Introduction to Neural Networks

Chapter 1
Introduction

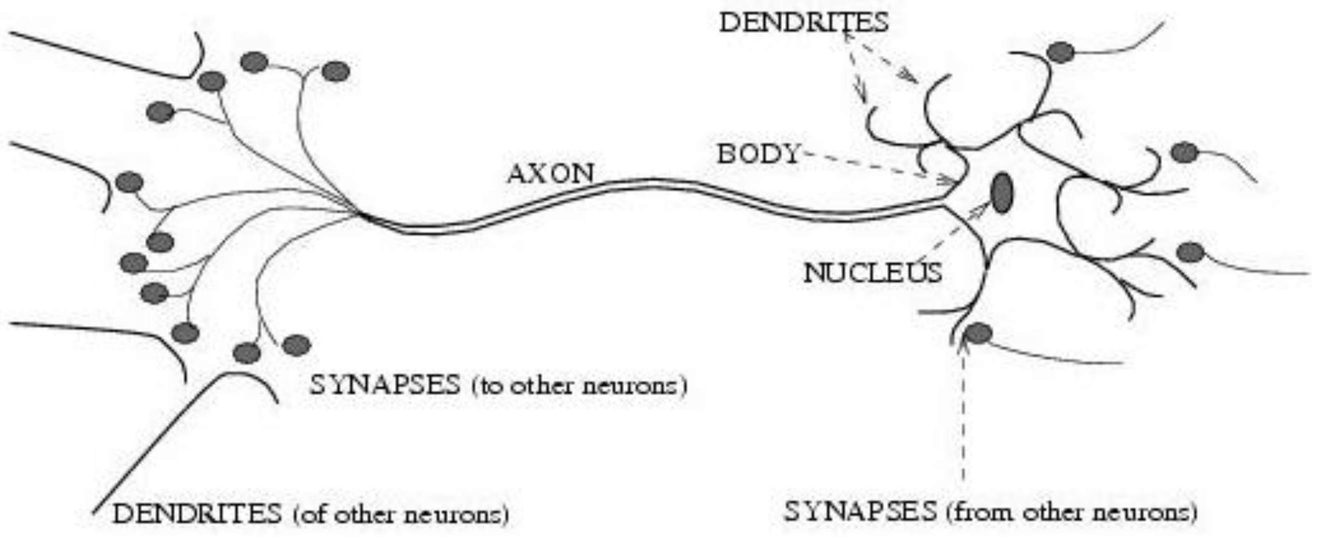
Introduction

- **Why ANN**
  - Some tasks can be done easily (effortlessly) by humans but are hard by conventional paradigms on Von Neumann machine with algorithmic approach
    - Pattern recognition (old friends, hand-written characters)
    - Content addressable recall
    - Approximate, common sense reasoning (driving, playing piano, baseball player)
  - These tasks are often ill-defined, experience based, hard to apply logic

Introduction

Von Neumann machine	Human Brain
• One or a few high speed (ns) processors with considerable computing power	• Large # (10^{11}) of low speed processors (ms) with limited computing power
• One or a few shared high speed buses for communication	• Large # (10^{15}) of low speed connections
• Sequential memory access by address	• Content addressable recall (CAM)
• Problem-solving knowledge is separated from the computing component	• Problem-solving knowledge resides in the connectivity of neurons
• Hard to be adaptive	• Adaptation by changing the connectivity

- **Biological neural activity**



  – Each neuron has a *body*, an *axon*, and many *dendrites*
    - Can be in one of the two states: *firing* and *rest.*
    - Neuron fires if the total incoming stimulus exceeds the threshold
  – *Synapse*: thin gap between axon of one neuron and dendrite of another.
    - Signal exchange
    - Synaptic strength/efficiency

Introduction

- **What is an (artificial) neural network**
  - A set of **nodes** (units, neurons, processing elements)
    - Each node has input and output
    - Each node performs a simple computation by its **node function**
  - **Weighted connections** between nodes
    - Connectivity gives the structure/architecture of the net
    - What can be computed by a NN is primarily determined by the connections and their weights
  - A very much simplified version of networks of neurons in animal nerve systems

Introduction

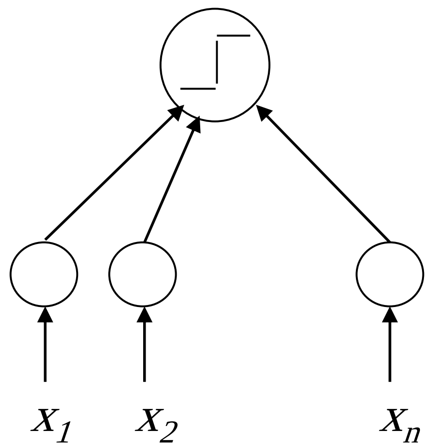
ANN	Bio NN
• **Nodes**	• **Cell body**
– input	– signal from other neurons
– output	– firing frequency
– node function	– firing mechanism
• **Connections**	• **Synapses**
– connection strength	– synaptic strength

- Highly parallel, simple local computation (at neuron level) achieves global results as emerging property of the interaction (at network level)
- Pattern directed (meaning of individual nodes only in the context of a pattern)
- Fault-tolerant/graceful degrading
- Learning/adaptation plays important role.

History of NN

- **Pitts & McCulloch (1943)**
  - First mathematical model of biological neurons
  - All Boolean operations can be implemented by these neuron-like nodes (with different threshold and excitatory/inhibitory connections).
  - Competitor to Von Neumann model for general purpose computing device
  - Origin of automata theory.
- **Hebb (1949)**
  - Hebbian rule of learning: increase the connection strength between neurons i and j whenever both i and j are activated.
  - Or increase the connection strength between nodes i and j whenever both nodes are simultaneously ON or OFF.

History of NN



- **Early booming (50's – early 60's)**
  - Rosenblatt (1958)
    - Perceptron: network of threshold nodes for pattern classification
      Perceptron learning rule
    - Percenptron convergence theorem:
      everything that can be represented by a perceptron can be learned
  - Widow and Hoff (1960, 19062)
    - Learning rule based on gradient descent (with differentiable unit)
  - Minsky's attempt to build a general purpose machine with Pitts/McCullock units

History of NN

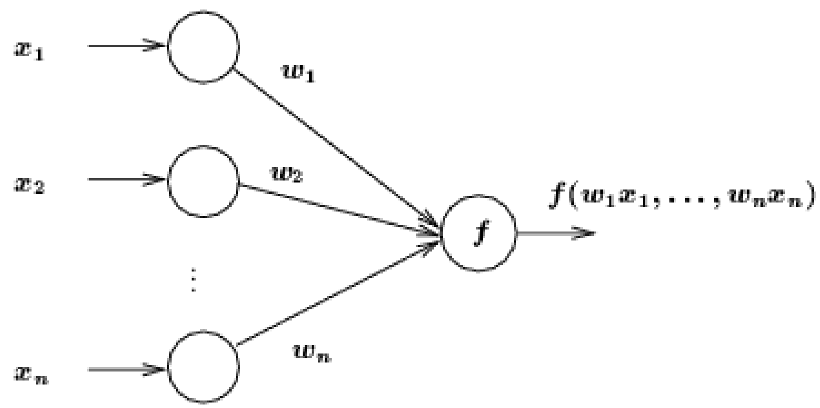
- **The setback (mid 60's – late 70's)**
  - Serious problems with perceptron model (Minsky's book 1969)
    - Single layer perceonptrons cannot represent (learn) simple functions such as XOR
    - Multi-layer of non-linear units may have greater power but there is no learning rule for such nets
    - Scaling problem: connection weights may grow infinitely
  - The first two problems overcame by latter effort in 80's, but the scaling problem persists
  - Death of Rosenblatt (1964)
  - Striving of Von Neumann machine and AI

History of NN

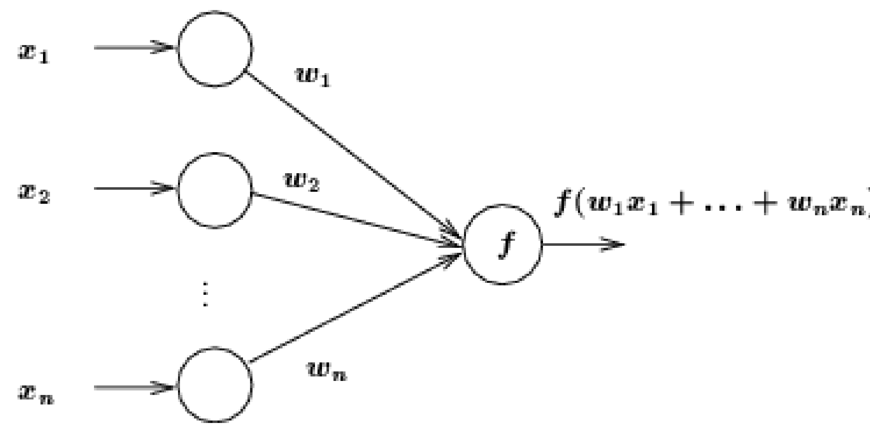
- **Renewed enthusiasm and flourish (80's – present)**
  - New techniques
    - Backpropagation learning for multi-layer feed forward nets (with non-linear, differentiable node functions)
    - Thermodynamic models (Hopfield net, Boltzmann machine, etc.)
    - Unsupervised learning
  - Impressive application (character recognition, speech recognition, text-to-speech transformation, process control, associative memory, etc.)
  - Traditional approaches face difficult challenges
  - Caution:
    - Don't underestimate difficulties and limitations
    - Poses more problems than solutions

ANN Neuron Models

- Each node has one or more inputs from other nodes, and one output to other nodes
- Input/output values can be
  - Binary {0, 1}
  - Bipolar {-1, 1}
  - Continuous
- All inputs to one node come in at the same time and remain activated until the output is produced
- Weights associated with links
- $f(net)$ is the node function
  $net = \sum_{i=1}^{n} w_i x_i$ is most popular



General neuron model



Weighted input summation

Node Function

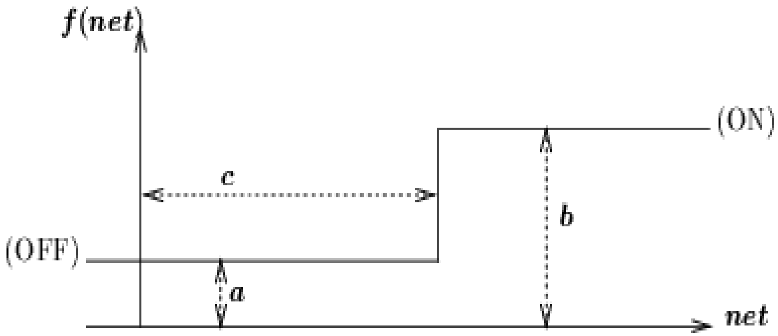
- Identity function: $f(net) = net.$
- Constant function: $f(net) = c.$
- Step (threshold) function

$$f(\text{net}) = \begin{cases} a & \text{if net} < c \\ b & \text{if net} > c \end{cases}$$

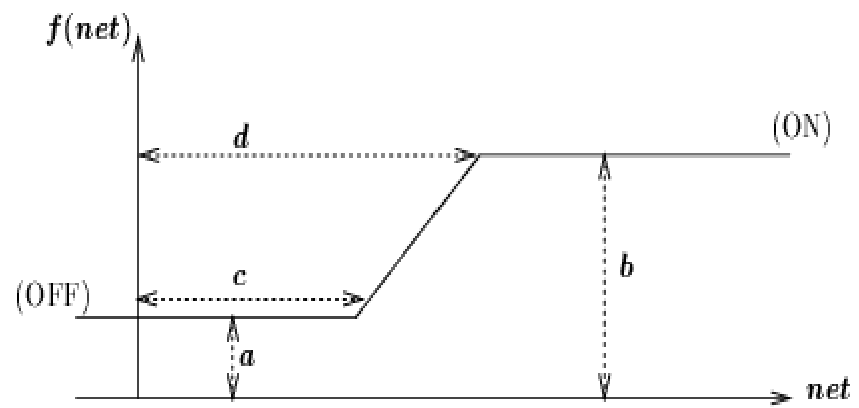
where c is called the threshold

- Ramp function

$$f(\text{net}) = \begin{cases} a & \text{if net} \le c \\ b & \text{if net} \ge d \\ a + \frac{(\text{net}-c)(b-a)}{(d-c)} & \text{otherwise} \end{cases}$$



Step function



Ramp function

Node Function

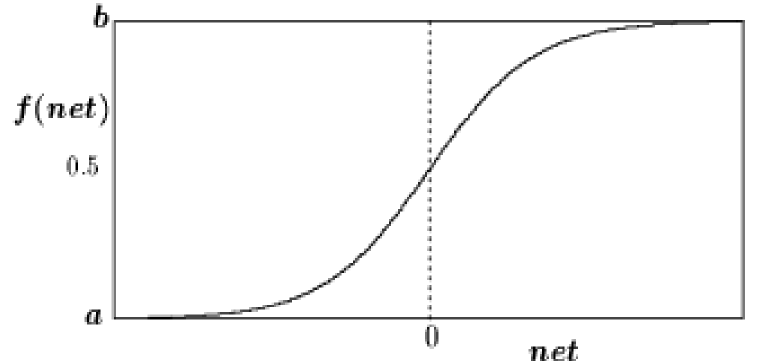
- **Sigmoid function**
  - S-shaped
  - Continuous and everywhere differentiable
  - Rotationally symmetric about some point (*net = c*)
  - Asymptotically approach saturation points

$$\lim_{\text{net}\to-\infty} f(\text{net}) = a \quad \lim_{\text{net}\to\infty} f(\text{net}) = b$$

  - Examples:

$$f(\text{net}) = z + \frac{1}{1+\exp(-x\cdot\text{net}+y)}$$

$$f(\text{net}) = \tanh(x\cdot\text{net}-y)+z,$$



Sigmoid function

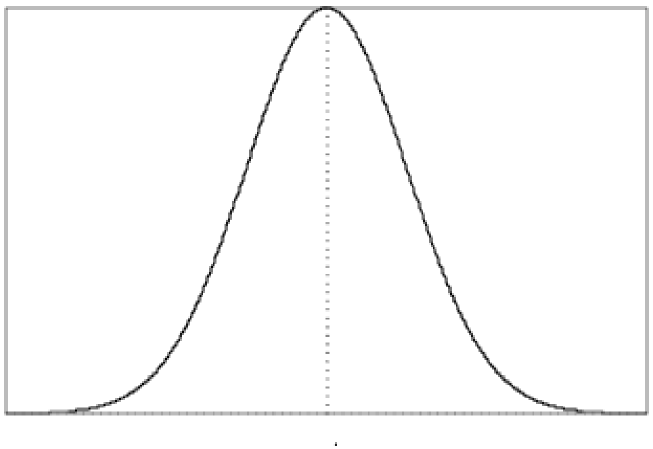
When $y = 0$ and $z = 0$:
$a = 0$, $b = 1$, $c = 0$.
When $y = 0$ and $z = -0.5$
$a = -0.5$, $b = 0.5$, $c = 0$.

Larger x gives steeper curve

Node Function

- **Gaussian function**
  - Bell-shaped (radial basis)
  - Continuous
  - $f(net)$ asymptotically approaches 0 (or some constant) when $|net|$ is large
  - Single maximum (when $net = \mu$)
  - Example:

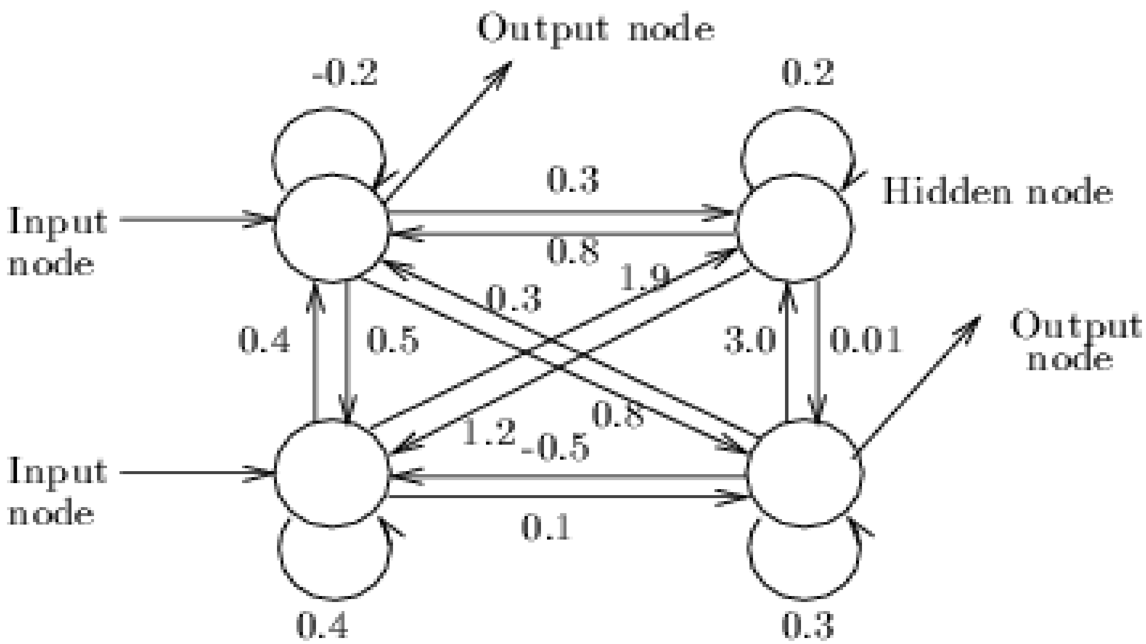
$$f(\mathrm{net}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{\mathrm{net}-\mu}{\sigma}\right)^2\right]$$



Gaussian function

Network Architecture

- **(Asymmetric) Fully Connected Networks**
  – Every node is connected to every other node
  – Connection may be excitatory (positive), inhibitory (negative), or irrelevant (≈ 0).
  – Most general
  – Symmetric fully connected nets: weights are symmetric ($w_{ij} = w_{ji}$)



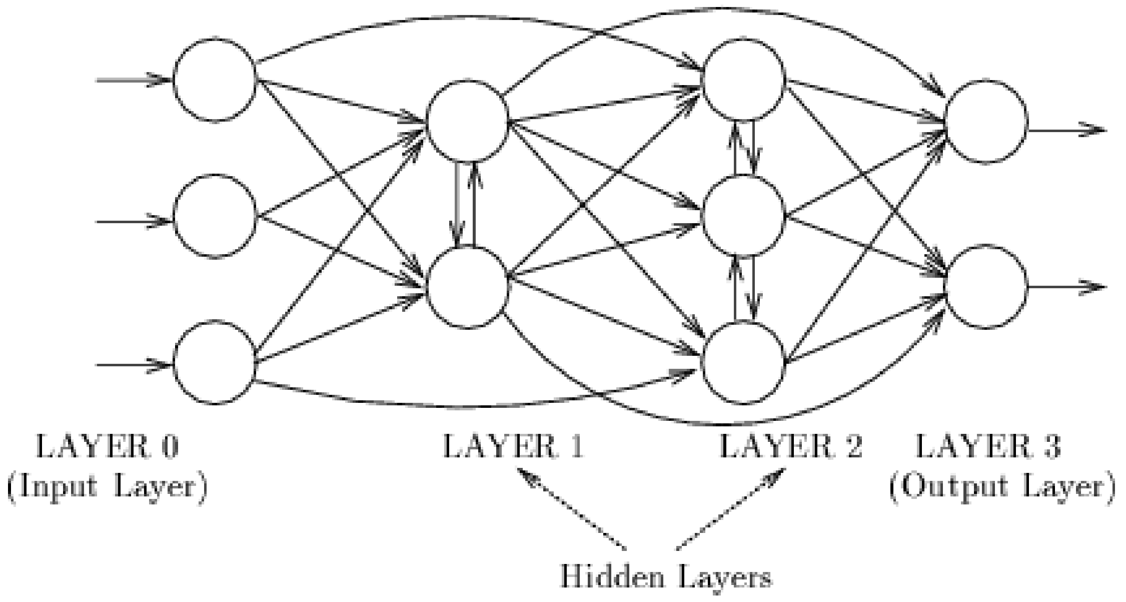
Input nodes: receive input from the environment

Output nodes: send signals to the environment

Hidden nodes: no direct interaction to the environment

Network Architecture

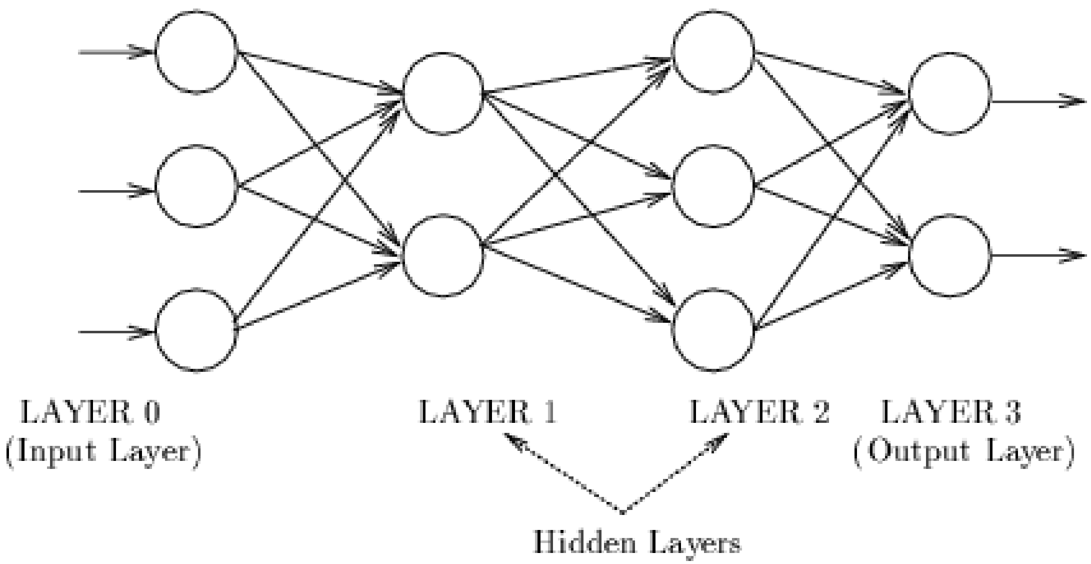
- **Layered Networks**
  - Nodes are partitioned into subsets, called layers.
  - No connections that lead from nodes in layer j to those in layer k if $j > k$.



- Inputs from the environment are applied to nodes in layer 0 (**input layer**).
- Nodes in input layer are place holders with no computation occurring (i.e., their node functions are identity function)

Network Architecture

- **Feedforward Networks**
  - A connection is allowed from a node in layer *i* only to nodes in layer *i* + 1.
  - Most widely used architecture.



Conceptually, nodes at higher levels successively abstract features from preceding layers

Network Architecture

- **Acyclic Networks**
  - Connections do not form directed cycles.
  - Multi-layered feedforward nets are acyclic
- **Recurrent Networks**
  - Nets with directed cycles.
  - Much harder to analyze than acyclic nets.
- **Modular nets**
  - Consists of several modules, each of which is itself a neural net for a particular sub-problem
  - Sparse connections between modules