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

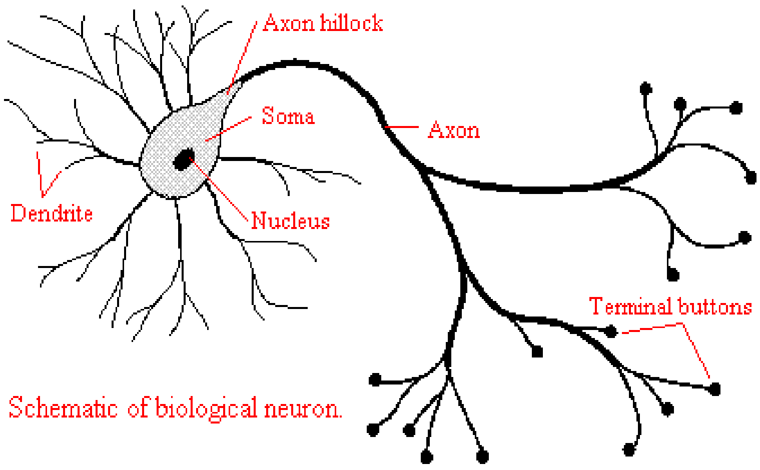
What can you do with an NN and what not?

- 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.

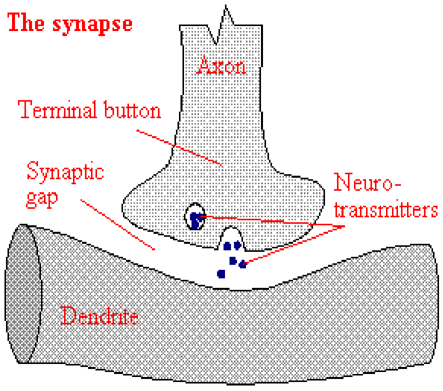
Who is concerned with NNs?

- 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



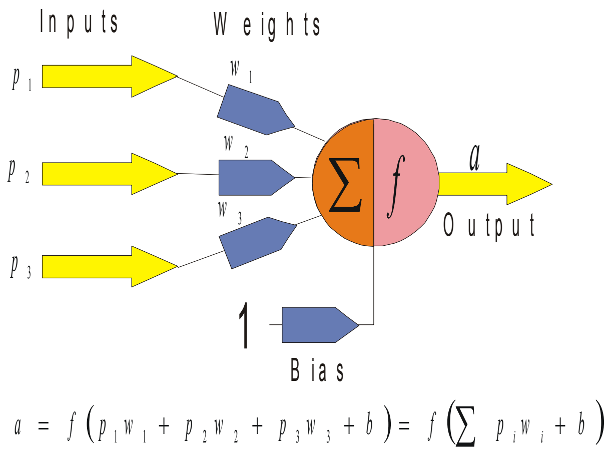
Schematic of biological neuron.



- 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.

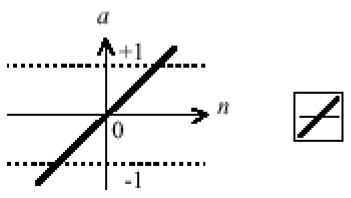
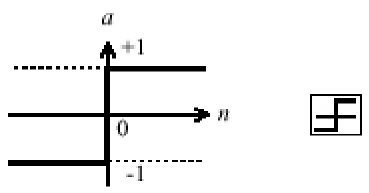
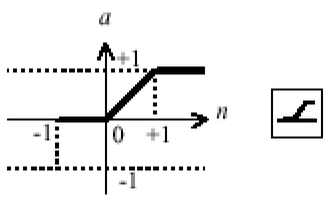
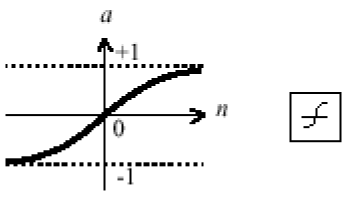
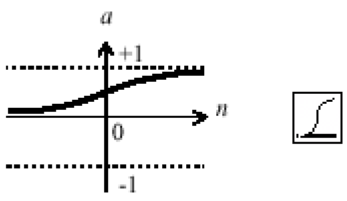
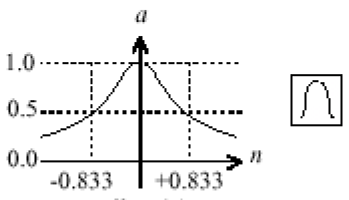


$$a = f(p_1 w_1 + p_2 w_2 + p_3 w_3 + b) = f\left(\sum p_i w_i + b\right)$$

- 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.

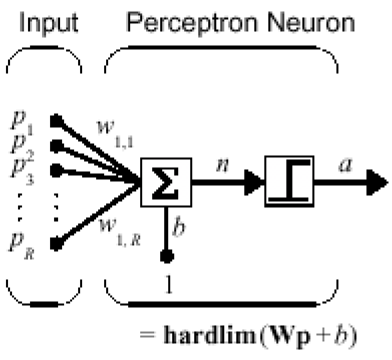
 $a = purelin(n)$ Linear Transfer Function	 $a = hardlims(n)$ Symmetric Hard Limit Trans. Funct.
 $a = satlin(n)$ Satlin Transfer Function	 $a = tansig(n)$ Tan-Sigmoid Transfer Function
 $a = logsig(n)$ Log-Sigmoid Transfer Function	 $a = radbas(n)$ Radial Basis Function

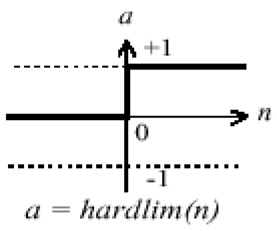
Training methods

- **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

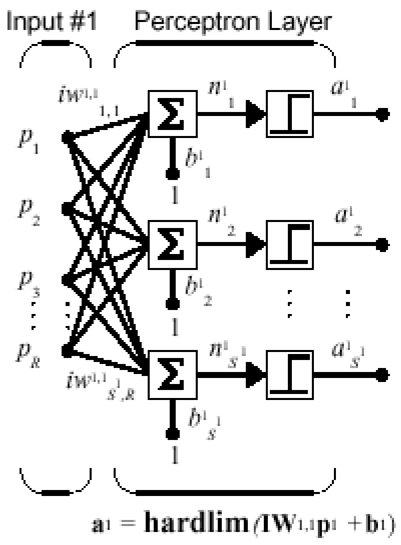
Neuron Model



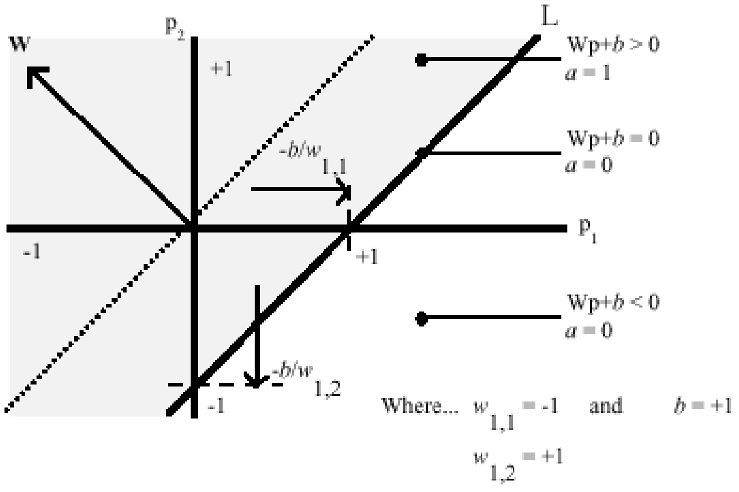


The perceptron neuron produces a 1 if the net input into the transfer function is equal to or greater than 0, otherwise it produces a 0.

Architecture



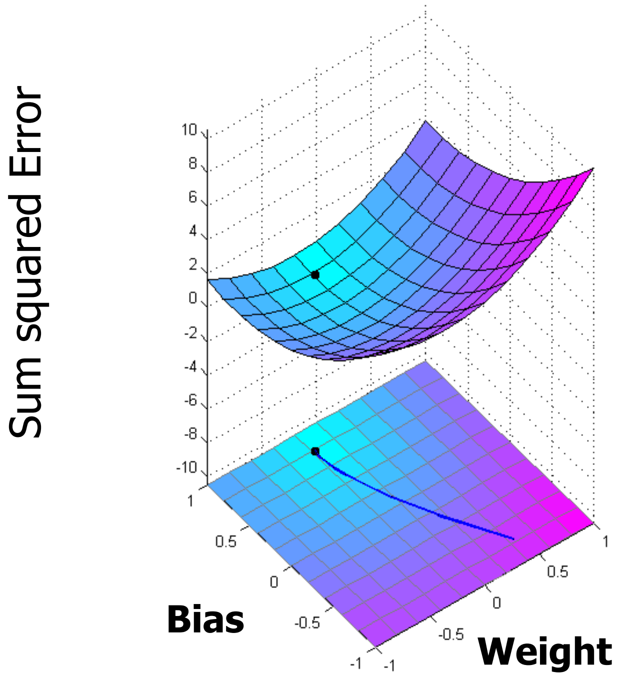
Decision boundaries

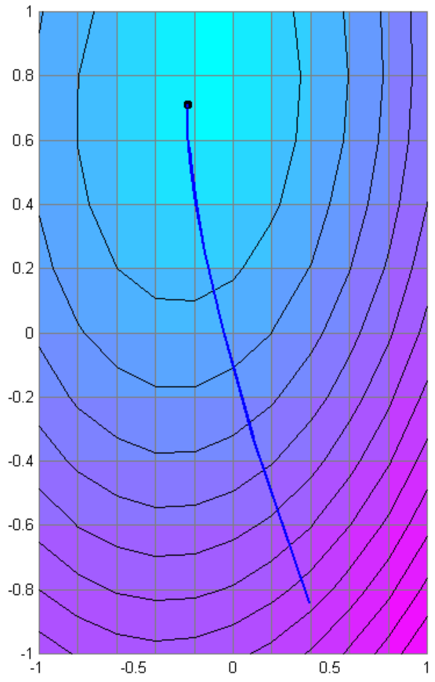


Error Surface

Error surface

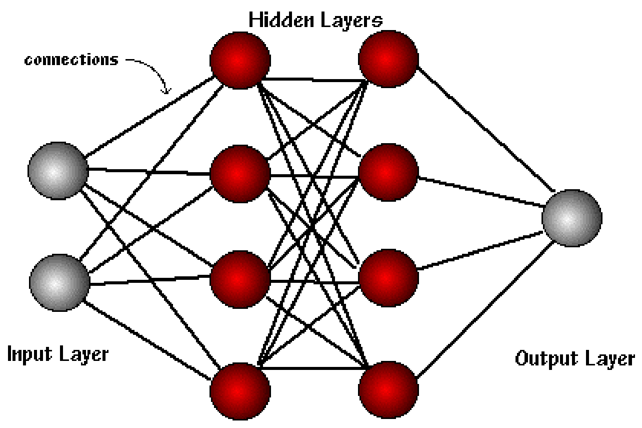
Error Contour



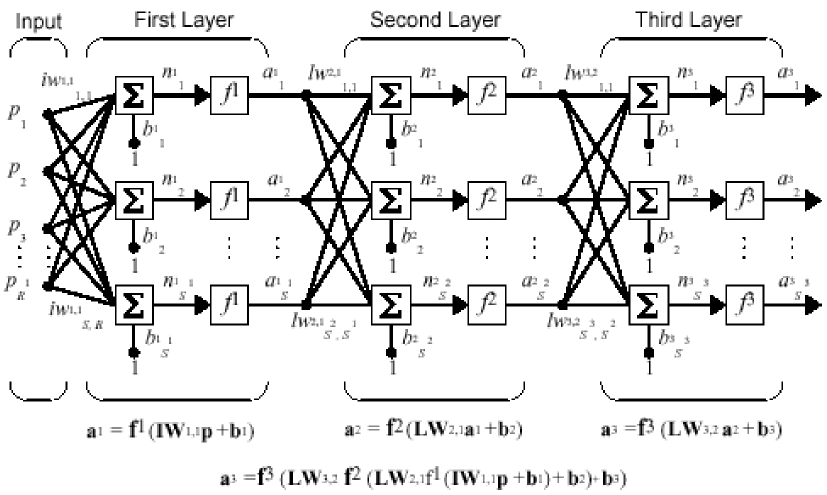


Feedforword NNs

- The basic structure off a feedforward Neural Network

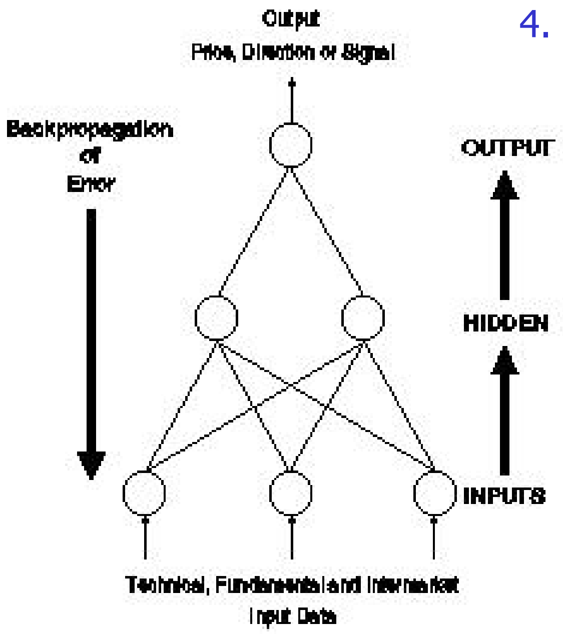


- The learning rule modifies the weights according to the input patterns that it is presented with. In a sense, ANNs learn by example as do their biological counterparts.
- When the desired output are known we have supervised learning or learning with a teacher.


Input
First Layer
Second Layer
Third Layer
$a_1 = f^1(IW_{1,1}p + b_1)$
$a_2 = f^2(LW_{2,1}a_1 + b_2)$
$a_3 = f^3(LW_{3,2}a_2 + b_3)$
$a_3 = f^3(LW_{3,2}f^2(LW_{2,1}f^1(IW_{1,1}p + b_1) + b_2) + b_3)$

An overview of the backpropagation

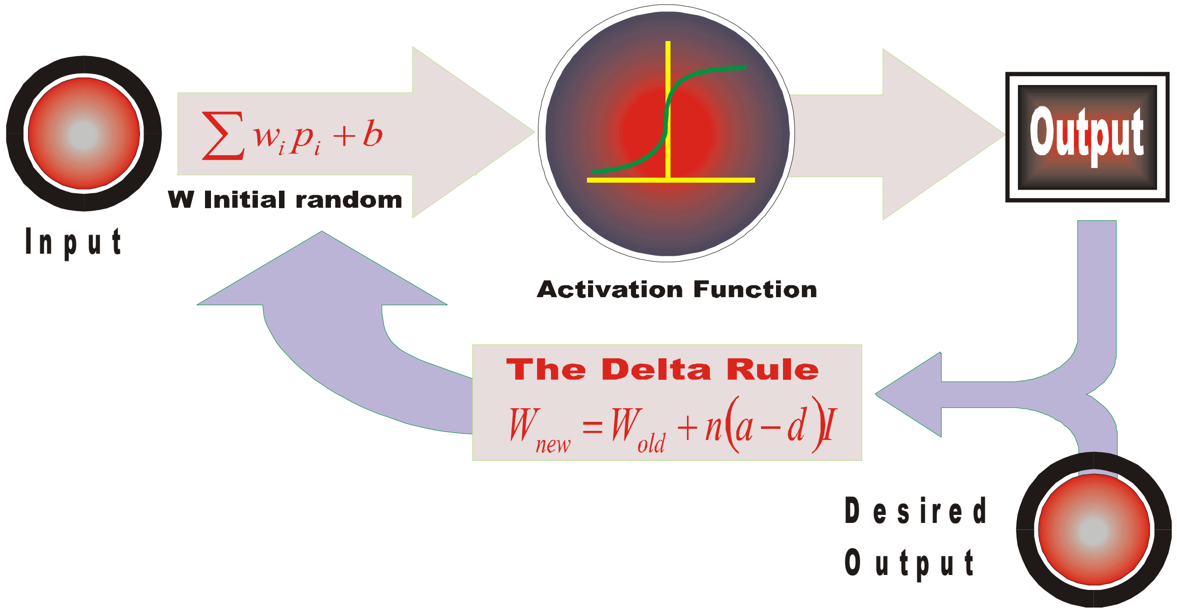
1. A set of examples for training the network is assembled. Each case consists of a problem statement (which represents the input into the network) and the corresponding solution (which represents the desired output from the network).
2. The input data is entered into the network via the input layer.
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

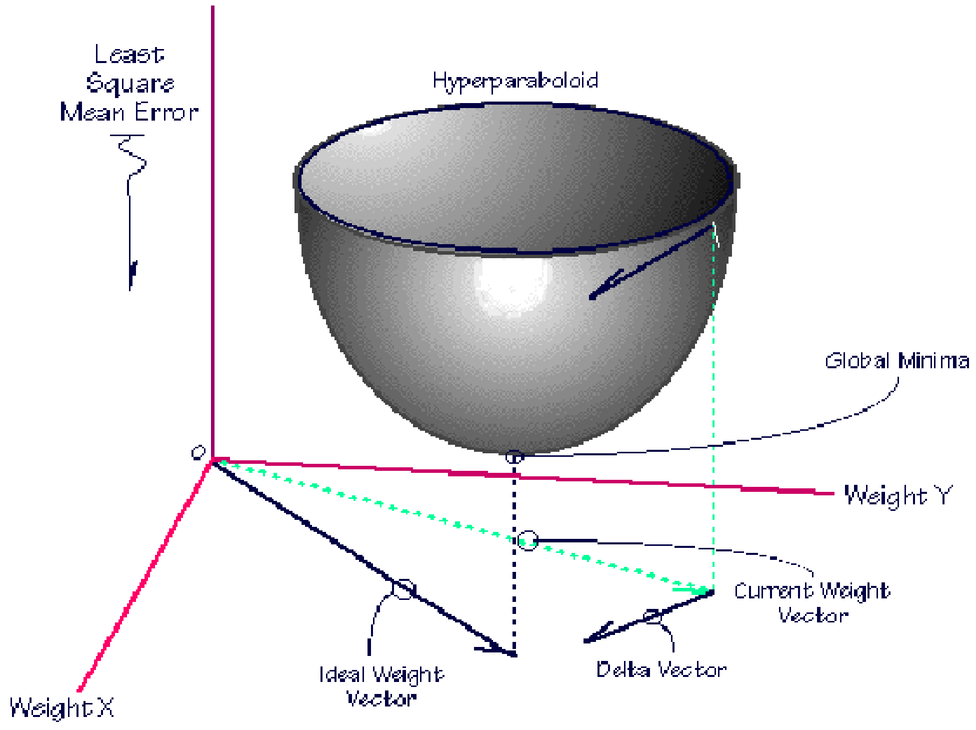
- The delta rule is often utilized by the most common class of ANNs called backpropagational neural networks.



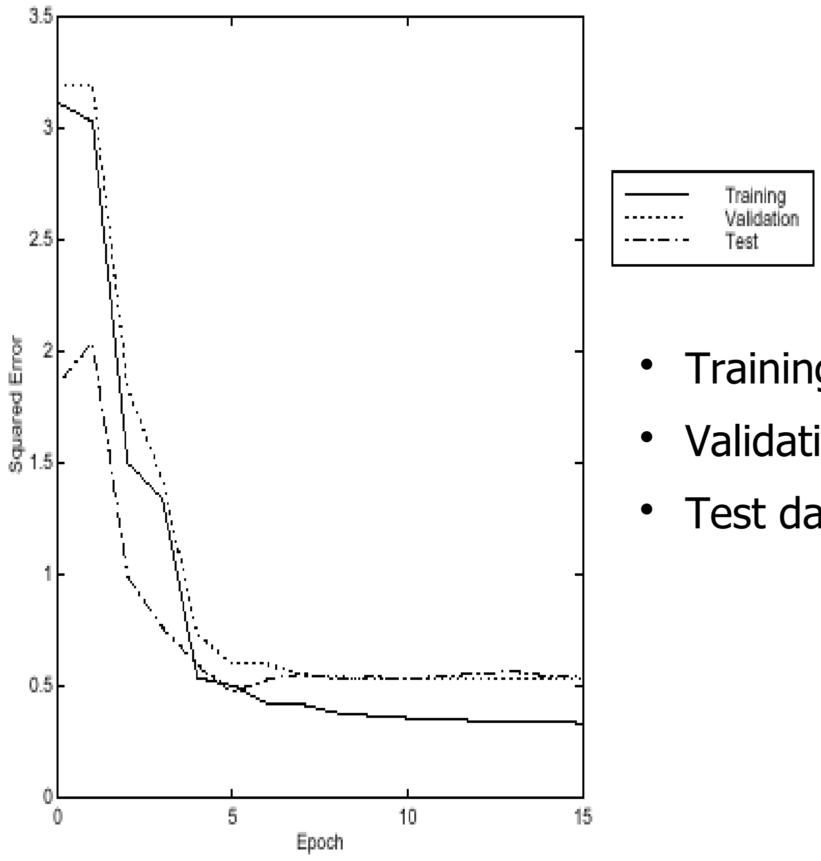
- When a neural network is initially presented with a pattern it makes a random guess as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights.

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.



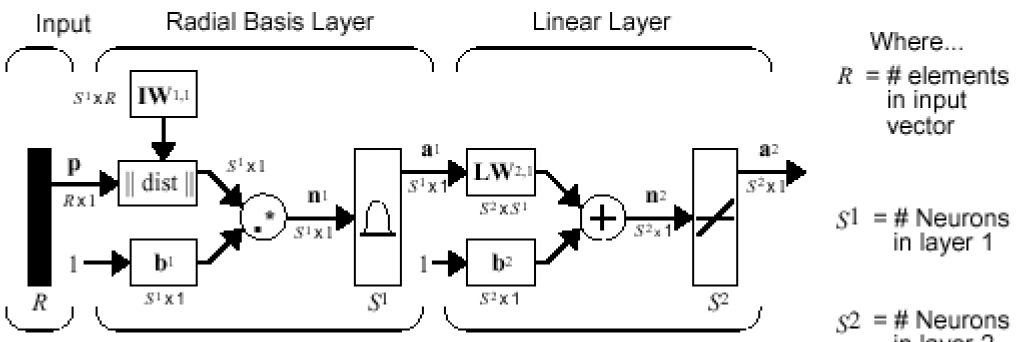
Early stopping



- Training data
- Validation data
- Test data

Other architectures

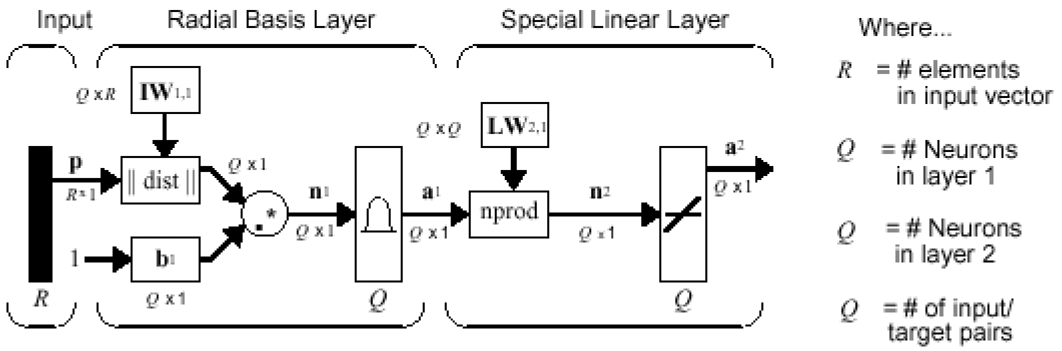
Radial Basis Network Architecture



$a_i^1 = radbas(\| {}_i\mathbf{IW}^{1,1} - \mathbf{p} \| \, b_i^1)$ $\mathbf{a}^2 = purelin(\mathbf{LW}^{2,1}\mathbf{a}^1 + \mathbf{b}^2)$

a_i^1 is ith element of $\mathbf{a}^1$ where ${}_i\mathbf{IW}^{1,1}$ is a vector made of the ith row of $\mathbf{IW}^{1,1}$

Generalized Regression Neural Network Architecture



$a_i^1 = radbas(\| {}_i\mathbf{IW}^{1,1} - \mathbf{p} \| \, b_i^1)$ $\mathbf{a}^2 = purelin(\mathbf{n}^2)$

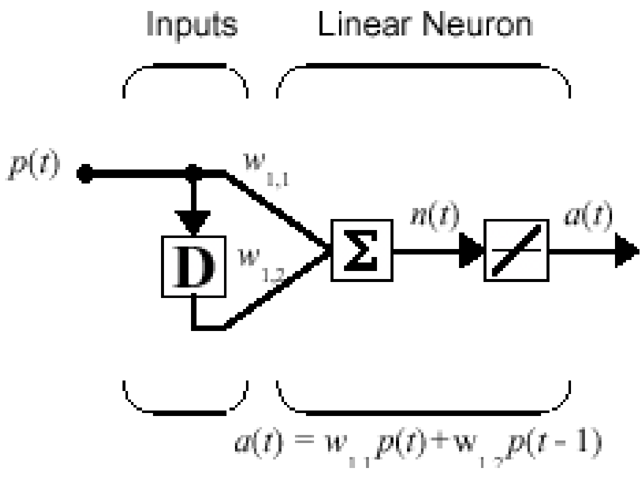
a_i^1 is ith element of $\mathbf{a}^1$ where ${}_i\mathbf{IW}^{1,1}$ is a vector made of the ith row of $\mathbf{IW}^{1,1}$

Design Conciderations

- What transfer function should be used?
- How many inputs does the network need?
- How many hidden layers does the network need?
- How many hidden neurons per hidden layer?
- How many outputs should the network have?

There is no standard methodology to determinate these values. Even there is some heuristic points, final values are determinate by a trial and error procedure.

Time Delay NNs



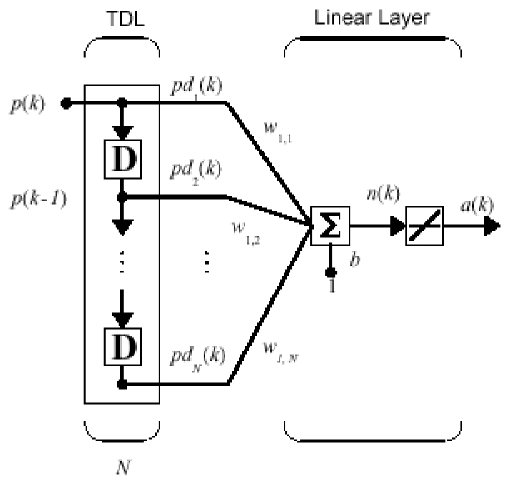
$a(t) = w_{1,1}p(t) + w_{1,2}p(t-1)$

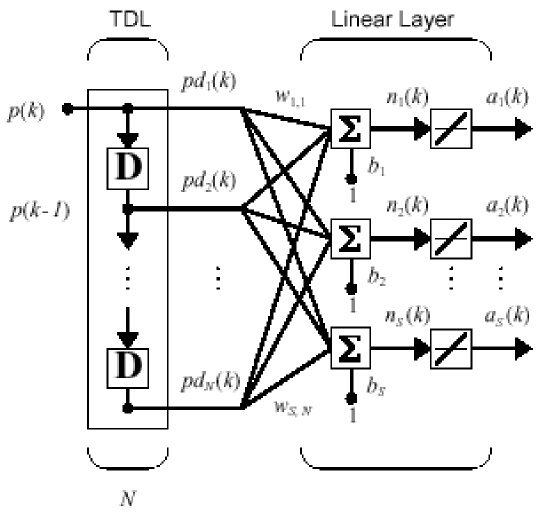
A recurrent neural network is one in which the outputs from the output layer are fed back to a set of input units. This is in contrast to feed-forward networks, where the outputs are connected only to the inputs of units in subsequent layers.

Neural networks of this kind are able to store information about time, and therefore they are particularly suitable for forecasting and control applications: they have been used with considerable success for predicting several types of time series.

TD NNs applications

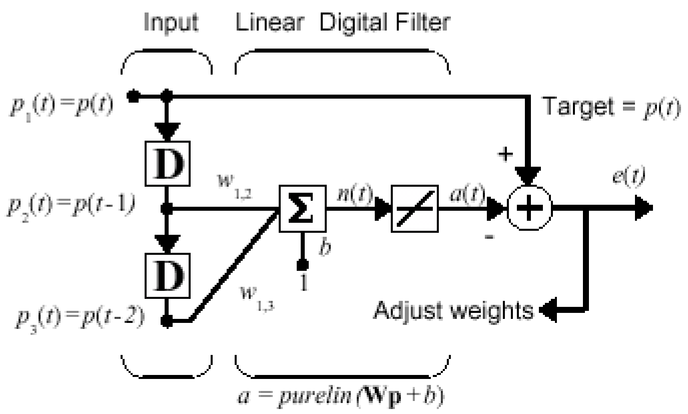
- Adaptive Filter





$$a(k) = purelin(\mathbf{W}\mathbf{p} + b) = \sum_{i=1}^{R} w_{1,i}\, a(k-i+1) + b$$

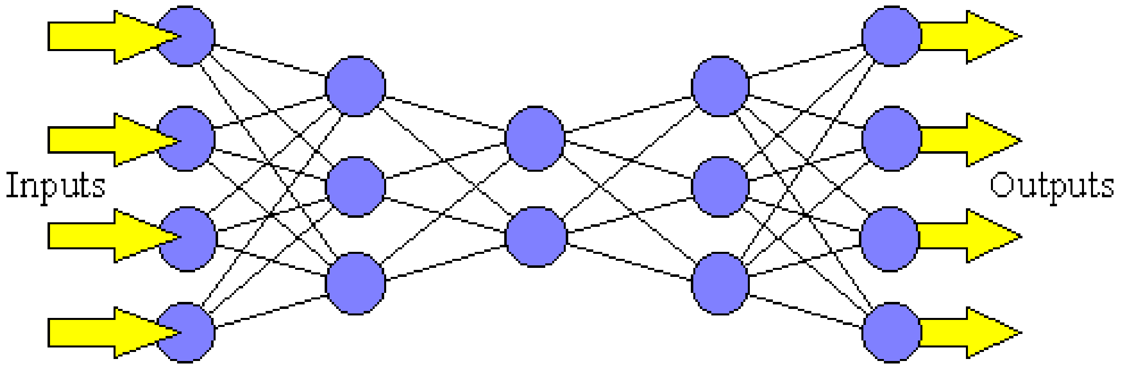
- Prediction example



Predictive Filter: $a(t)$ is approximation to $p(t)$

Auto-associative NNs

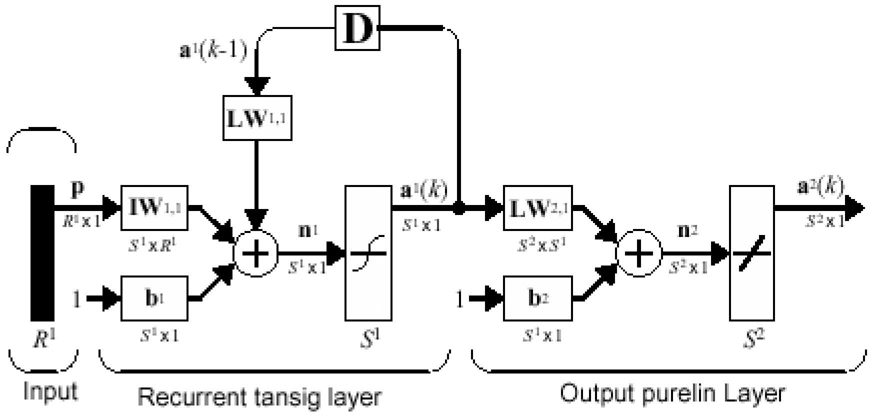
The auto-associative neural network is a special kind of MLP - in fact, it normally consists of two MLP networks connected "back to back‟. The other distinguishing feature of auto-associative networks is that they are trained with a target data set that is identical to the input data set.



In training, the network weights are adjusted until the outputs match the inputs, and the values assigned to the weights reflect the relationships between the various input data elements. This property is useful in, for example, data validation: when invalid data is presented to the trained neural network, the learned relationships no longer hold and it is unable to reproduce the correct output. Ideally, the match between the actual and correct outputs would reflect the closeness of the invalid data to valid values. Auto-associative neural networks are also used in data compression applications.

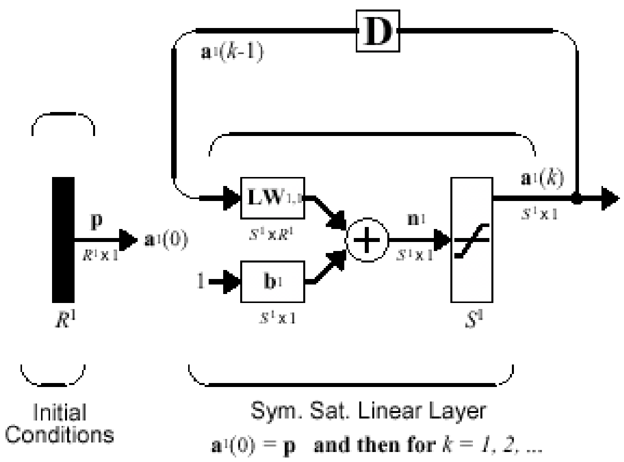
Recurrent Networks

- Elman Networks



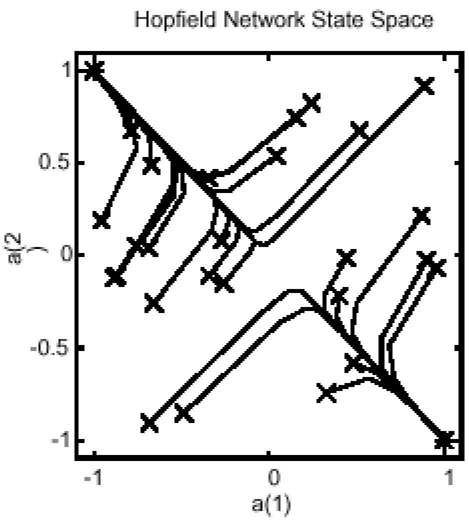
$\mathbf{a}^1(k) = \mathbf{tansig}(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{LW}^{1,1}\mathbf{a}^1(k-1) + \mathbf{b}^1)$ $\quad$ $\mathbf{a}^2(k) = \mathbf{purelin}(\mathbf{LW}^{2,1}\mathbf{a}^1(k) + \mathbf{b}^2)$

- Hopfield



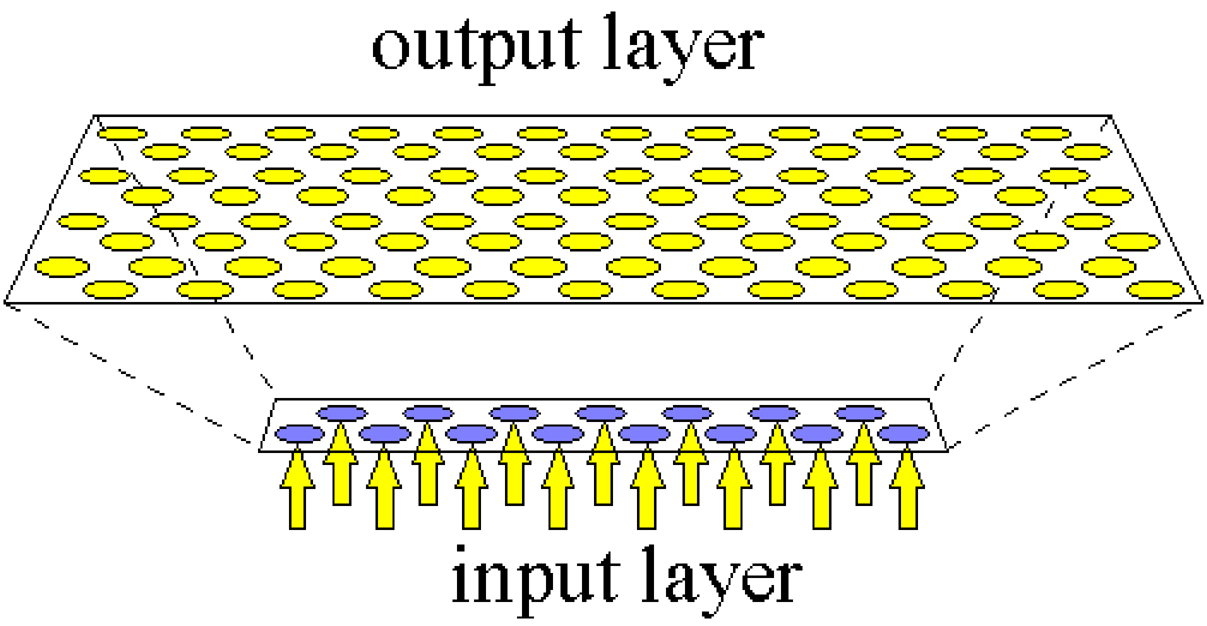
$\mathbf{a}^1(0) = \mathbf{p}$ and then for $k = 1, 2, \ldots$

$\mathbf{a}^1(k) = \mathbf{satlins}(\mathbf{LW}^{1,1}\mathbf{a}^1(k-1)) + \mathbf{b}^1)$

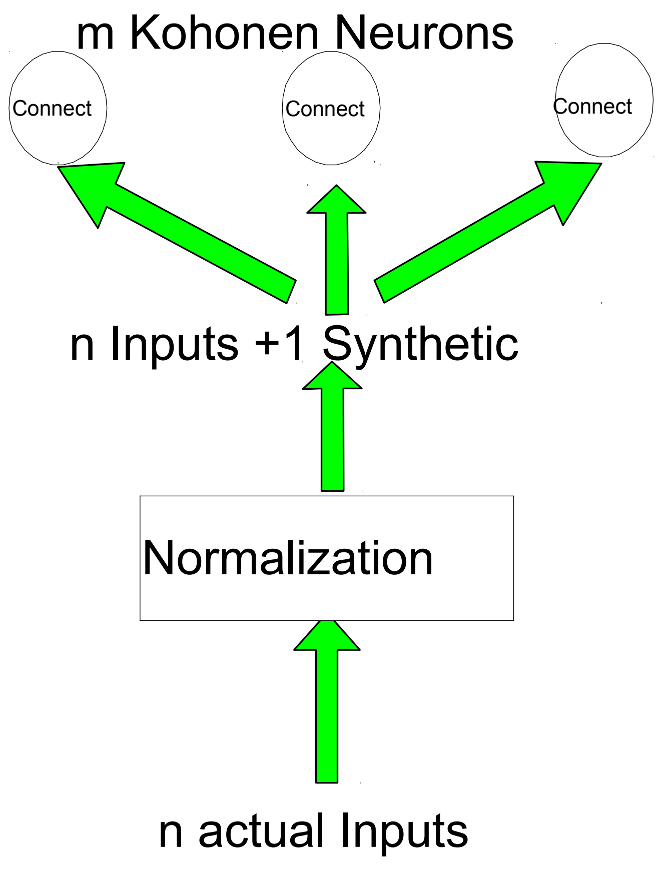


Self Organising Maps (Kohonen)

- The Self Organising Map or Kohonen network uses unsupervised learning.
- Kohonen networks have a single layer of units and, during training, clusters of units become associated with different classes (with statistically similar properties) that are present in the training data. The Kohonen network is useful in clustering applications.



Normalization



- Normalization

$$\|x\| = \sum_i x_i^2 \cong 1$$

Inputs must be in a hyperdimension sphere
The dimension shinks from n to n-1. (-2,1,3) and (-4,2,6) becomes the same.

- Composite inputs

The classical method

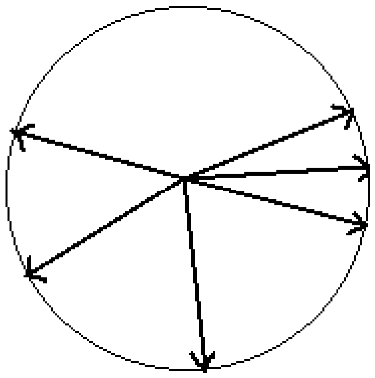
$$\ell = \sqrt{\sum_i x_i^2}$$

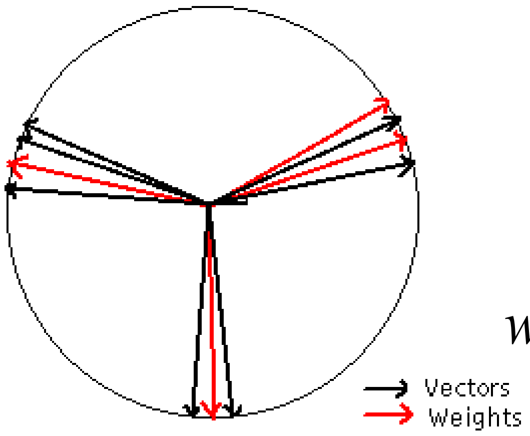
$$\widehat{x_i} = \frac{x_i}{\ell} \in [-1,+1]$$

z-Axis Normalization

$$f = \frac{1}{\sqrt{n}} \qquad \widehat{x_i} = f \cdot x_i \qquad s = f \cdot \sqrt{n - \ell^2}$$

Learning procedure





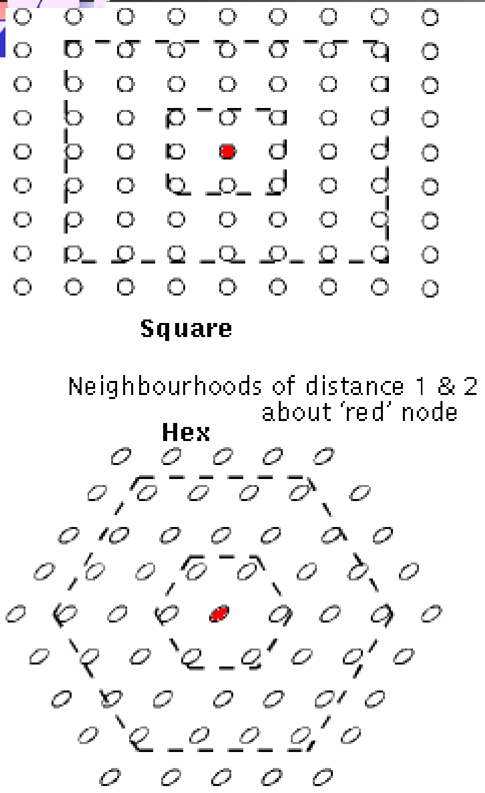
- In the begging the weights take random values.
- For an input vector we declare the winning neuron.
- Weights are changing in winner neighborhood.
- Iterate till balance.

Basic Math Relations

$$w_c^{(t+1)} = w_c^{(t)} + h_{ci}^{(t)} \left[x^{(t)} - w_c^{(t)} \right]$$

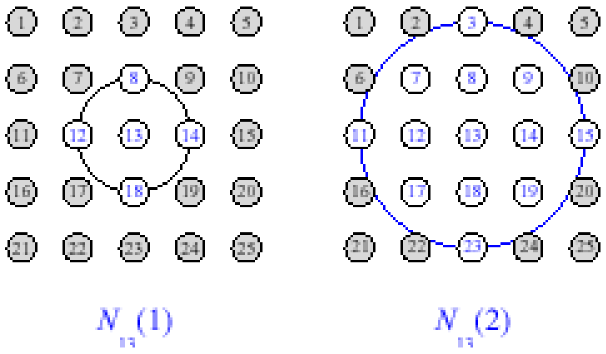
$$d_j = \sum_i \left(o_i^{(t)} - w_{ij}^{(t)} \right)^2$$

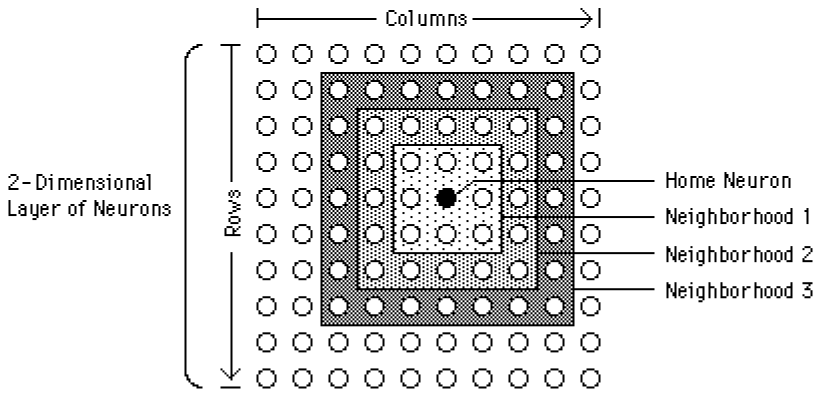
Neighborhood kernel function



$$h_{ci} = \alpha(t) e^{-\frac{\|r_c - r_i\|}{2\sigma^2(t)}}$$

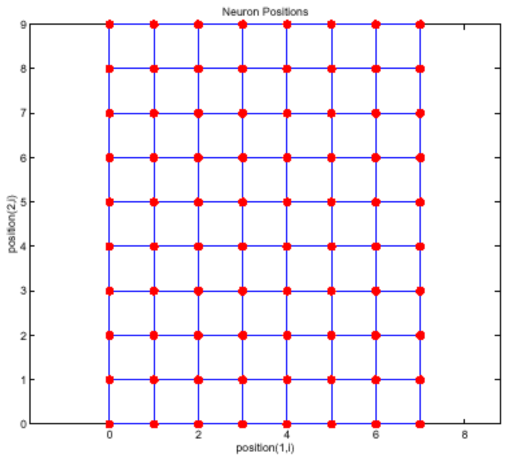
$$\alpha(t) = \frac{A}{B+t}$$

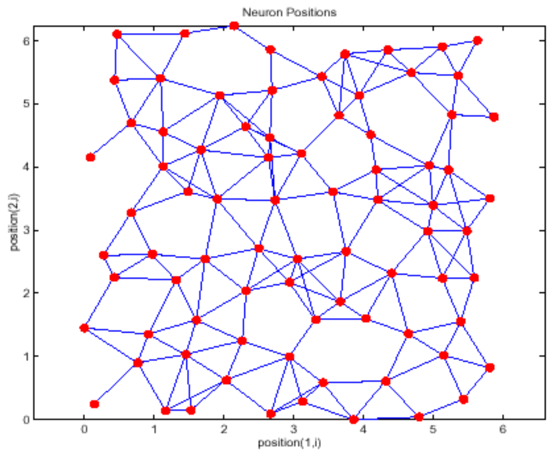


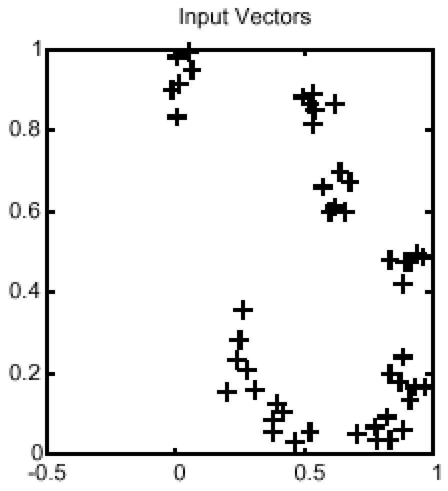


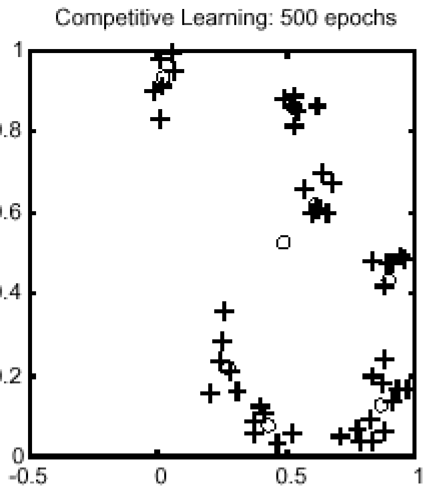
Γι

Self Organizing Maps









Introduction To Neural Networks

Prof. George Papadourakis, Ph.D.

Part II
Application Development And Portofolio

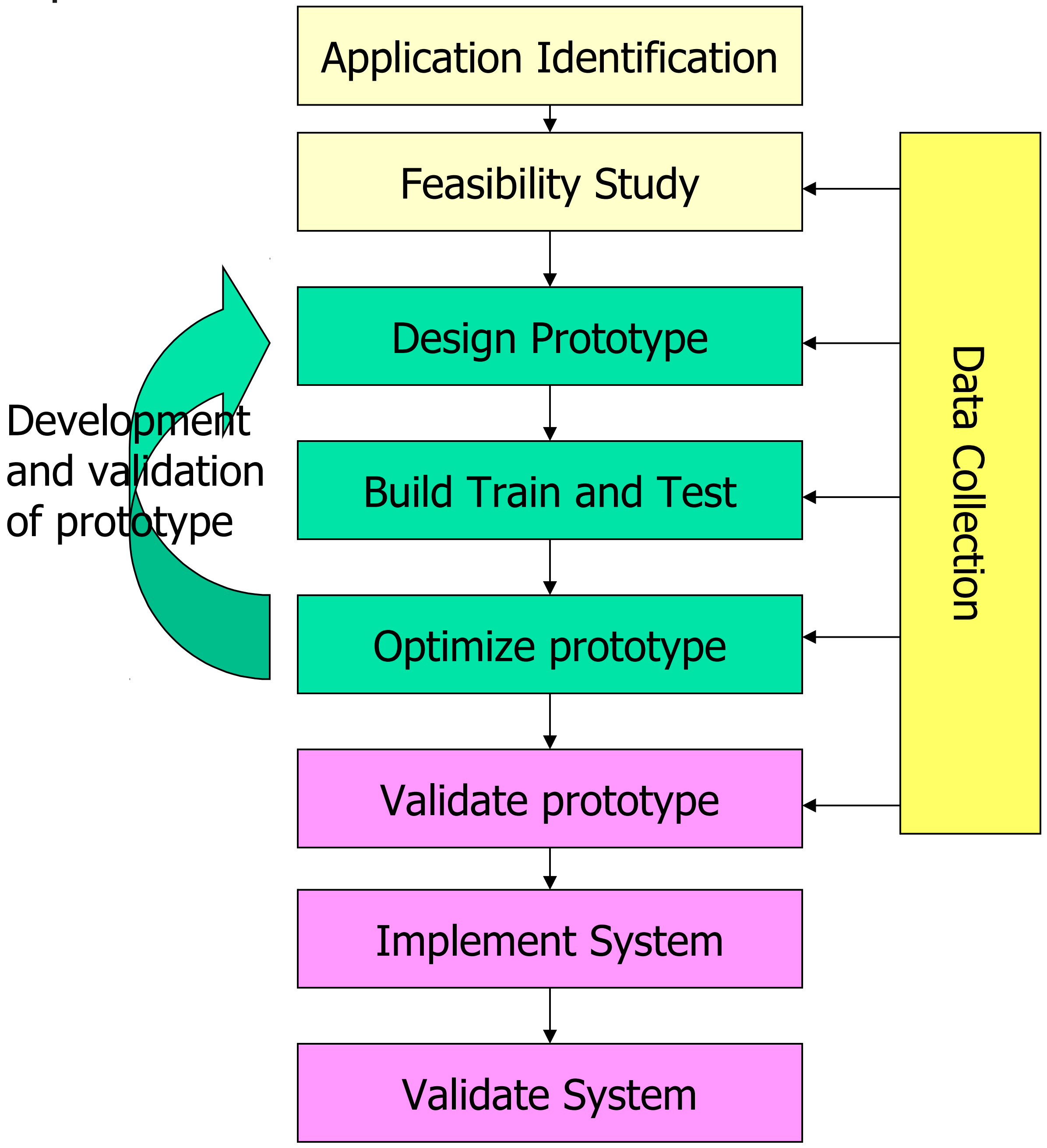
Characteristics of NNs

- 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

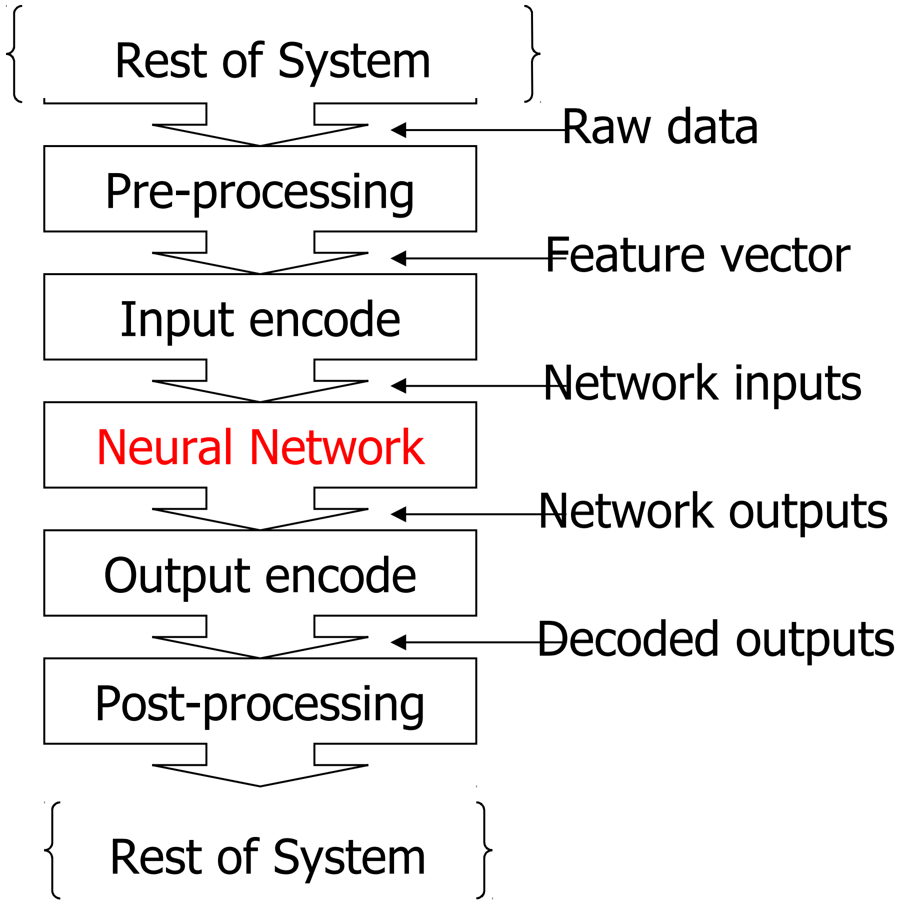
Neural Networks Projects Are Different

- 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

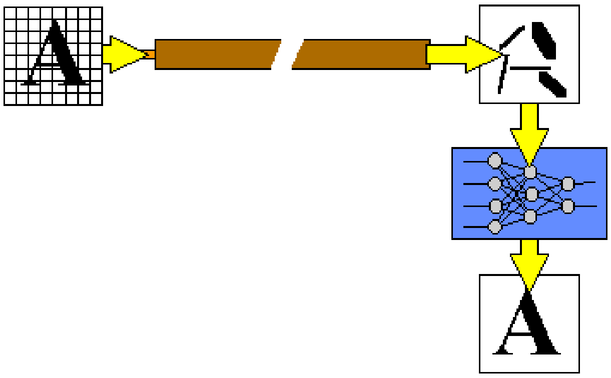


Pre-processing

- 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

Fibre Optic Image Transmission

Transmitting image without the distortion



In addition to transmitting data fiber optics, they also offer a potential for transmitting images. Unfortunately images transmitted over long distance fibre optic cables are more susceptible to distortion due to noise.

A large Japanese telecommunications company decided to use neural computing to tackle this problem. Rather than trying to make the transmission line as perfect and noise-free as possible, they used a neural network at the receiving end to reconstruct the distorted image back into its original form.

Related Applications : Recognizing Images from Noisy data

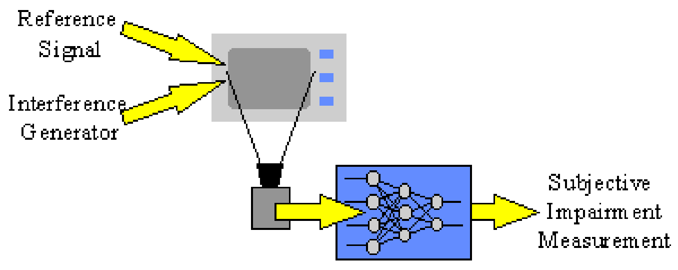
- Speech recognition
- Facial identification
- Forensic data analysis
- Battlefield scene analysis

TV Picture Quality Control

Assessing picture quality

One of the main quality controls in television manufacture is, a test of picture quality when interference is present. Manufacturers have tried to automate the tests, firstly by analysing the pictures for the different factors that affect picture quality as seen by a customer, and then by combining the different factors measured into an overall quality assessment. Although the various factors can be measured accurately, it has proved very difficult to combine them into a single measure of quality because they interact in very complex ways.

Neural networks are well suited to problems where many factors combine in ways that are difficult to analyse. ERA Technology Ltd, working for the UK Radio Communications Agency, trained a neural network with the results from a range of human assessments. A simple network proved easy to train and achieved excellent results on new tests. The neural network was also very fast and reported immediately



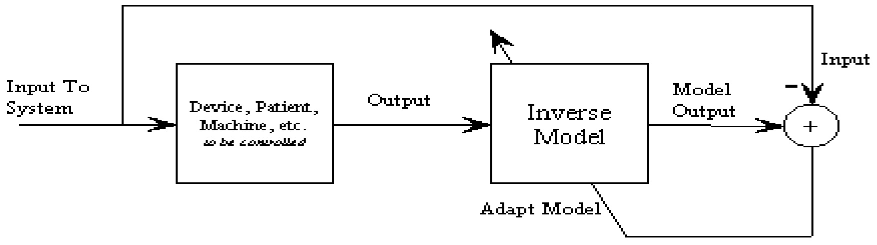
The neural system is able to carry out the range of required testing far more quickly than a human assessor, and at far lower cost. This enables manufacturers to increase the sampling rate and achieve higher quality, as well as reducing the cost of their current level of quality control.

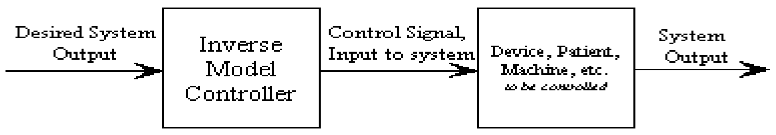
Related Applications : Signal Analysis

- Testing equipment for electromagnetic compatibility (EMC)
- Testing faulty equipment
- Switching car radios between alternative transmitters

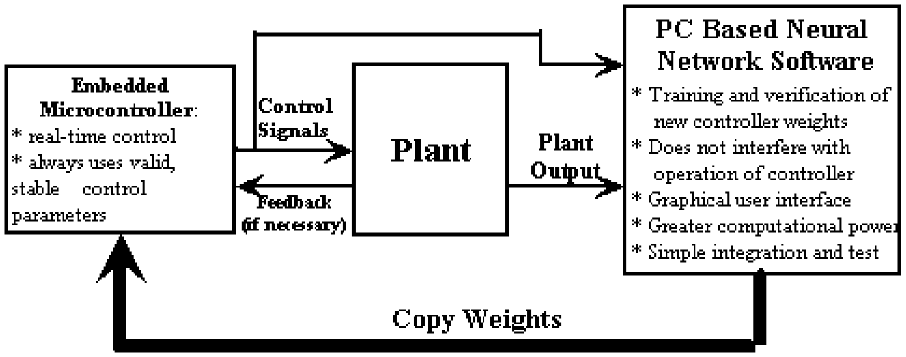
Adaptive Inverse Control

NNs can be used in adaptive control applications. The top block diagram shows the training of the inverse model. Essentially, the neural network is learning to recreate the input that created the current output of the plant. Once properly trained, the inverse model (which is another NN) can be used to control the plant since it can create the necessary control signals to create the desired system output.




Desired System Output
Inverse Model Controller
Control Signal, Input to system
Device, Patient, Machine, etc. to be controlled
System Output

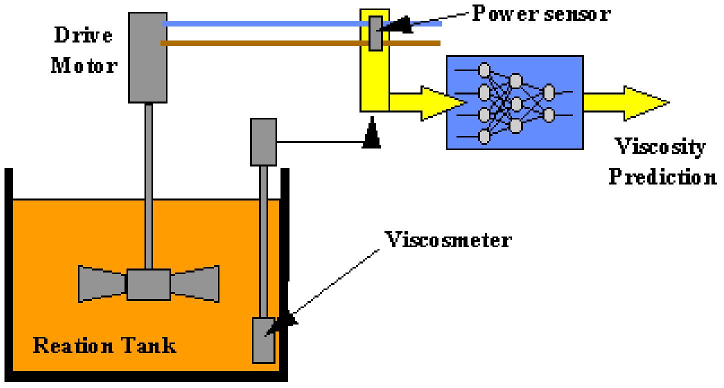
Block diagram for neural network adaptive control


Embedded Microcontroller:
* real-time control
* always uses valid, stable control parameters
Control Signals
Feedback (if necessary)
Plant
Plant Output
PC Based Neural Network Software
* Training and verification of new controller weights
* Does not interfere with operation of controller
* Graphical user interface
* Greater computational power
* Simple integration and test
Copy Weights

A computerized system for adaptive control

Chemical Manufacture

Getting the right mix



In a chemical tank various catalysts are added to the base ingredients at differing rates to speed up the chemical processes required. Viscosity has to be controlled very carefully, since inaccurate control leads to poor quality and hence costly wastage

The system was trained on data recorded from the production line. Once trained, the neural network was found to be able to predict accurately over the three-minute measurement delay of the viscometer, thereby providing an immediate reading of the viscosity in the reaction tank. This predicted viscosity will be used by a manufacturing process computer to control the polymerisation tank.

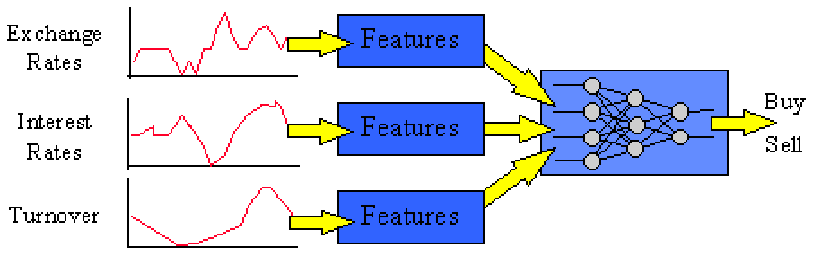
A more effective modelling tool

- Speech recognition (signal analysis)
- Environmental control
- Power demand analysis

Stock Market Prediction

Improving portfolio returns

A major Japanese securities company decided to user neural computing in order to develop better prediction models. A neural network was trained on 33 months' worth of historical data. This data contained a variety of economic indicators such as turnover, previous share values, interest rates and exchange rates. The network was able to learn the complex relations between the indicators and how they contribute to the overall prediction. Once trained it was then in a position to make predictions based on "live" economic indicators.



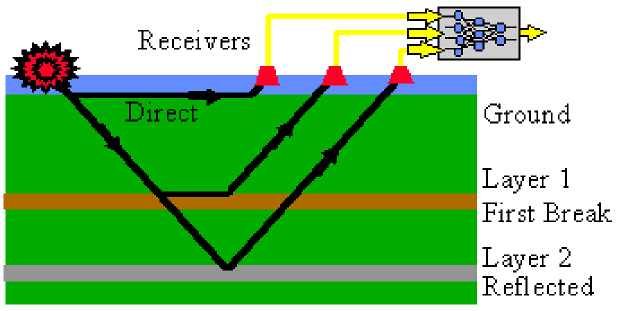
The neural network-based system is able to make faster and more accurate predictions than before. It is also more flexible since it can be retrained at any time in order to accommodate changes in stock market trading conditions. Overall the system outperforms statistical methods by a factor of 19%, which in the case of a £1 million portfolio means a gain of £190,000. The system can therefore make a considerable difference on returns.

Making predictions based on key indicators

- Predicting gas and electricity supply and demand
- Predicting sales and customer trends
- Predicting the route of a projectile
- Predicting crop yields

Oil Exploration

Getting the right signal



The vast quantities of seismic data involved are cluttered with noise and are highly dependent on the location being investigated. Classical statistical analysis techniques lose their effectiveness when the data is noisy and comes from an environment not previously encountered. Even a small improvement in correctly identifying first break signals could result in a considerable return on investment.

A neural network was trained on a set of traces selected from a representative set of seismic records, each of which had their first break signals highlighted by an expert.

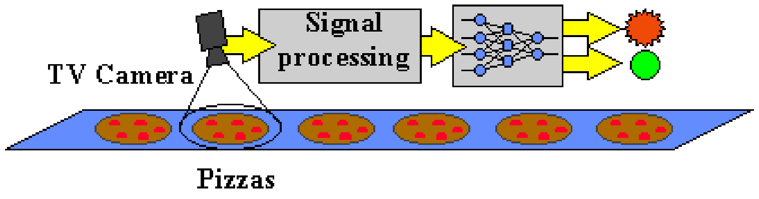
The neural network achieves better than 95 % accuracy, easily outperforming existing manual and computer-based methods. As well as being more accurate, the system also achieves an 88% improvement in the time taken to identify first break signals. Considerable cost savings have been made as a result.

Analysing signals buried in background noise

- Defence radar and sonar analysis
- Medical scanner analysis
- Radio astronomy signal analysis

Automated Industrial Inspection

Making better pizza



The design of an industrial inspection system is specific to a particular task and product, such as examining a particular kind of pizza. If the system was required to examine a different kind of pizza then it would need to be completely re-engineered. These systems also require stable operating environments, with fixed lighting conditions and precise component alignment on the conveyer belt.

A neural network was trained by personnel in the Quality Assurance Department to recognise different variations of the item being inspected. Once trained, the network was then able to identify deviant or defective items.

If requirements change, for example the need to identify a different kind of ingredient in a pizza or the need to handle a totally new type of pizza altogether, the neural network is simply retrained. There is no need to perform a costly system re-engineering exercise. Costs are therefore saved in system maintenance and production line down time.

Automatic inspection of components

- Inspecting paintwork on cars
- Checking bottles for cracks
- Checking printed circuit boards for surface defects

.

A Brief Introduction To Neural Networks

Prof. George Papadourakis Phd

Part III
Neural Networks Hardware

Hardware vs Software

- Implementing your Neural Network in special hardware can entail a substantial investment of your time and money:
  - the cost of the hardware
  - cost of the software to execute on the hardware
  - time and effort to climb the learning curve to master the use of the hardware and software.
- Before making this investment, you would like to be sure it is worth it.
- A scan of applications in a typical NNW conference proceedings will show that many, if not most, use feedforward networks with 10-100 inputs, 10-100 hidden units, and 1-10 output units.
- A forward pass through networks of this size will run in millisecs on a Pentium.
- Training may take overnight but if only done once or occasionally, this is not usually a problem.
- Most applications involve a number of steps, many not NNW related, that cannot be made parallel. So Amdahl's law limits the overall speedup from your special hardware.
- Intel 86 series chips and other von Neuman processors have grown rapidly in speed, plus one can take advantage of huge amount of readily available software.
- One quickly begins to see why the business of Neural Network hardware has not boomed the way some in the field expected back in the 1980's.

Applications of Hardware NNWs

While not yet as successful as NNWs in software, there are in fact hardware NNW's hard at work in the real world. For example:

- OCR (Optical Character Recognition)
  - Adaptive Solutions high volume form and image capture systems.
  - Ligature Ltd. OCR-on-a-Chip
- Voice Recognition
  - Sensory Inc. RSC Microcontrollers and ASSP speech recognition specific chips.
- Traffic Monitoring
  - Nestor TrafficVision Systems
- High Energy Physics
  - Online data filter at H1 electon-proton collider experiment in Hamburg using Adaptive Solutions CNAPS boards.

However, most NNW applications today are still run with conventional software simulation on PC's and workstations with no special hardware add-ons.

NNets in VLSI

Neural networks are parallel devices, but usually is implement in traditional Von Neuman architectures. There is also exist Hardware implementations of NNs.Such hardware includes digital and analog hardware chips, PC accelerator boards, and multi-board neurocomputers.

- Digital
- Slice Architectures
  - Multi-processor Chips
  - Radial Basis Functions
  - Other Digital Designs
- Analog
- Hybrid
- Optical hardware

NNW Features

- Neural Network architecture(s)
- Programmable or hardwired network(s)
- On-chip learning or chip-in-the-loop training
- Low, medium or high number of parallel processing elements (PE's)
- Maximum network size.
- Can chips be chained together to increase network size.
- Bits of precision (estimate for analog)
- Transfer function on-chip or off-chip, e.g. in lookup table (LUT).
- Accumulator size in bits.
- Expensive or cheap

NeuroComputers

- **Neurocomputers are defined here as standalone systems with elaborate hardware and software.**
- **Examples:**
  - **Siemens Synapse 1 Neurocomputer:**
    - **Uses 8 of the MA-16 systolic array chips.**
    - **It resides in its own cabinet and communicates via ethernet to a host workstation.**
    - **Peak performance of 3.2 billion multiplications (16-bit x 16-bit) and additions (48-bit) per sec. at 25MHz clock rate.**
  - **Adaptive Solutions - CNAPServer VME System**
    - VME boards in a custom cabinet run from a UNIX host via an ethernet link.
    - Boards come with 1 to 4 chips and up to two boards to give a total of 512 PE's.
    - Software includes a C-language library, assembler, compiler, and a package of NN algorithms.

Analog & Hybrid NNW Chips

- Analog advantages:
  - Exploit physical properties to do network operations, thereby obtain high speed and densities.
  - A common output line, for example, can sum current outputs from synapses to sum the neuron inputs.
- Analog disadvantages
  - Design can be very difficult because of the need to compensate for variations in manufacturing, in temperature, etc.
  - Analog weight storage complicated, especially if non-volatility required.
  - Weight*input must be linear over a wide range.
- Hybrids combine digital and analog technology to attempt to get the best of both. Variations include:
  - Internal processing analog for speed but weights set digitally, e.g. capacitors refreshed periodically with DAC's.
  - Pulse networks use rate or widths of pulses to emulate amplitude of I/O and weights.

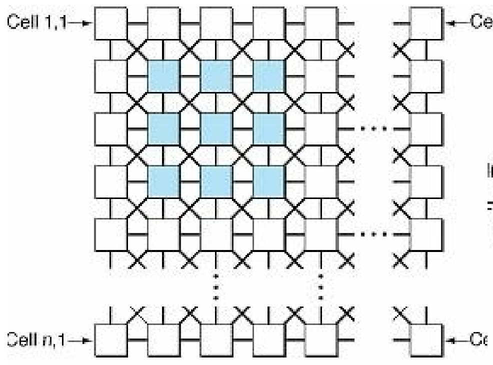
NNW Accelerator Cards

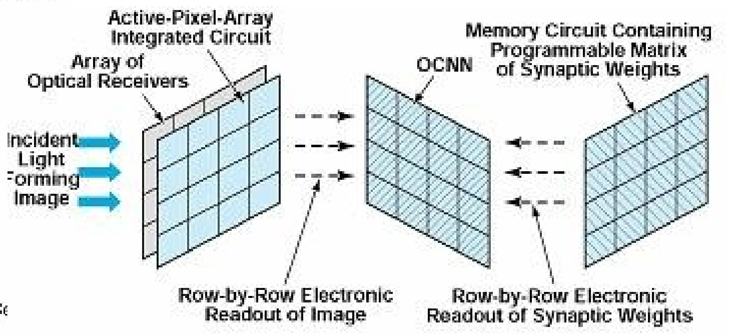
- Another approach to dealing with the PC, is to work with it in partnership.
- Accelerator cards reside in the expansion slots and are used to speed up the NNW computations.
- Cheaper than NeuroComputers.
- Usually based on NNW chips but some just use fast digital signal processors (DSP) that do very fast multiple-accumulate operations.
- Examples:
  - IBM ZISC ISA and PCI Cards:
    - ZISC implements a RBF architecture with RCE learning (more ZISC discussion later.)
    - ISA card holds to 16 ZISC036 chips, giving 576 prototype neurons.
    - PCI card holds up to 19 chips for 684 prototypes.
    - PCI card can process 165,000 patterns/sec, where patterns are 64 8-bit element vectors.
  - California Scientific CNAPS accelerators:
    - Runs with CalSci's popular BrainMaker NNW software.
    - With either 4 or 8 chips (16-PE/chip) to give 64 or 128 total PEs.
    - Up to 2.27GCPS. See their Benchmarks
    - Speeds can vary depending on transfer speeds of particular machines.
    - Hardware and software included
  - DataFactory NeuroLution PCI Card:
    - contains up to four SAND/1 neurochips.
    - Cascadable SAND neurochips use a systolic architecture to do fast 4x4 matrix multiplies and accumulates.
    - Four parallel 16 bit multipliers and eight 40 bit adders execute in one clock cycle. The clock rate is 50 Mhz.
    - With 4 chips peak performance of the board is 800 MCPS.
    - Used with the NeuoLution Manager and Connect scripting language.
    - Feedforward neural networks with a maximum of 512 input neurons and three hidden layers.
    - The activation function of the neurons can be programmed in a lookup table.
    - Kohonen feature maps and radial basis function networks also implemented.

OCNNs inVLSI

Optimization cellular neural network (OCNN) can be implemented VLSI. The OCNN concept is founded on the concept of the cellular neural network (CNN), which is a recursive neural network that comprises a multidimensional array of mainly identical artificial neural cells, wherein

1. Each cell is a dynamic subsystem with continuous state variables
2. Each cell is connected to only the few other cells that lie within a specified radius




Active-Pixel-Array Integrated Circuit
Array of Optical Receivers
Incident Light Forming Image
OCNN
Memory Circuit Containing Programmable Matrix of Synaptic Weights
Row-by-Row Electronic Readout of Image
Row-by-Row Electronic Readout of Synaptic Weights

A **Typical *n*-by-*m* Rectangular Cellular Neural Network** contains cells that are connected to their nearest neighbors only.

A **"Smart" Optoelectronic Image Sensor** could include an OCNN sandwiched between a planar array of optical receivers and a planar array of optical transmitters, along with circuitry that would implement a programmable synaptic-weight matrix memory. This combination of optics and electronics would afford fast processing of sensory information within the sensor package.