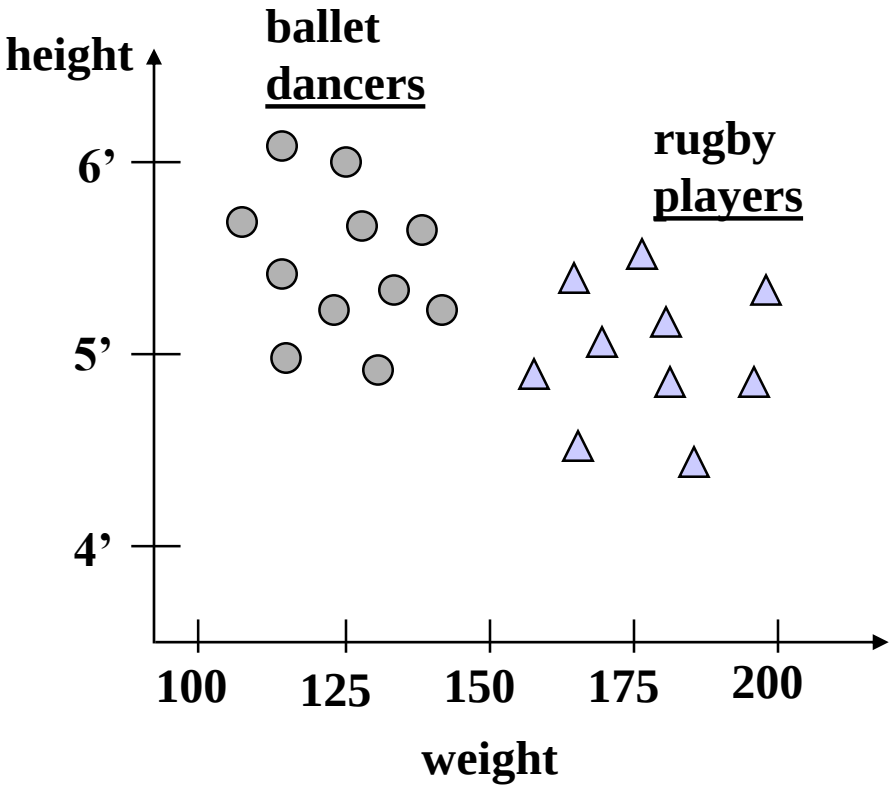

**sigmoid function**

# Thought experiments

Replace a neuron with a simple embedded processor

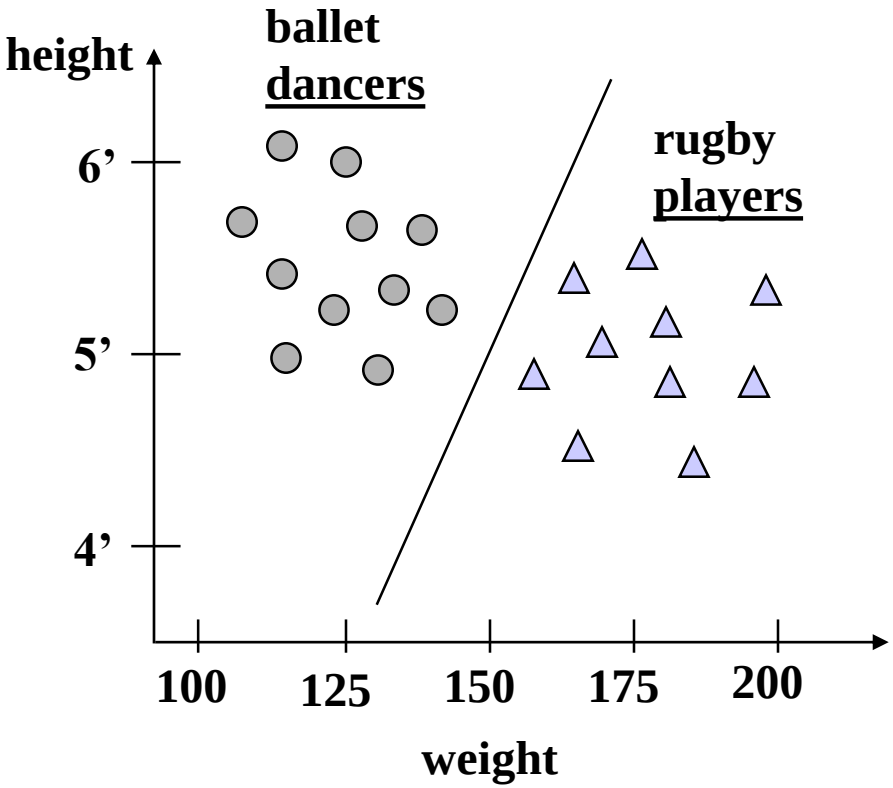
Replace a neuron with a simple wireless device

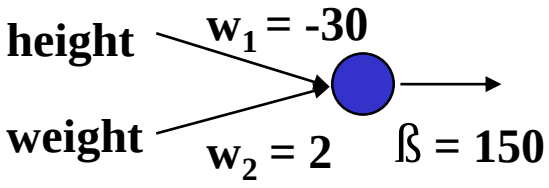
# Example training data




| Training Data | |
|---|---|
| **input** | **output** |
| 125, 5’3” | ballet |
| 175, 5’6” | rugby |
| 115, 6’1” | ballet |
| … | |

# A single layered network




**Compute:**

**$w_1$(height) +**

**$w_2$(weight) - ß**

**Apply step function**

**Output is**

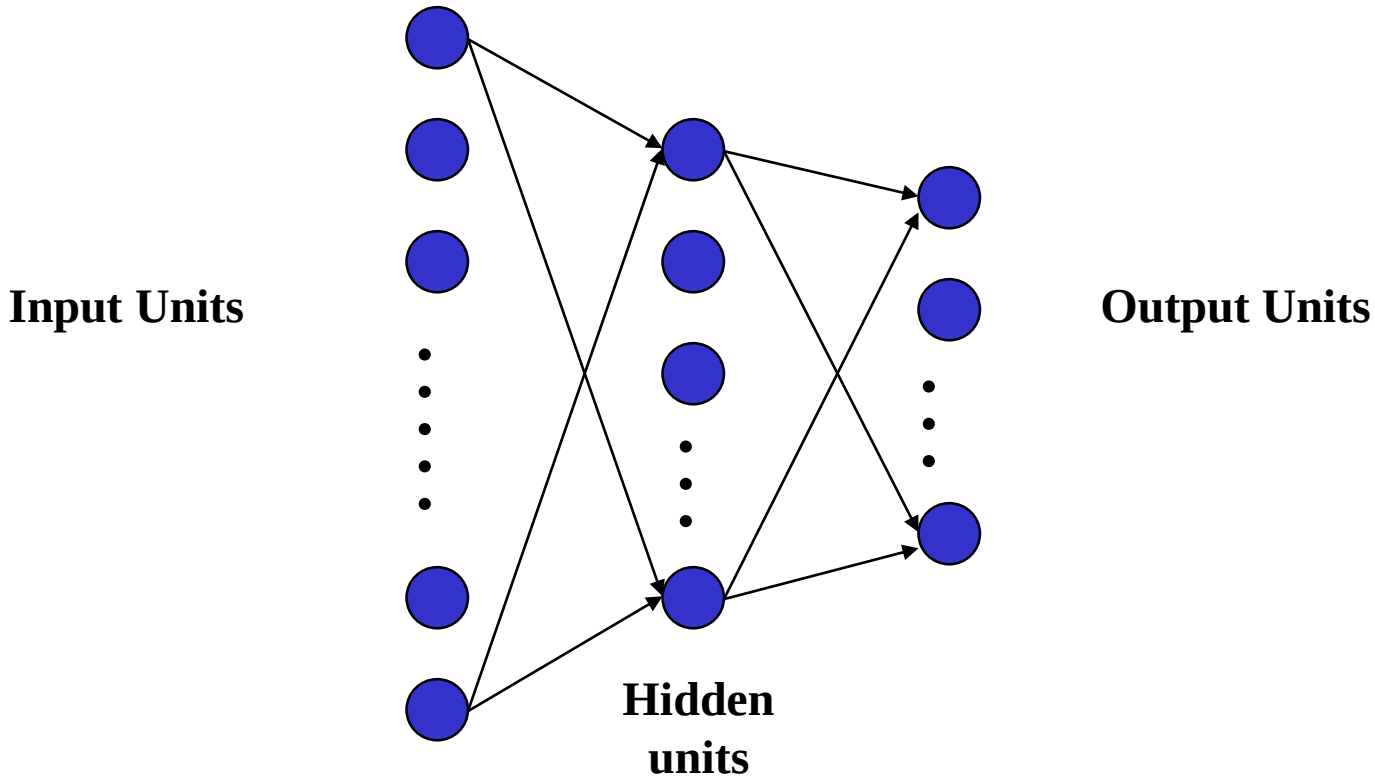
**0 if ballet dancer**

**1 if rugby player**

# Multi-layered, feed-forward networks

- Multi-layered:
  - a layer of input units
  - one or more layers of hidden units
  - a layer of output units
- Feed-forward:
  - information flows from input layer, to hidden layer, to output layer
  - the activity of a unit cannot influence its own inputs (no loops)
- Each unit uses the sigmoid threshold function

# A three-layered network



# Training a neural network

- The networks must learn to solve a problem
  - learn the weights which allow it to solve the problem
  - programming the weights by hand is impossible
- Supervised learning:
  - network is provided with a set of training examples
  - each example is an input/output pair
  - modifies its weights to give the right output for an input

# Supervised learning

Initialize the network with random weights

Repeat until the error is acceptably small

1. present an example
2. calculate actual output
3. calculate error: difference between desired output and actual output
4. if there is error, determine which weights were at fault and adjust those weights

# Backpropagation learning rule

- The problem is to determine which weights in the entire network were at fault for the error
- Backpropagation:
  - most popular learning rule for updating the weights in a network
  - updates a weight in proportion to the degree to which changes in the weight will lead to changes in the error

# Backpropagation learning algorithm