

Introduction To Neural Networks



Development of Neural Networks date back to the early 1940s. It experienced an upsurge in popularity in the late 1980s. This was a result of the discovery of new techniques and developments and general advances in computer hardware technology.

Some NNs are models of biological neural networks and some are not, but historically, much of the inspiration for the field of NNs came from the desire to produce artificial systems capable of sophisticated, perhaps *intelligent*, computations similar to those that the human brain routinely performs, and thereby possibly to enhance our understanding of the human brain.

Most NNs have some sort of *training* rule. In other words, NNs *learn* from examples (as children learn to recognize dogs from examples of dogs) and exhibit some capability for *generalization* beyond the training data.

Neural computing must not be considered as a competitor to conventional computing. Rather, it should be seen as complementary as the most successful neural solutions have been those which operate in conjunction with existing, traditional techniques.



Neural Network Techniques

Computers have to be explicitly programmed

- Analyze the problem to be solved.

- Write the code in a programming language.

Neural networks learn from examples

- No requirement of an explicit description of the problem.

- No need for a programmer.

- The neural computer adapts itself during a training period, based on examples of similar problems even without a desired solution to each problem. After sufficient training the neural computer is able to relate the problem data to the solutions, inputs to outputs, and it is then able to offer a viable solution to a brand new problem.

- Able to generalize or to handle incomplete data.

NNs vs Computers

Digital Computers

Deductive Reasoning. We apply known rules to input data to produce output.

Computation is centralized, synchronous, and serial.

Memory is packetted, literally stored, and location addressable.

Not fault tolerant. One transistor goes and it no longer works.

Exact.

Static connectivity.

Applicable if well defined rules with precise input data.

Neural Networks

Inductive Reasoning. Given input and output data (training examples), we construct the rules.

Computation is collective, asynchronous, and parallel.

Memory is distributed, internalized, short term and content addressable.

Fault tolerant, redundancy, and sharing of responsibilities.

Inexact.

Dynamic connectivity.

Applicable if rules are unknown or complicated, or if data are noisy or partial.



Applications off NNs

classification

in marketing: consumer spending pattern classification

In defence: radar and sonar image classification

In agriculture & fishing: fruit and catch grading

In medicine: ultrasound and electrocardiogram image classification, EEGs, medical diagnosis

recognition and identification

In general computing and telecommunications: speech, vision and handwriting recognition

In finance: signature verification and bank note verification

assessment

In engineering: product inspection monitoring and control

In defence: target tracking

In security: motion detection, surveillance image analysis and fingerprint matching

forecasting and prediction

In finance: foreign exchange rate and stock market forecasting

In agriculture: crop yield forecasting

In marketing: sales forecasting

In meteorology: weather prediction

What can you do with an NN and what not?



In principle, NNs can compute any computable function, i.e., they can do everything a normal digital computer can do. Almost any mapping between vector spaces can be approximated to arbitrary precision by feedforward NNs

In practice, NNs are especially useful for **classification** and **function approximation** problems usually when rules such as those that might be used in an expert system cannot easily be applied.

NNs are, at least today, difficult to apply successfully to problems that concern manipulation of symbols and memory. And there are no methods for training NNs that can magically create information that is not contained in the training data.



Who is concerned with NNs?

Computer scientists want to find out about the properties of non-symbolic information processing with neural nets and about learning systems in general.

Statisticians use neural nets as flexible, nonlinear regression and classification models.

Engineers of many kinds exploit the capabilities of neural networks in many areas, such as signal processing and automatic control.

Cognitive scientists view neural networks as a possible apparatus to describe models of thinking and consciousness (High-level brain function).

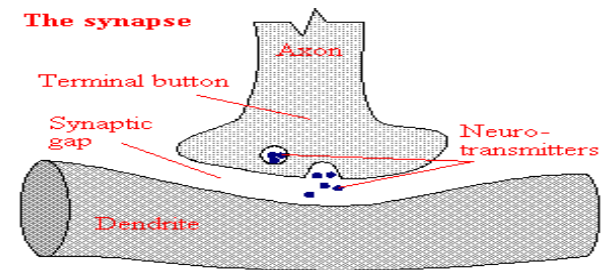
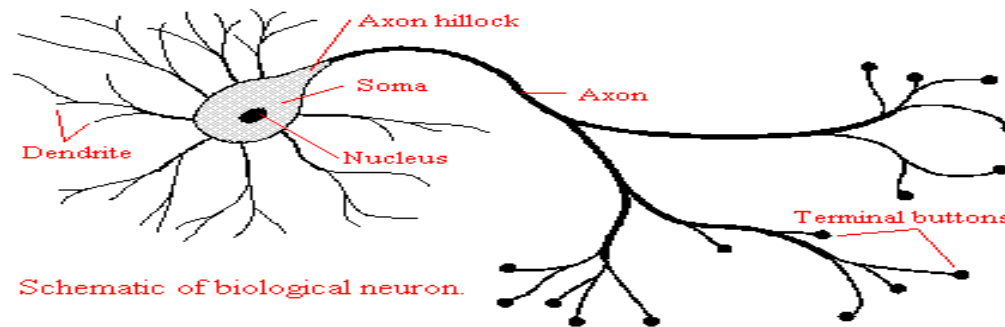
Neuro-physiologists use neural networks to describe and explore medium-level brain function (e.g. memory, sensory system, motorics).

Physicists use neural networks to model phenomena in statistical mechanics and for a lot of other tasks.

Biologists use Neural Networks to interpret nucleotide sequences.

Philosophers and some other people may also be interested in Neural Networks for various reasons

The Biological Neuron



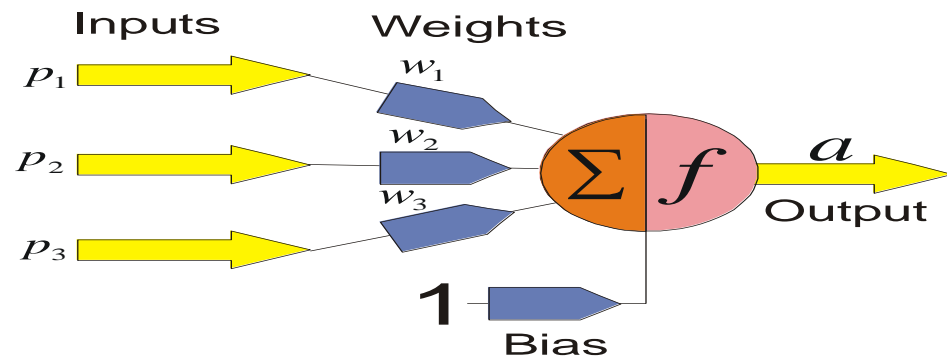
The brain is a collection of about 10 billion interconnected neurons. Each neuron is a cell that uses biochemical reactions to receive, process and transmit information.

Each terminal button is connected to other neurons across a small gap called a synapse.

A neuron's dendritic tree is connected to a thousand neighbouring neurons. When one of those neurons fire, a positive or negative charge is received by one of the dendrites. The strengths of all the received charges are added together through the processes of spatial and temporal summation.

The Key Elements of Neural Networks

Neural computing requires a number of **neurons**, to be connected together into a **neural network**. Neurons are arranged in layers.

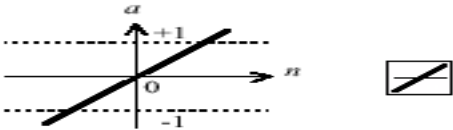
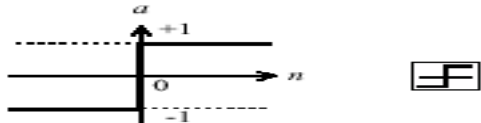
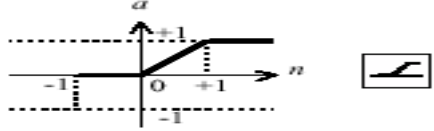
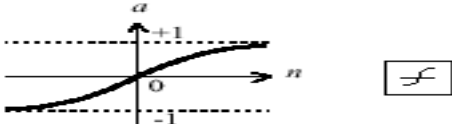
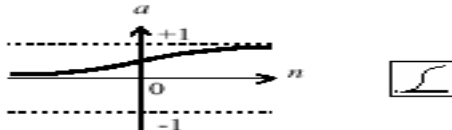
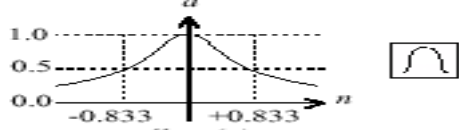


$$a = f(p_1 w_1 + p_2 w_2 + p_3 w_3 + b) = f(\sum p_i w_i + b)$$

Each neuron within the network is usually a simple processing unit which takes one or more inputs and produces an output. At each neuron, every input has an associated **weight** which modifies the strength of each input. The neuron simply adds together all the inputs and calculates an output to be passed on.

Activation functions

The activation function is generally non-linear. Linear functions are limited because the output is simply proportional to the input.

 <p>$a = \text{purelin}(n)$ Linear Transfer Function</p>	 <p>$a = \text{hardlims}(n)$ Symmetric Hard Limit Trans. Funct.</p>
 <p>$a = \text{satlin}(n)$ Satlin Transfer Function</p>	 <p>$a = \text{tansig}(n)$ Tan-Sigmoid Transfer Function</p>
 <p>$a = \text{logsig}(n)$ Log-Sigmoid Transfer Function</p>	 <p>$a = \text{radbas}(n)$ Radial Basis Function</p>

Training methods

Supervised learning

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the **training set**. During the training of a network the same set of data is processed many times as the connection weights are ever refined.

Example architectures : Multilayer perceptrons

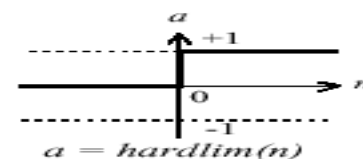
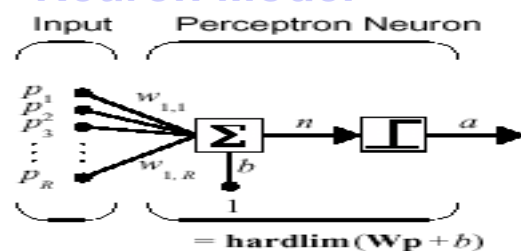
Unsupervised learning

In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaption.

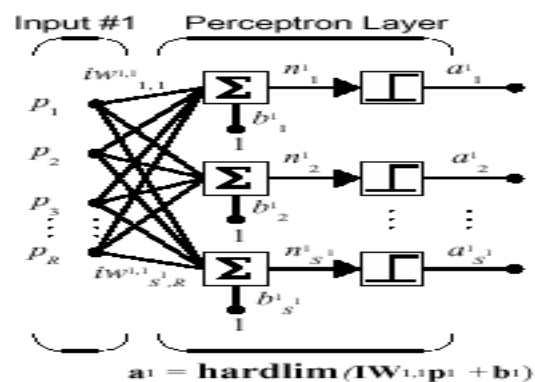
Example architectures : Kohonen, ART

Perceptrons

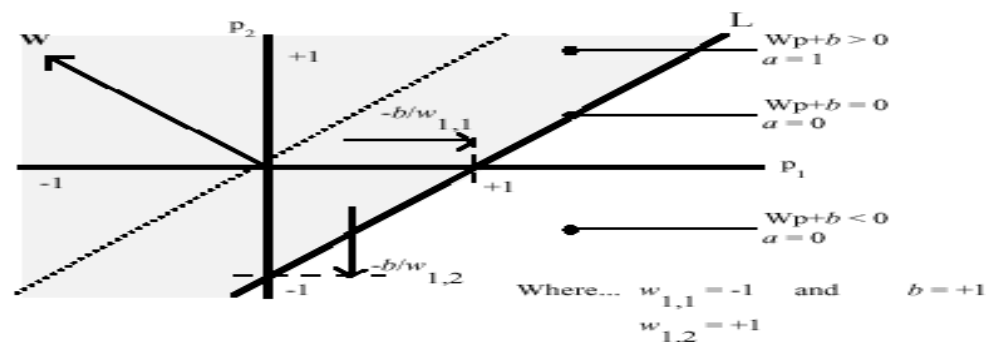
Neuron Model



Architecture

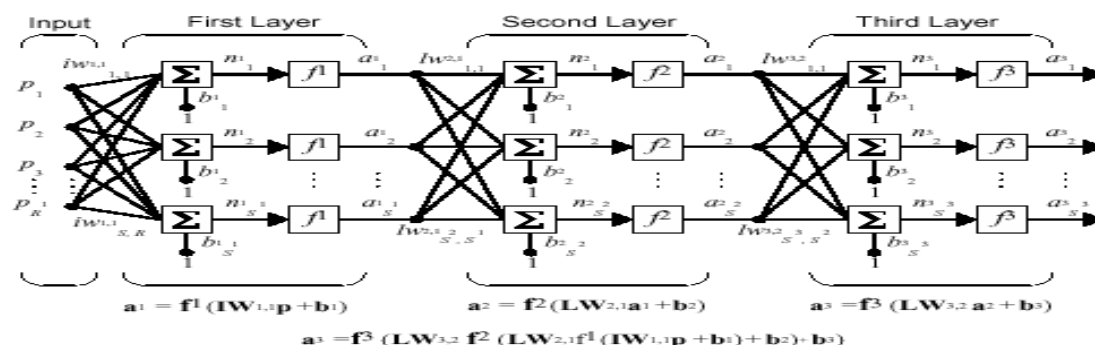
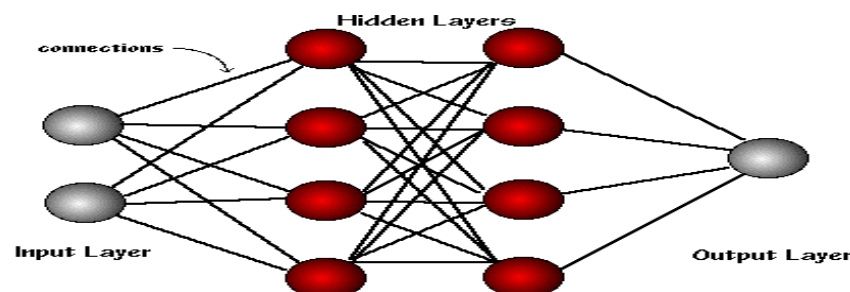


Decision boundaries



Feedforward NNs

The basic structure off a feedforward Neural Network



The learning rule modifies the weights in a sense, ANNs learn

When the desired output is achieved

with. In

Genetic Algorithm

Biological Evolution

Lamarck and others:

- Species “transmute” over time

Darwin and Wallace:

- Consistent, heritable variation among individuals in population
- Natural selection of the fittest

Mendel and genetics:

- A mechanism for inheriting traits
- genotype \rightarrow phenotype mapping

Genetic Algorithm

$GA(Fitness, Fitness_threshold, p, r, m)$

- *Initialize*: $P \leftarrow p$ random hypotheses
- *Evaluate*: for each h in P , compute $Fitness(h)$
- While $[\max_h Fitness(h)] < Fitness_threshold$

1. *Select*: Probabilistically select $(1 - r)p$ members of P to add to P_s .

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^p Fitness(h_j)}$$

2. *Crossover*: Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from P . For each pair, $\langle h_1, h_2 \rangle$, produce two offspring by applying the Crossover operator. Add all offspring to P_s .

3. *Mutate*: Invert a randomly selected bit in $m \cdot p$ random members of P_s

4. *Update*: $P \leftarrow P_s$

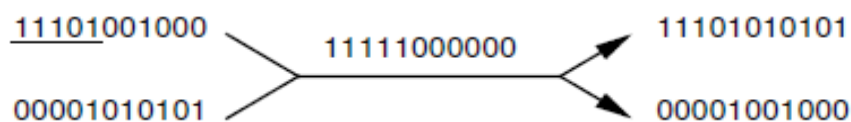
5. *Evaluate*: for each h in P , compute $Fitness(h)$

- Return the hypothesis from P that has the highest fitness.

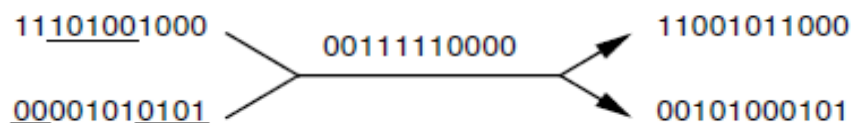
Operators for Genetic Algorithms

Initial strings Crossover Mask Offspring

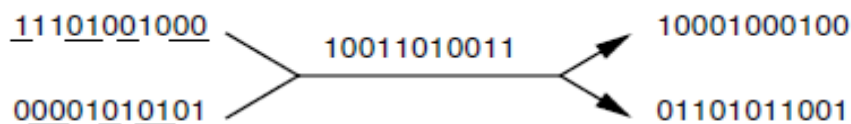
Single-point crossover:



Two-point crossover:



Uniform crossover:



Point mutation:

