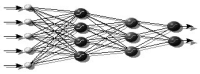
Artificial Neural Networks

- The Brain
- Brain vs. Computers
- The Perceptron
- Multilayer networks
- Some Applications

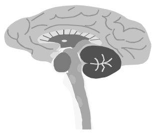


Artificial Neural Networks

- Other terms/names
  - connectionist
  - parallel distributed processing
  - neural computation
  - adaptive networks..
- History
  - 1943-McCulloch & Pitts are generally recognised as the designers of the first neural network
  - 1949-First learning rule
  - 1969-Minsky & Papert - perceptron limitation - Death of ANN
  - 1980's - Re-emergence of ANN - multi-layer networks

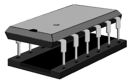
Brain and Machine

- The Brain
  - Pattern Recognition
  - Association
  - Complexity
  - Noise Tolerance

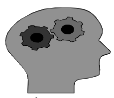


- The Machine
  - Calculation
  - Precision
  - Logic

The contrast in architecture

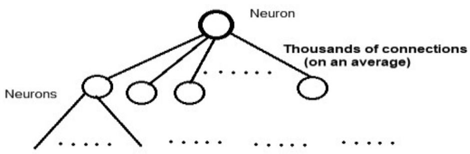


- The Von Neumann architecture uses a single processing unit;
  - Tens of millions of operations per second
  - Absolute arithmetic precision
- The brain uses many slow unreliable processors acting in parallel



Features of the Brain

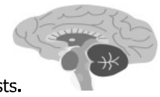
- Ten billion (10^{10}) neurons
- On average, several thousand connections
- Hundreds of operations per second



Features of the Brain

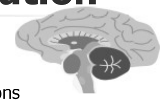
- Die off frequently (never replaced)
- Compensates for problems by massive parallelism

The biological inspiration



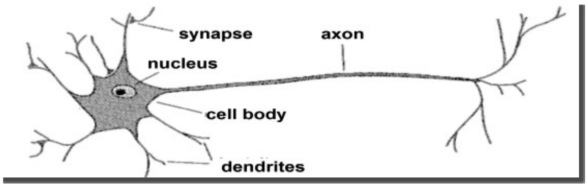
- The brain has been extensively studied by scientists.
- Vast complexity prevents all but fundamental understanding.
- Even the behaviour of an individual neuron is extremely complex

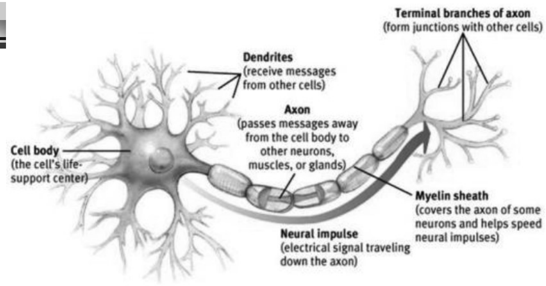
The biological inspiration



- Single "percepts" distributed among many neurons
- Localized parts of the brain are responsible for certain well-defined functions (e.g. vision, motion).
- Which features are integral to the brain's performance?
- Which are incidentals imposed by the fact of biology?

The Structure of Neurons





The Structure of Neurons

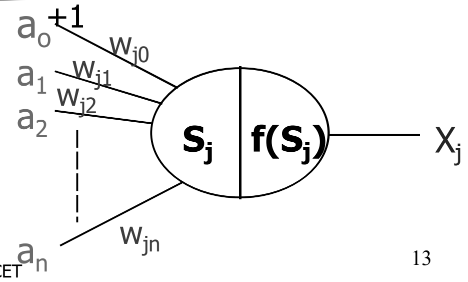
A neuron has a cell body, a branching **i**nput structure (the dendr**I**te) and a branching **o**utput structure (the ax**O**n)

- Axons connect to dendrites via synapses.
- Electro-chemical signals are propagated from the dendritic input, through the cell body, and down the axon to other neurons

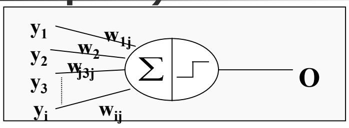
The Structure of Neurons

- A neuron only fires if its input signal exceeds a certain amount (the threshold) in a short time period.
- Synapses vary in strength
  - Good connections allowing a large signal
  - Slight connections allow only a weak signal.
  - Synapses can be either excitatory or inhibitory.

The Artificial Neuron (Perceptron)



A Simple Model of a Neuron (Perceptron)



- Each neuron has a threshold value
- Each neuron has weighted inputs from other neurons
- The input signals form a weighted sum
- If the activation level exceeds the threshold, the neuron "fires"

A Theory of Memory

Hopfield [1982] introduced a neural network that he proposed as a theory of memory. A Hopfield network has the following interesting features:

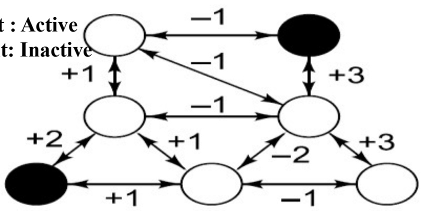
- Distributed Representation
- Distributed, Asynchronous
- Content-Addressable Memory
- Fault Tolerance

A Theory of Memory

- **Distributed Representation** : A memory is stored as a pattern across a set of processing elements.
- **Distributed, Asynchronous** : each processing elements makes decisions based only on its own local situation.
- **Content-Addressable Memory** : A number of patterns can be stored in a network.
- **Fault Tolerance** : if few processing elements fails, the network functions properly.

A Simple Hopfield Network

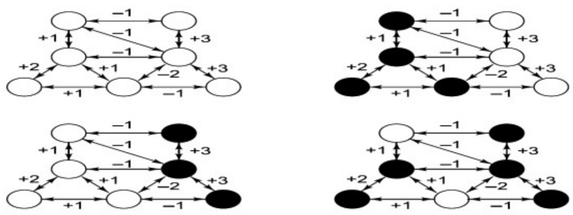
Black Unit : Active
White Unit: Inactive



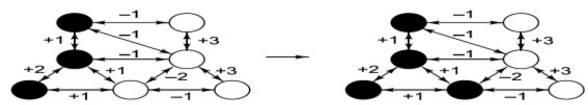
Unites are connected via Weighted Symmetric Connections.

- Positive connection tends to activate unit each other
- How is network operated
  - A random unit is chosen
  - If any of its neighbours are active, the unit computes the sum of the weights on the connections to those active neighbours
    - If the sum is positive, the unit becomes active
    - Otherwise it becomes inactive.
  - Another Random unit is chosen and the process is repeated until no more units can change state
    - This process is called as **Parallel Relaxation**

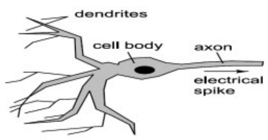
The Four Stable States of a Particular Hopfield Net

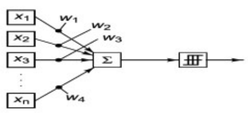


A Hopfield Net as a Model of Content-Addressable Memory

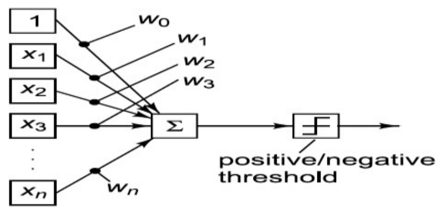


A Neuron and a Perceptron

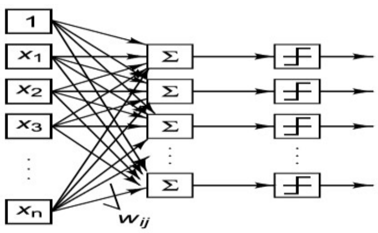




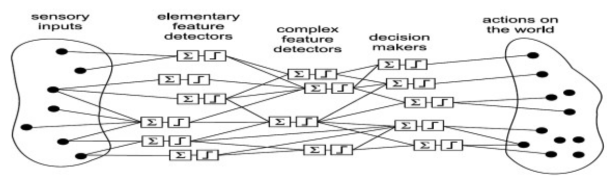
Perceptron with Adjustable Threshold Implemented as Additional Weight



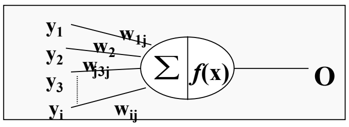
A Perceptron with Many Inputs and Many Outputs



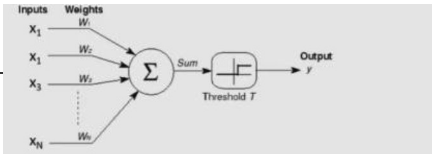
An Early Notion of an Intelligent System Built from Trainable Perceptrons



An Artificial Neuron



- Each hidden or output neuron has weighted input connections from each of the units in the preceding layer.
- The unit performs a weighted sum of its inputs, and subtracts its threshold value, to give its activation level.
- Activation level is passed through a sigmoid activation function to determine output.



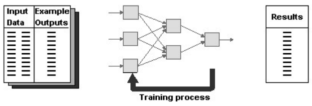
Number of inputs combine linearly

– Threshold logic: Fire if combined input exceeds threshold

$$Y = \begin{cases} 1 \ if \ \sum_i w_i x_i - T > 0 \\ 0 \quad\quad else \end{cases}$$

Supervised Learning

- Training and test data sets
- Training set; input & target



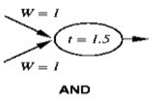
Sepal length	Sepal width	Petal length	Petal width	Class
5.1	3.5	1.4	0.2	0
4.9	3.0	1.4	0.2	2
4.7	3.2	1.3	0.2	0
4.6	3.1	1.5	0.2	1

What is required for ANN sol

For any problem solving using ANN

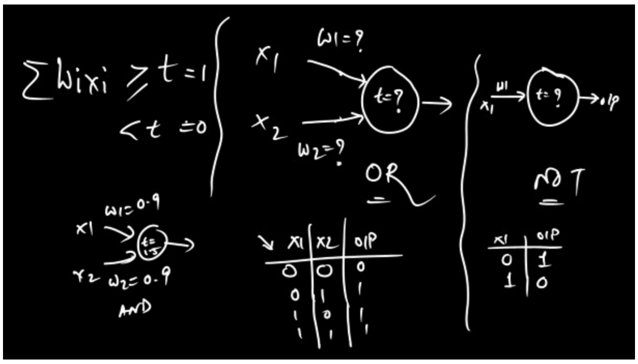
- There are two aspects
  1. Network Design (structure)
  2. Weights

Perceptron Training

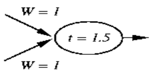


$$\text{Output} = \begin{cases} 1 \text{ if } \sum w_i x_i > t \\ 0 \text{ otherwise} \end{cases}$$

- Linear threshold is used.
- W - weight value
- t - threshold value



Perceptron Training



AND OR NOT

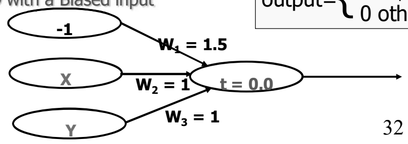
$$\text{Output}=\begin{cases}1 \text{ if } \sum w_i x_i > t\\ 0 \text{ otherwise}\end{cases}$$

- Linear threshold is used.
- W - weight value
- t - threshold value

Simple network

$$\text{output}=\begin{cases}1 \text{ if } \sum_{i=0} w_i x_i > t\\ 0 \text{ otherwise}\end{cases}$$

AND with a Biased input



Learning algorithm

While epoch produces an error

Present network with next inputs from epoch

Error = T – O

If Error <> 0 then

$W_j = W_j + LR * I_j * Error$

End If

End While

Learning algorithm

Epoch : Presentation of the entire training set to the neural network. In the case of the AND function an epoch consists of four sets of inputs being presented to the network (i.e. [0,0], [0,1], [1,0], [1,1])

Error: The error value is the amount by which the value output by the network differs from the target value. For example, if we required the network to output 0 and it output a 1, then Error = -1

Learning algorithm

Target Value, T : When we are training a network we not only present it with the input but also with a value that we require the network to produce. For example, if we present the network with [1,1] for the AND function the target value will be 1

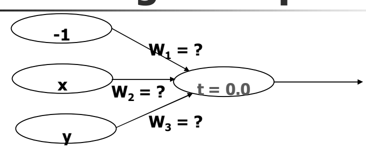
Output , O : The output value from the neuron

Ij : Inputs being presented to the neuron

Wj : Weight from input neuron (I_j) to the output neuron

LR : The learning rate. This dictates how quickly the network converges. It is set by a matter of experimentation. It is typically 0.1

Training Perceptrons

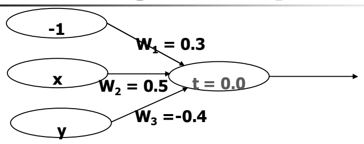


For AND

A B	Output
0 0	0
0 1	0
1 0	0
1 1	1

- What are the weight values?
- Initialize with random weight values

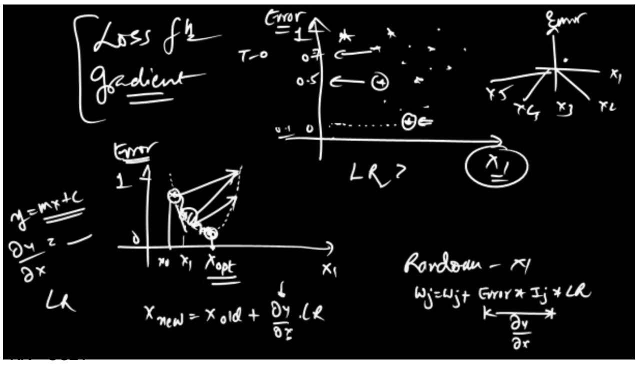
Training Perceptrons



For AND

A B	Output
0 0	0
0 1	0
1 0	0
1 1	1

I_1	I_2	I_3	Summation	Output
-1	0	0	(-1*0.3) + (0*0.5) + (0*-0.4) = -0.3	0
-1	0	1	(-1*0.3) + (0*0.5) + (1*-0.4) = -0.7	0
-1	1	0	(-1*0.3) + (1*0.5) + (0*-0.4) = 0.2	1
-1	1	1	(-1*0.3) + (1*0.5) + (1*-0.4) = -0.2	0



Example : AND Gate

Let.s take a look at an example. The initial weight values are 0.3, 0.5, and -0.4 (taken from the above example) and we are trying to learn the AND function.

If we present the network with the first training pair ([0,0]), from the first epoch, nothing will happen to the weights (due to multiplying by zero).

The next training pair ([0,1]) will result in the network producing zero (by virtue of the step0 function). As zero is the required output there is no error so training continues.

The next training pair ([1,0]) produces an output of one. The required output is 0. **Therefore the error is -1**

This means we have to adjust the weights.

This is done as follows (assuming LR = 0.1)

W0 = 0.3 + 0.1 * -1 * -1 = 0.4

W1 = 0.5 + 0.1 * 1 * -1 = 0.4

W2 = -0.4 + 0.1 * 0 * -1 = -.04

Therefore, the new weights are 0.4, 0.4, -0.4.

- Finally we apply the input [1,1] to the network. This also produces an error and the new weight values will be 0.3, 0.5 and -0.3.
- Training continues until an epoch is presented that does not produce an error.
- If we continue with the training until we get no errors from an entire epoch the weights would be 0.4, 0.4, 0.3.

Learning in Neural Networks

- Learn values of weights from I/O pairs
- Start with random weights
- Load training example's input
- Observe computed input
- Modify weights to reduce difference
- Iterate over all training examples
- Terminate when weights stop changing OR when error is very small

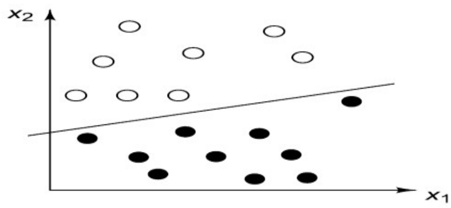
Neural Network

- OR Gate
- Ex-OR Gate

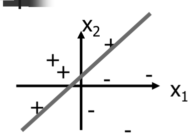
Decision boundaries

- In simple cases, divide feature space by drawing a hyperplane across it.
- Known as a decision boundary.
- Discriminant function: returns different values on opposite sides. (straight line)
- Problems which can be thus classified are linearly separable.

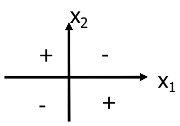
A Linearly Separable Pattern Classification Problem



Decision Surface of a Perceptron



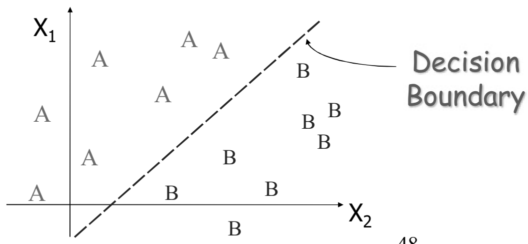
Linearly separable



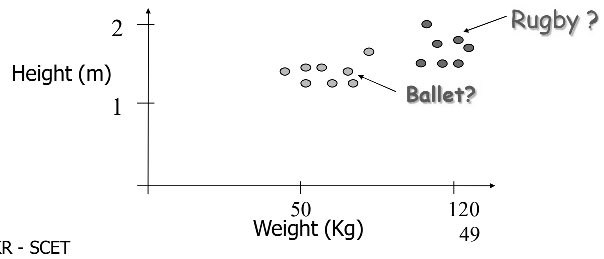
Non-Linearly separable

- Perceptron is able to represent some useful functions
- AND(x_1,x_2) choose weights w_0=-1.5, w_1=1, w_2=1
- But functions that are not linearly separable (e.g. XOR) are not representable

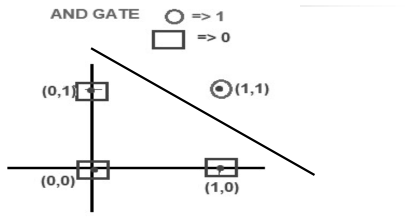
Linear Separability



Rugby players & Ballet dancers

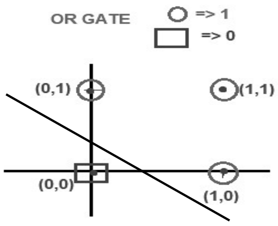


AND GATE



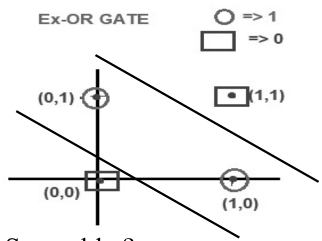
Linearly Separable ?

OR GATE



Linearly Separable ?

Ex-OR GATE

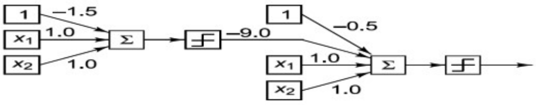


Linearly Separable ?

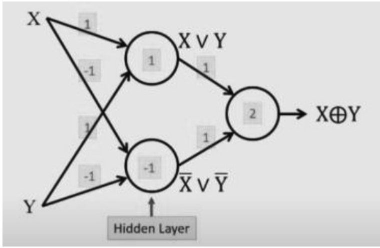
Hyperplane partitions

- A single Perceptron (i.e. output unit) with connections from each input can perform, and learn, a linear separation.
- Perceptrons have a step function activation.
- Units with a sigmoid activation also act as a linear discriminant, if interpreted correctly.
  - Use activation mid-point

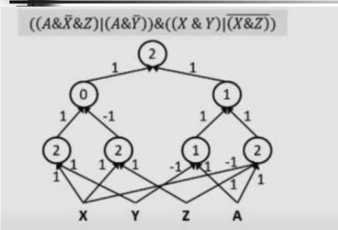
A Multilayer Perceptron That Solves the XOR Problem



A Multilayer Perceptron That Solves the XOR Problem

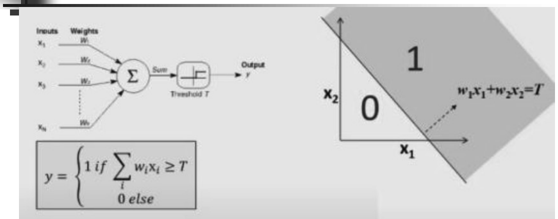


A more generic model

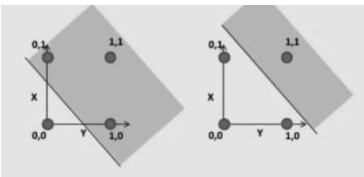


- A multi layer "perceptron"
- Can compose arbitrarily complicated Boolean functions

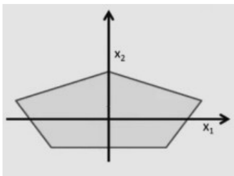
Perceptron on real value



Complicated decision boundaries

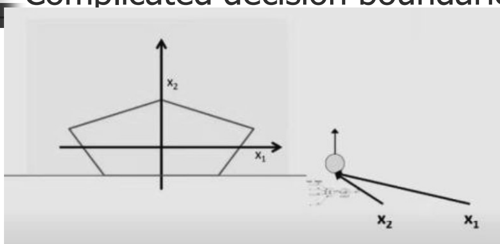


Complicated decision boundaries

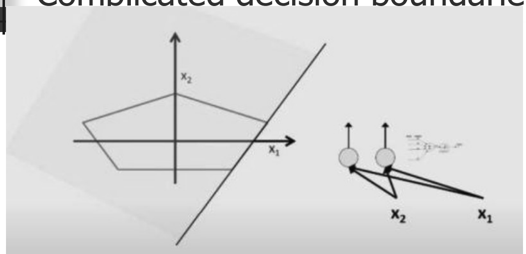


- Build a network of units with a single output that fires if the input is in the coloured area
- Can now be composed into "networks" to compute arbitrary classification "boundaries"

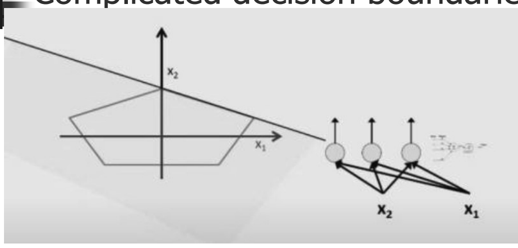
Complicated decision boundaries



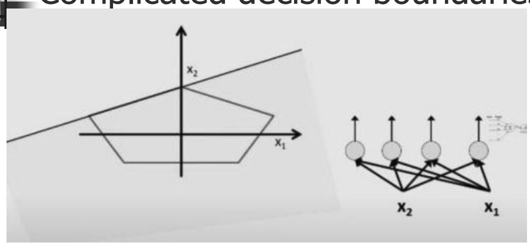
Complicated decision boundaries



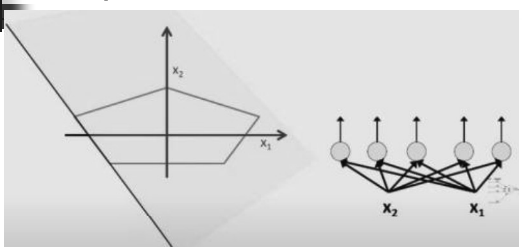
Complicated decision boundaries



Complicated decision boundaries



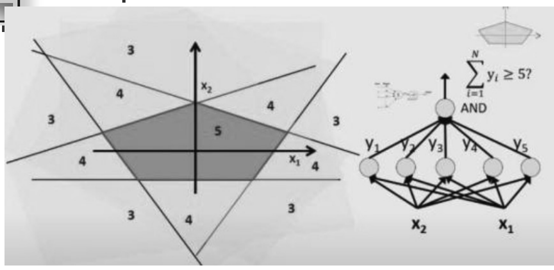
Complicated decision boundaries



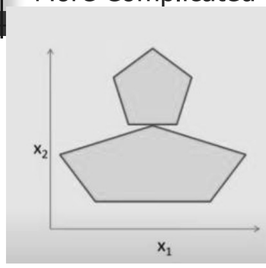
Complicated decision boundaries

- If all networks are added with a weight of 1 each, we get our desire solution

Complicated decision boundaries

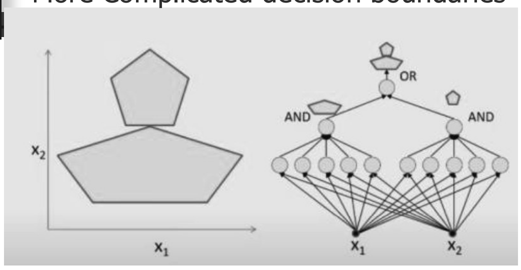


More Complicated decision boundaries



- Network to fire if the input is in the coloured area
  - OR two polygons
  - A Third Layer is required

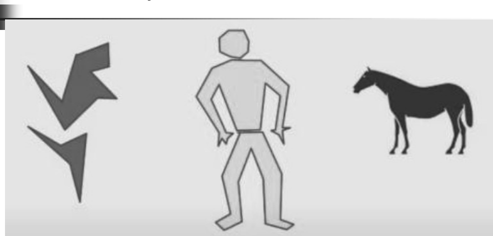
More Complicated decision boundaries



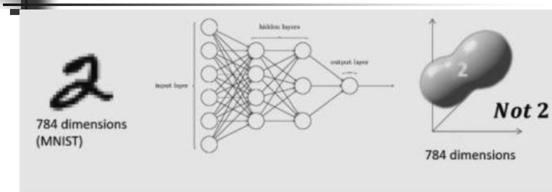
More Complicated decision boundaries

- If we can do these boundaries, we can also solve any complex problem in the same manner.

More Complicated decision boundaries



More Complicated decision boundaries

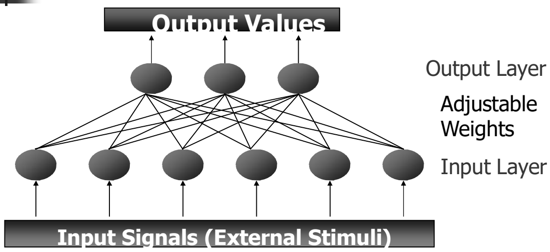


- Classification problems : Finding decision boundaries in high-dimension space
  - Can be performed by MLP

Different Non-Linearly Separable Problems

Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer	Half Plane Bounded By Hyperplane	A B B A	B A	
Two-Layer	Convex Open Or Closed Regions	A B B A	B A	
Three-Layer	Arbitrary (Complexity Limited by No. of Nodes)	A B B A	B A	

Multilayer Perceptron (MLP)

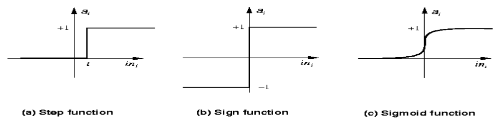


Types of Layers

- The input layer.
  - Introduces input values into the network.
  - No activation function or other processing.
- The hidden layer(s).
  - Perform classification of features
  - Two hidden layers are sufficient to solve any problem
  - Features imply more layers may be better
- The output layer.
  - Functionally just like the hidden layers
  - Outputs are passed on to the world outside the neural network.

Activation functions

- Transforms neuron's input into output.
- Features of activation functions:
  - A squashing effect is required
    - Prevents accelerating growth of activation levels through the network.
  - Simple and easy to calculate



Standard activation functions

- The hard-limiting threshold function
  - Corresponds to the biological paradigm
    - either fires or not
- Sigmoid functions ('S'-shaped curves)
  - The logistic function $\phi(x) = \frac{1}{1 + e^{-ax}}$
  - The hyperbolic tangent (symmetrical)
  - Both functions have a simple differential
  - Only the shape is important

Training Algorithms

- Adjust neural network weights to map inputs to outputs.
- Use a set of sample patterns where the desired output (given the inputs presented) is known.
- The purpose is to learn to generalize
  - Recognize features which are common to good and bad exemplars

Back-Propagation

- A training procedure which allows multi-layer feedforward Neural Networks to be trained;
- Can theoretically perform "any" input-output mapping;
- Can learn to solve linearly inseparable problems.

Activation functions and training

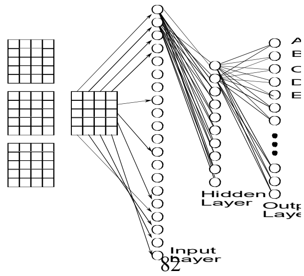
- For feed-forward networks:
  - A continuous function can be differentiated allowing gradient-descent.
  - Back-propagation is an example of a gradient-descent technique.
  - Reason for prevalence of sigmoid

Applications

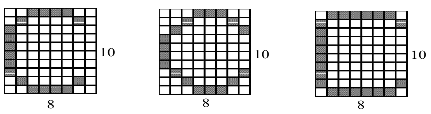
- The properties of neural networks define where they are useful.
  - Can learn complex mappings from inputs to outputs, based solely on samples
  - Difficult to analyse: firm predictions about neural network behaviour difficult;
    - Unsuitable for safety-critical applications.
  - Require limited understanding from trainer, who can be guided by heuristics.

Neural network for OCR

- feedforward network
- trained using Back-propagation



OCR for 8x10 characters



- NN are able to generalise
- learning involves generating a partitioning of the input space
- for single layer network input space must be linearly separable
- what is the dimension of this input space?
- how many points in the input space?
- this network is binary(uses binary values)
- networks may also be continuous

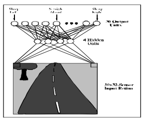
Engine management

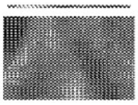
- The behaviour of a car engine is influenced by a large number of parameters
  - temperature at various points
  - fuel/air mixture
  - lubricant viscosity.
- Major companies have used neural networks to dynamically tune an engine depending on current settings.

ALVINN

Drives 70 mph on a public highway

30 outputs for steering
4 hidden units
30x32 pixels as inputs



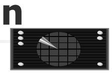


30x32 weights into one out of four hidden unit

Signature recognition

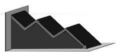
- Each person's signature is different.
- There are structural similarities which are difficult to quantify.
- One company has manufactured a machine which recognizes signatures to within a high level of accuracy.
  - Considers speed in addition to gross shape.
  - Makes forgery even more difficult.

Sonar target recognition



- Distinguish mines from rocks on sea-bed
- The neural network is provided with a large number of parameters which are extracted from the sonar signal.
- The training set consists of sets of signals from rocks and mines.

Stock market prediction



- “Technical trading” refers to trading based solely on known statistical parameters; e.g. previous price
- Neural networks have been used to attempt to predict changes in prices.
- Difficult to assess success since companies using these techniques are reluctant to disclose information.

Mortgage assessment

- Assess risk of lending to an individual.
- Difficult to decide on marginal cases.
- Neural networks have been trained to make decisions, based upon the opinions of expert underwriters.
- Neural network produced a 12% reduction in defaulters compared with human experts.

Neural Network Problems

- Many Parameters to be set
- Over fitting
- long training times
- ...

Parameter setting

- Number of layers
- Number of neurons
  - too many neurons, require more training time
- Learning rate
  - from experience, value should be small ~0.1
- Momentum term
- ..

Over-fitting

- With sufficient nodes can classify any training set exactly
- May have poor generalisation ability.
- Cross-validation with some patterns
  - Typically 30% of training patterns
  - Validation set error is checked each epoch
  - Stop training if validation error goes up

Training time

- How many epochs of training?
  - Stop if the error fails to improve (has reached a minimum)
  - Stop if the rate of improvement drops below a certain level
  - Stop if the error reaches an acceptable level
  - Stop when a certain number of epochs have passed

Questions

1. What is linearly Separable problem? Give example.
2. Which type of problems can not be solved by Single layer perceptron ? Give example.
3. What is Activation function?
4. Why non-linear activation functions are used in ANN?
5. What is Biased input?
6. What is Learning Rate and what is the typical value(s) of it ?
7. In order to solve a complex problem, what is required to change/add in a neural network?
8. What is Feed Forward Neural Network?
9. What is Back Propagation Learning algorithm?
10. State at-least 3 applications of ANN.