Technological Educational Institute Of Crete

Department Of Applied Informatics and Multimedia

Neural Networks Laboratory

Introduction To Neural Networks

Prof. George Papadourakis, Ph.D.

Part I

Introduction and Architectures

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.

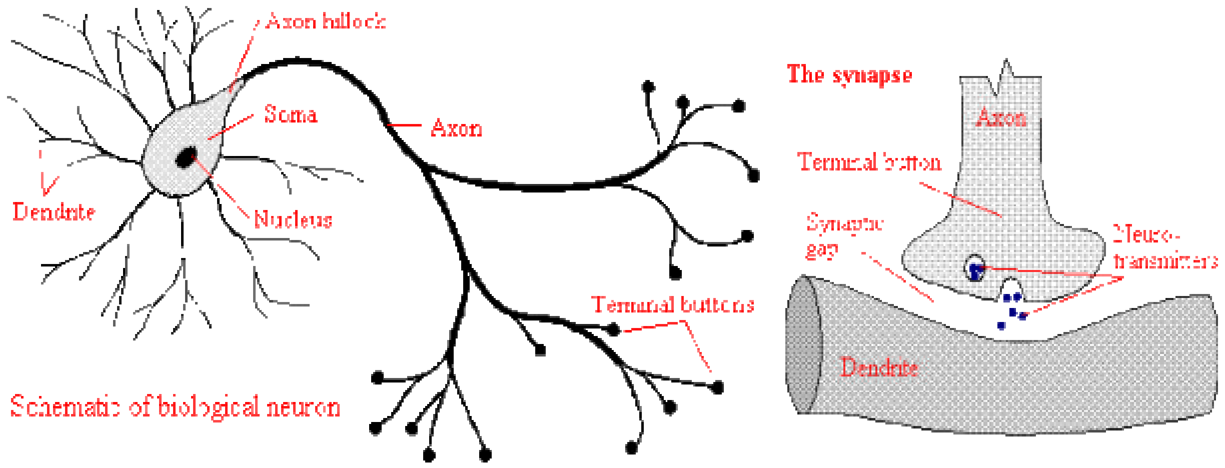
Applications off NNs

- **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.

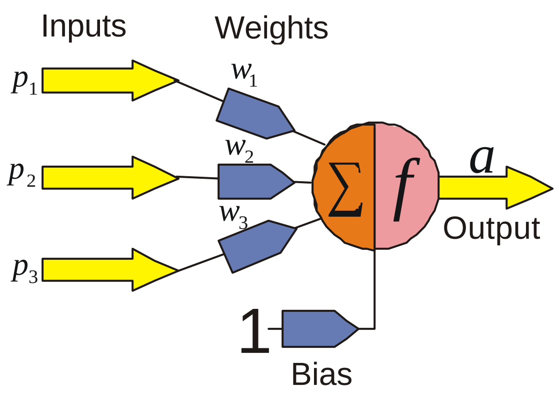
The Biological Neuron



- 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.

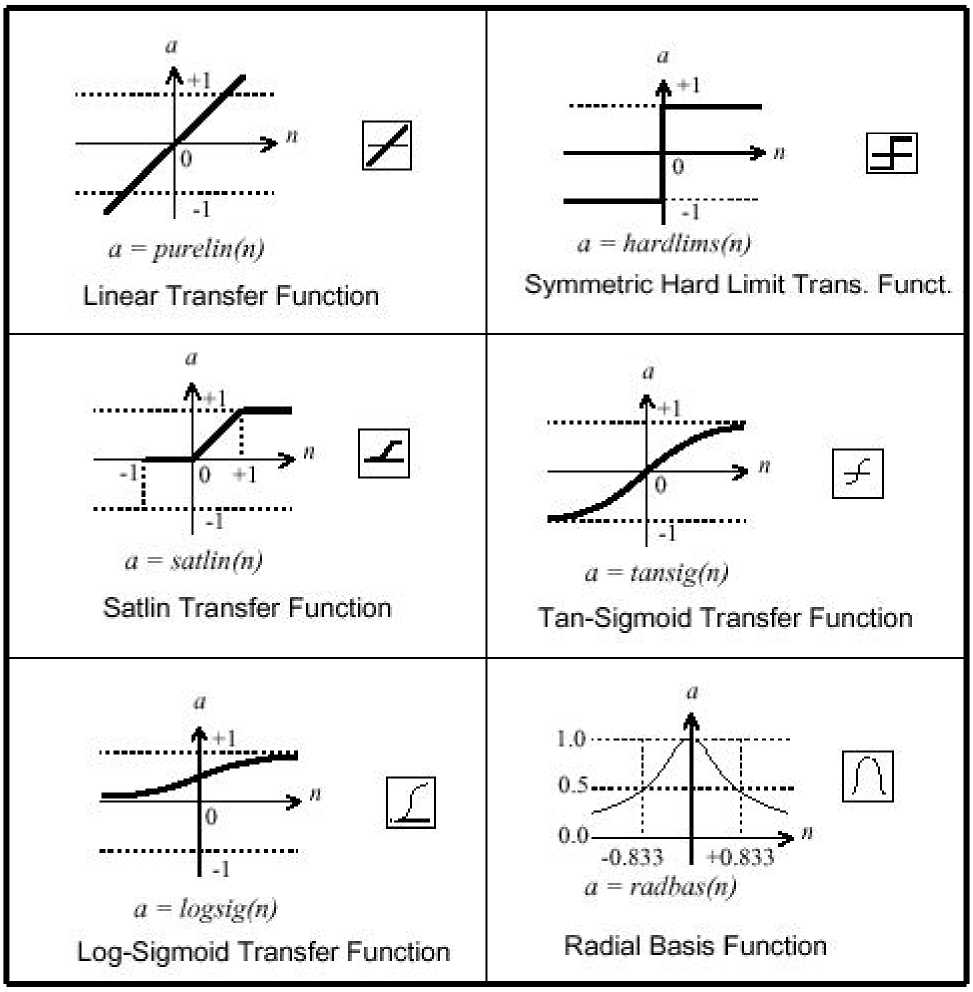


$$a = f(p_1w_1 + p_2w_2 + p_3w_3 + b) = f\left(\sum p_iw_i + b\right)$$

- 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.



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

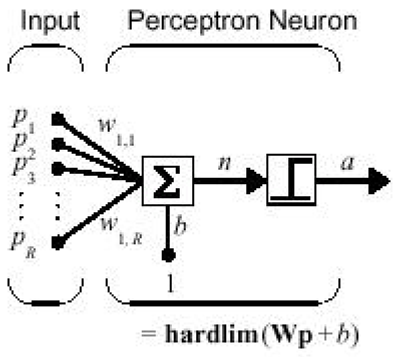
- **Unsupervised learning**

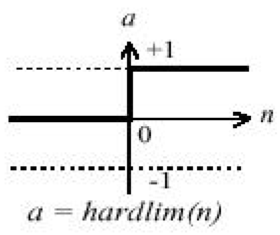
  In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaption.

  Example architectures : Kohonen, ART

Perceptrons

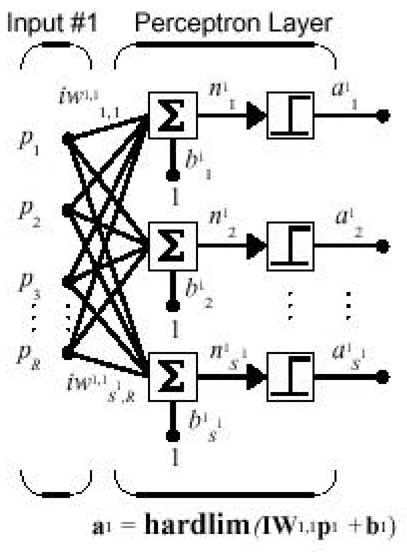
Neuron Model



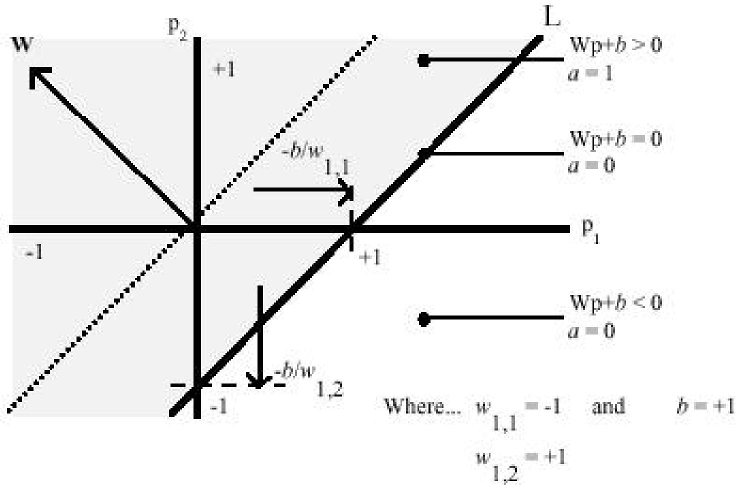


The perceptron neuron produces a 1 if the net input into the transfer function is equal to or greater than 0, otherwise it produces a 0.

Architecture

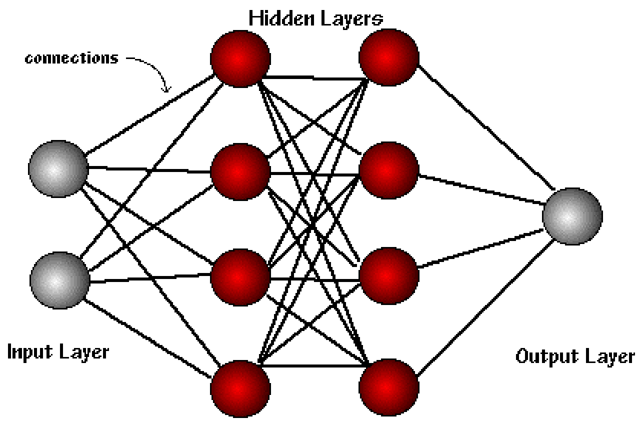


Decision boundaries

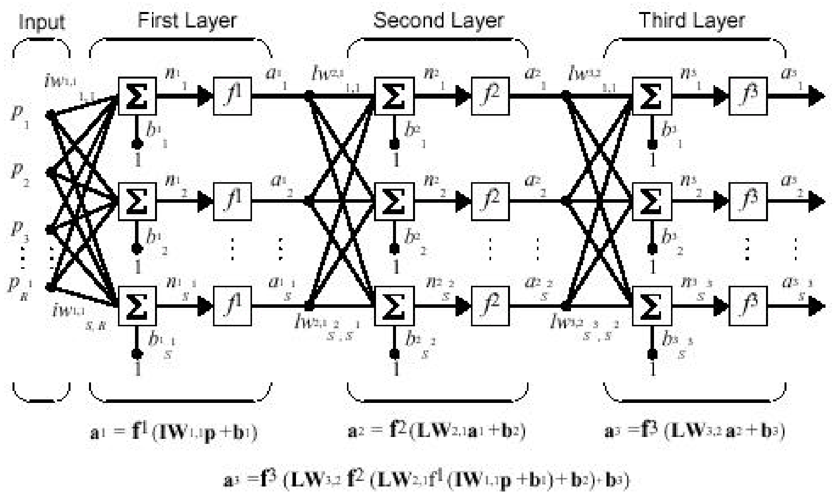


Feedforword NNs

- The basic structure off a feedforward Neural Network

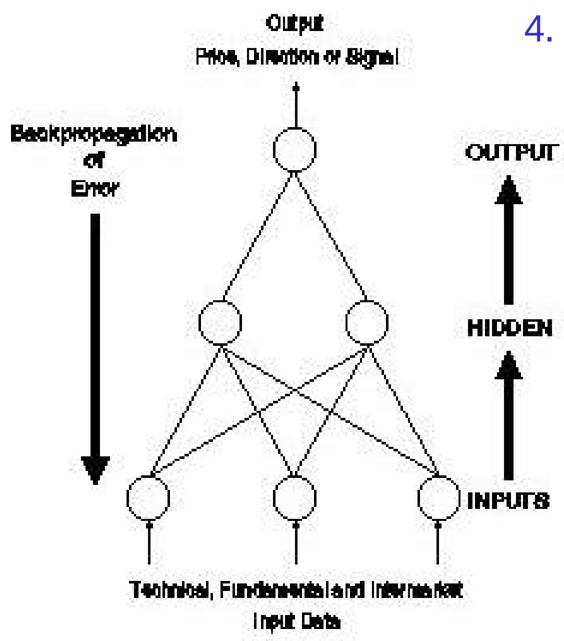


- The learning rule modifies the weights according to the input patterns that it is presented with. In a sense, ANNs learn by example as do their biological counterparts.
- When the desired output are known we have supervised learning or learning with a teacher.



An overview of the backpropagation

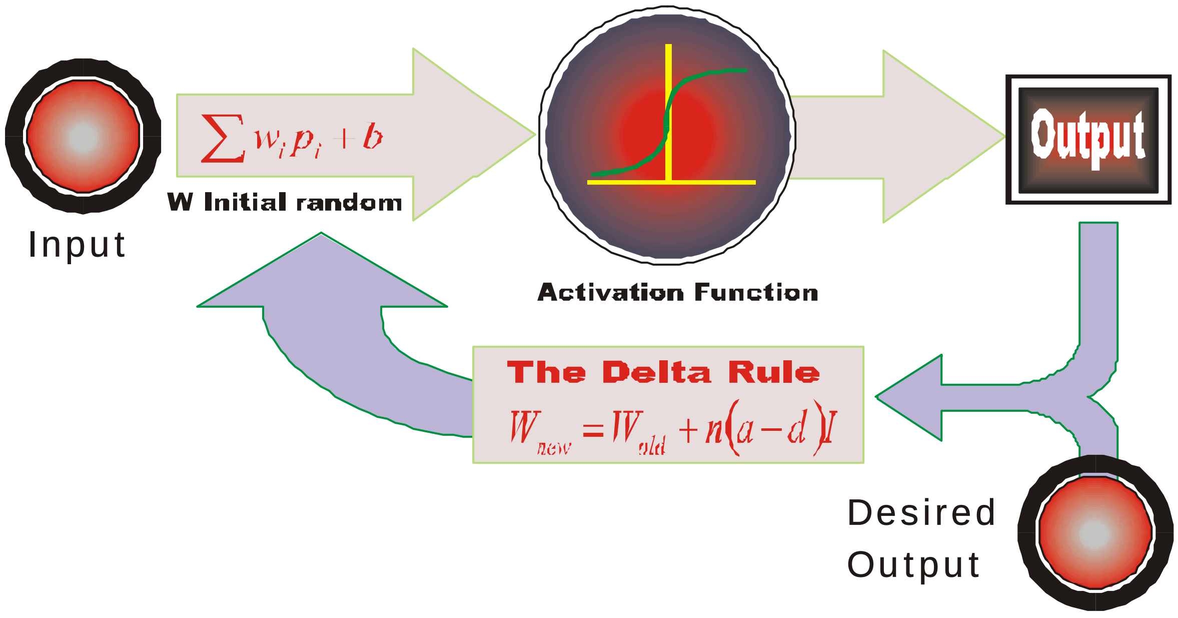
1. A set of examples for training the network is assembled. Each case consists of a problem statement (which represents the input into the network) and the corresponding solution (which represents the desired output from the network).
2. The input data is entered into the network via the input layer.
3. Each neuron in the network processes the input data with the resultant values steadily "percolating" through the network, layer by layer, until a result is generated by the output layer.


Output
Price, Direction or Signal
Backpropagation of Error
OUTPUT
HIDDEN
INPUTS
Technical, Fundamental and Intermarket Input Data

4. The actual output of the network is compared to expected output for that particular input. This results in an *error value.*. The connection weights in the network are gradually adjusted, working backwards from the output layer, through the hidden layer, and to the input layer, until the correct output is produced. Fine tuning the weights in this way has the effect of teaching the network how to produce the correct output for a particular input, i.e. the network *learns*.

The Learning Rule

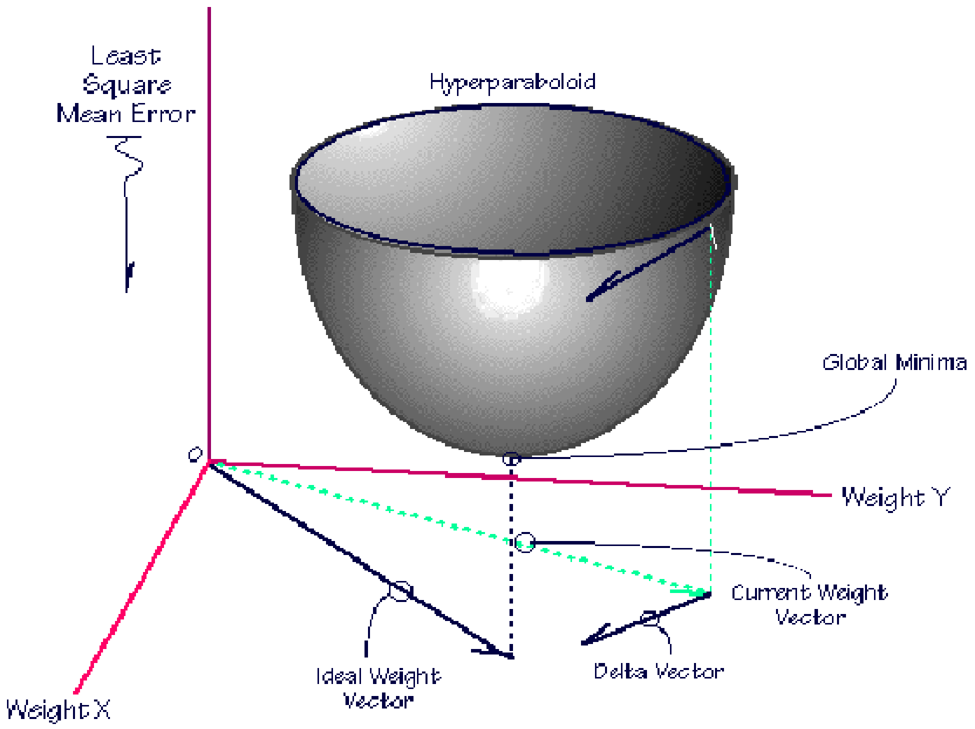
- The delta rule is often utilized by the most common class of ANNs called backpropagational neural networks.



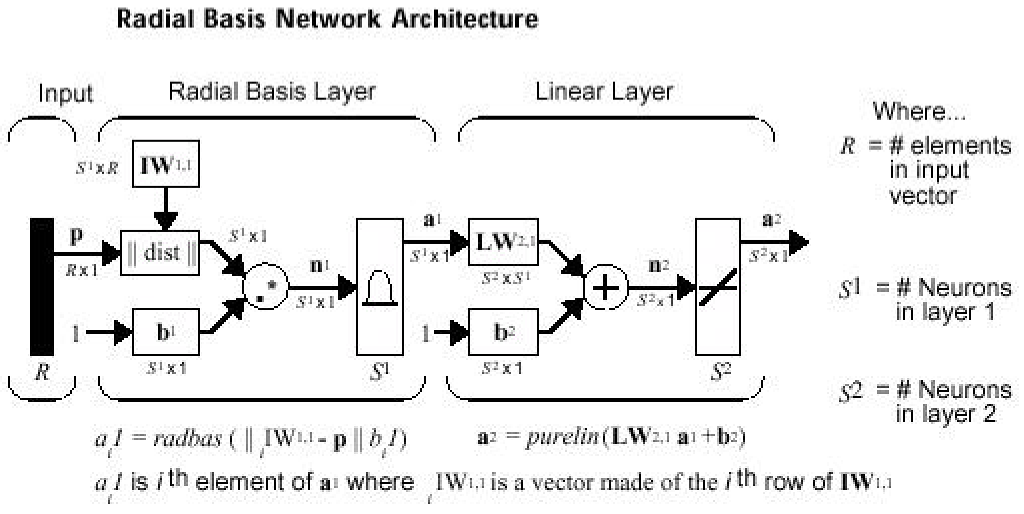
- When a neural network is initially presented with a pattern it makes a random guess as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights.

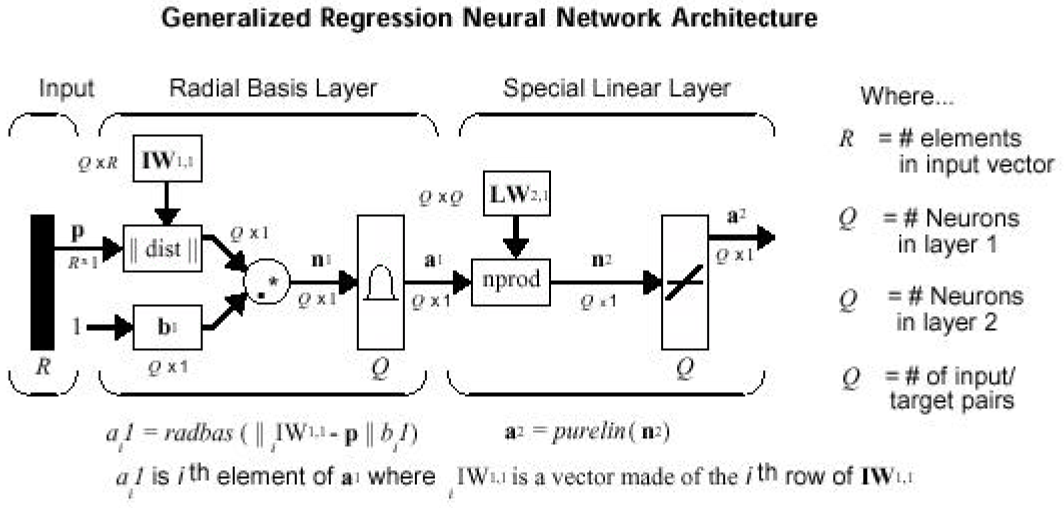
The Insides off Delta Rule

- Backpropagation performs a gradient descent within the solution's vector space towards a global minimum.
  The error surface itself is a hyperparaboloid but is seldom smooth as is depicted in the graphic below. Indeed, in most problems, the solution space is quite irregular with numerous pits and hills which may cause the network to settle down in a local minimum which is not the best overall solution.



Other architectures


Radial Basis Network Architecture
Input
Radial Basis Layer
Linear Layer
Where...
R = # elements in input vector
S1 = # Neurons in layer 1
S2 = # Neurons in layer 2
IW1,1
|| dist ||
b1
LW2,1
b2
a1 = radbas (|| IW1,1 - p || b1)
a2 = purelin(LW2,1 a1 +b2)
a1 is i th element of a1 where IW1,1 is a vector made of the i th row of IW1,1


Generalized Regression Neural Network Architecture
Input
Radial Basis Layer
Special Linear Layer
Where...
R = # elements in input vector
Q = # Neurons in layer 1
Q = # Neurons in layer 2
Q = # of input/ target pairs
IW1,1
|| dist ||
b1
LW2,1
nprod
a1 = radbas (|| IW1,1 - p || b1)
a2 = purelin(n2)
a1 is i th element of a1 where IW1,1 is a vector made of the i th row of IW1,1

Introduction To Neural Networks

Prof. George Papadourakis, Ph.D.

Part II
Application Development

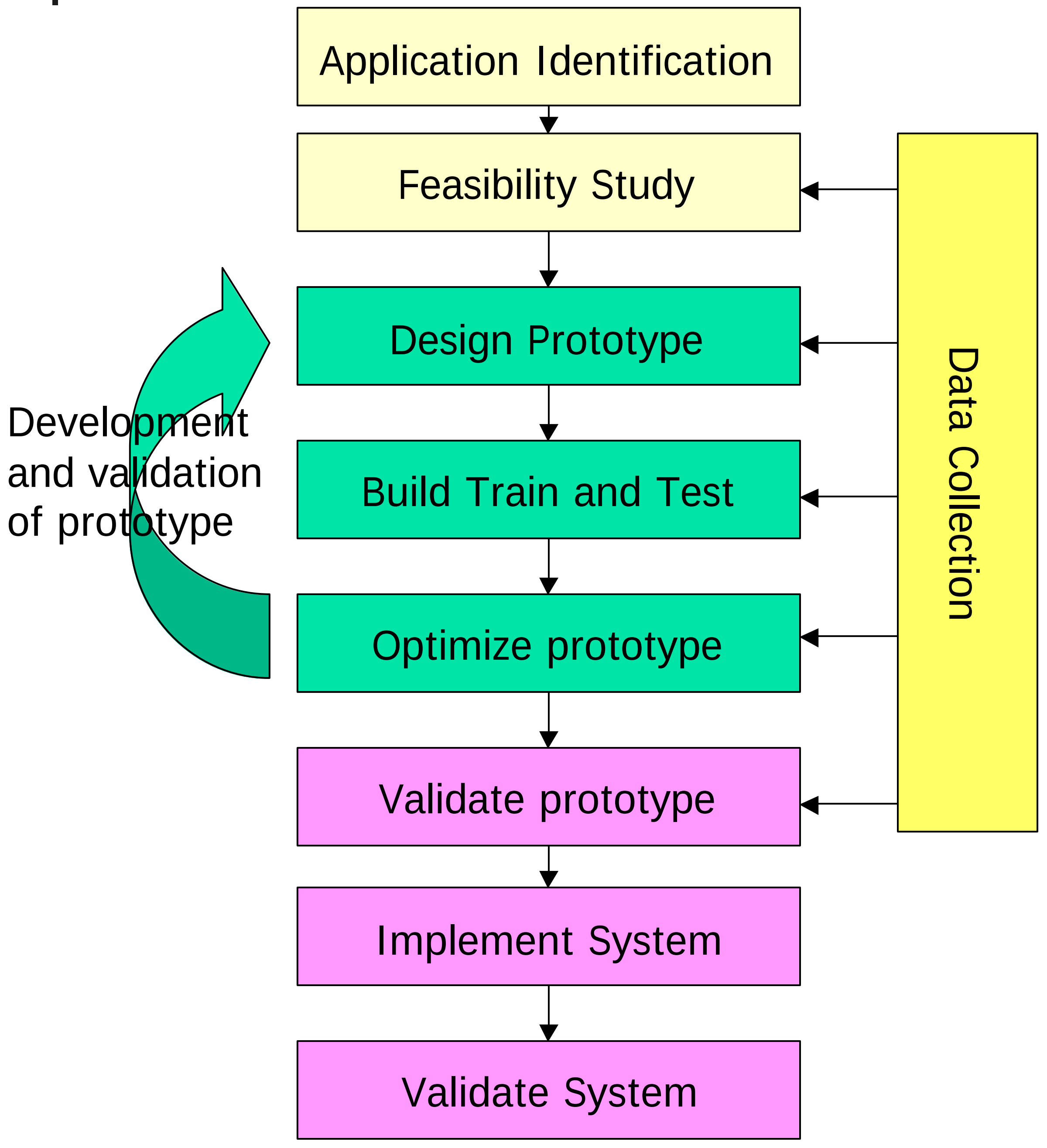
Characteristics of NNs

- Learning from experience: Complex difficult to solve problems, but with plenty of data that describe the problem
- Generalizing from examples: Can interpolate from previous learning and give the correct response to unseen data
- Rapid applications development: NNs are generic machines and quite independent from domain knowledge
- Adaptability: Adapts to a changing environment, if is properly designed
- Computational efficiency: Although the training off a neural network demands a lot of computer power, a trained network demands almost nothing in recall mode
- Non-linearity: Not based on linear assumptions about the real word

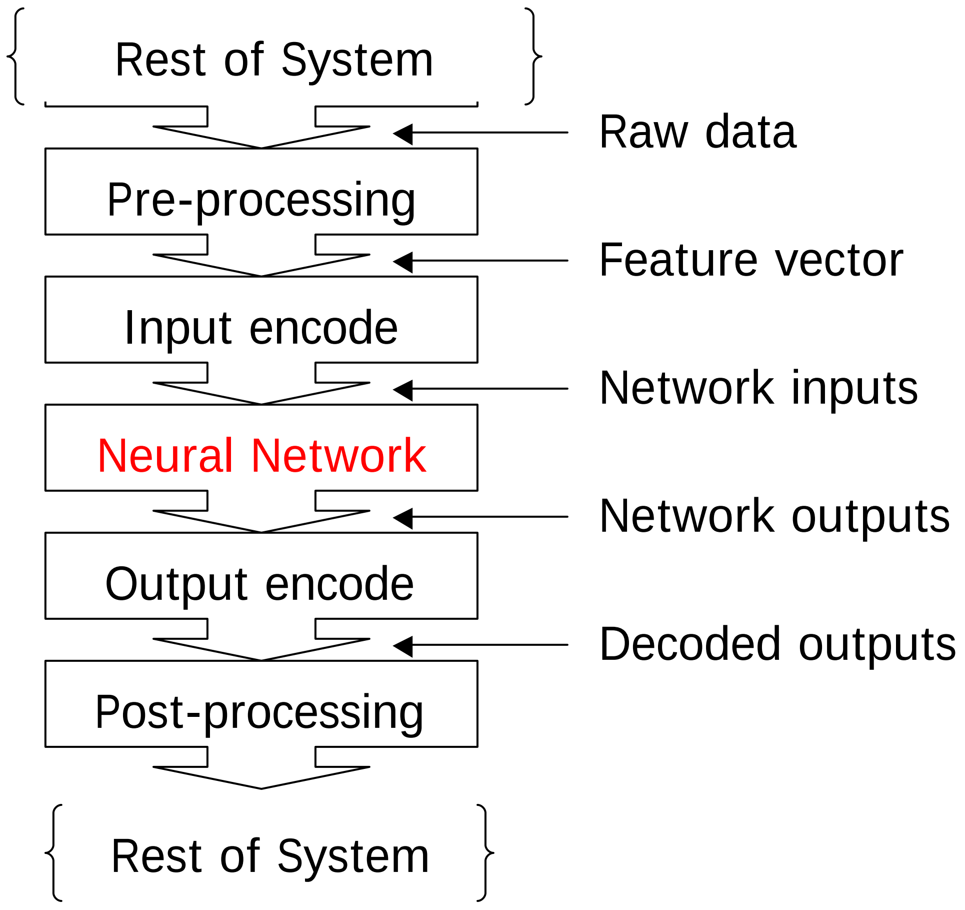
Neural Networks Projects Are Different

- Projects are data driven: Therefore, there is a need to collect and analyse data as part of the design process and to train the neural network. This task is often time-consuming and the effort, resources and time required are frequently underestimated
- It is not usually possible to specify fully the solution at the design stage: Therefore, it is necessary to build prototypes and experiment with them in order to resolve design issues. This iterative development process can be difficult to control
- Performance, rather than speed of processing, is the key issue: More attention must be paid to performance issues during the requirements analysis, design and test phases. Furthermore, demonstrating that the performance meets the requirements can be particularly difficult.
- These issues affect the following areas :
  - Project planning
  - Project management
  - Project documentation

Project life cycle



NNs in real problems



Pre-processing

- Transform data to NN inputs
  - Applying a mathematical or statistical function
  - Encoding textual data from a database
- Selection of the most relevant data and outlier removal
- Minimizing network inputs
  - Feature extraction
  - Principal components analysis
  - Waveform / Image analysis
- Coding pre-processing data to network inputs