

**Fundamentals of Artificial Intelligence**  
**Prof. Shyamanta M. Hazarika**  
**Department of Mechanical Engineering**  
**Indian Institute of Technology-Guwahati**

**Lecture - 33**  
**Learning in Neural Networks**

Welcome to Fundamentals of Artificial Intelligence. We continue our discussion on machine learning. So far we have looked at supervised learning and unsupervised learning. And in the last lecture, we had briefly introduced the essence of reinforcement learning.

All of these techniques of machine learning that we have looked at so far whether learning from observations and getting to the decision trees or for that matter linear regression that we have discussed in one of the lectures and the methods of unsupervised learning can broadly be put under what is referred to as symbol manipulation.

The very idea of using logic or formal systems for reasoning as well as representation is based on the concept that some form of symbol manipulation can lead to problem solving or what we can refer to as intelligent behavior. In today's lecture and the final lecture remaining in this week, we will look at a completely different approach to artificial intelligence which is called connectionist AI.

For here, we are more concerned with mimicking the way the human brain functions rather than getting to a set of symbols or a formal structure and manipulating it to arrive at new knowledge. So this lecture on learning in neural networks, we would try to see how the very complex network of neurons in our brain can be copied into an electrical system, can be copied and somehow be applied for learning and making decisions.

These networks for the very fact of being motivated by the neural structures is referred to as neural networks.

**(Refer Slide Time: 04:25)**

## Neural Networks



- Complex networks of simple computing elements
  - The simple arithmetic computing elements correspond to neurons – the cells that perform information processing in the brain.
  - The network as a whole corresponds to a collection of interconnected neurons.
- Capable of learning from examples
  - Through appropriate learning methods.
- Connectionist approach to computation
  - Exhibit complex global behavior, determined by the connections between the processing elements and element parameters.

So neural networks are in fact, some complex network of simple computing elements. In fact, in a more strict sense, we need to call them artificial neural networks. They are simple arithmetic computing elements, which corresponds to neurons, the cells that perform information processing in the brain. So the network as a whole corresponds to a collection of interconnected neurons.

And they are capable of learning from examples of course, through appropriate learning methods and as I was referring to this whole idea of computation is referred to as the connectionist approach. So they exhibit complex global behavior determined by the connections between the processing elements and the element parameters. In this lecture today, we will look at the very basic of such simple computing element and try to understand how they form the basis of computation.

But before we proceed, it would be pertinent to actually look at the structure of the human brain and try to understand at least how the brain works.

**(Refer Slide Time: 06:03)**

# How the Brain Works?



The exact way in which the brain enables thought is one of the great mysteries of science



The neuron is the fundamental functional unit of all nervous system tissue, including the brain.

## □ Brain

- Set of interconnected modules
- Performs information processing operations at various levels
  - sensory input analysis
  - memory storage and retrieval
  - reasoning
  - feelings
  - consciousness

## □ Neurons

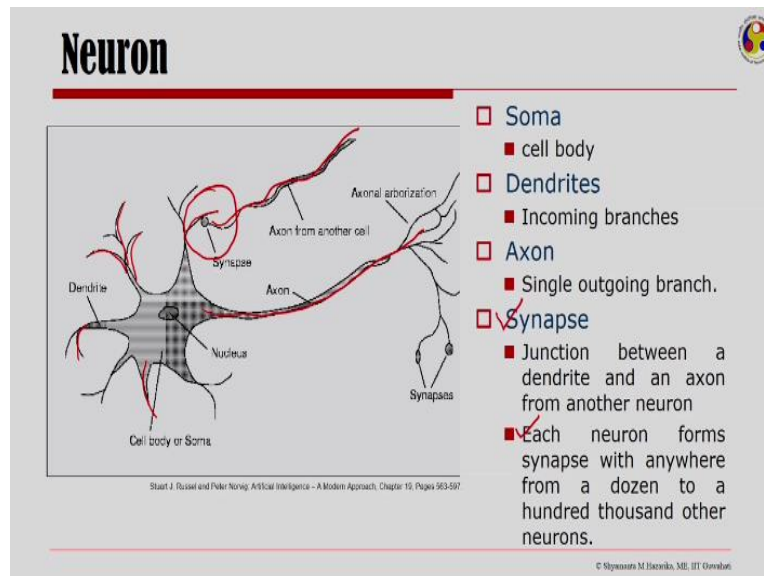
- basic computational elements
- heavily interconnected with other neurons

© Shyamanta M. Haerack, MB, III, Cleveland

The exact way in which the brain enables thought is one of the greatest mysteries of science. The brain, which is a set of interconnected modules, performs information processing operation at various levels. It does processing of sensory input that we receive. It definitely does memory storage and retrieval. And then, all our rational behavior that we exhibit is the result of reasoning that is there in the brain.

The very essence of feelings and consciousness lies in the brain. The neuron is actually the fundamental functional unit of all nervous system tissues, including the brain. And therefore, all these information processing operations at various levels that we have listed here must somehow be happening at the neural level. So neurons, the basic computational elements of the brain remain heavily interconnected with each other.

**(Refer Slide Time: 07:29)**



Here is what the neuron looks like. There is the cell body or soma. And then, we have a single outgoing branch, which is referred to as the axon and a number of incoming branches, which are referred to as the dendrites. Now a very important junction in this neural structure is the junction between the dendrites of one neuron and the axon from some other neuron. So this axon here is from a different neuron.

And this junction that we have between the dendrites of one neuron and the axon of another neuron is referred to as the synapse. So each neuron form synapse with anywhere from a dozen to a hundred thousand other neurons and this is something that makes it so very powerful as we will see in a minute. I would emphasize the very fact that the very important connection between the dendrite and an axon from another neuron that we got to remember is the synapse.

**(Refer Slide Time: 09:08)**

## Basis for Learning in the Brain



### □ Action potential

- Signals are propagated from neuron to neuron by a complicated electrochemical reaction. Transmitter substances are released from the synapses and enter the dendrite, raising or lowering the electrical potential of the cell body.
- When the potential reaches a threshold, an electrical pulse or action potential is sent down the axon.

### □ Excitatory and Inhibitory

- The pulse spreads out along the branches of the axon, eventually reaching synapses and releasing transmitters into the bodies of other cells.
- Synapses that increase the potential are called excitatory, and those that decrease the potential are called inhibitory.

### □ Plasticity

- Synaptic connections exhibit plasticity — long-term changes in the strength of connections in response to the pattern of stimulation.

Now the basis for learning in the brain lies in what is called the action potential. So signals are propagated from neuron to neuron by a very complicated electrochemical reaction. And some transmitter substances are released from the synapse and enter the dendrite raising or lowering the electrical potential of the cell body. So what happens is some form of activation comes and enters the neuron via the dendrites raising or lowering the electrical potential.

And when the potential reaches a threshold, an electrical pulse or what is called an action potential is sent down the axon. That is the first. And then the second important thing is how the pulse spreads out along the branches of the axon. Now these pulse that arrive at the dendrites, if the potential reaches a threshold, it is sent down the axon and the pulse spreads out along the branches of the axon and eventually reaching synapses and releasing transmitters into the bodies of other cells.

That is they now activate the next set of dendrites that is connected to them. Synapses that increase the potential are called excitatory and those that decrease the potential are called inhibitory. One very important fact that makes such neural structures interesting is that the synaptic connections that we have exhibit what is called plasticity. That is long term changes in the strength of connections in response to pattern of stimulation remains there.

**(Refer Slide Time: 11:37)**

# Computer vs. Brain



A crude comparison of the raw computational resources available to computers (circa 1994) and brains.

	Computer	Human Brain
Computational units	1 CPU, $10^5$ gates	$10^{11}$ neurons
Storage units	$10^9$ bits RAM, $10^{10}$ bits disk	$10^{11}$ neurons, $10^{14}$ synapses
Cycle time	$10^{-8}$ sec	$10^{-3}$ sec
Bandwidth	$10^9$ bits/sec	$10^{14}$ bits/sec
Neuron updates/sec	$10^5$	$10^{14}$

Stuart J. Russell and Peter Norvig Artificial Intelligence - A Modern Approach, Chapter 19, Pages 563-587

Even though a computer is a million times faster in raw switching speed, the brain ends up being a billion times faster at what it does!

✓ Computer chips can execute an instruction in tens of nanoseconds, whereas neurons require ✓ milliseconds to fire. ✓ Brains more than make up for this, however, because all the neurons and synapses are active simultaneously.

So if you compare the computer versus the brain, a crude comparison of raw computational resources available to computers and brains would make it clear that computer chips execute instructions in tens of nanoseconds, whereas neuron requires milliseconds to fire.

However, as I was trying to emphasize, when I was talking about the synaptic connections between neurons, one important thing that makes brain more than make up for its slow speed is that all neurons and synapses act simultaneously and this is what makes the brain so very powerful than any computational resources that we can think of.

Even though a computer is a million times faster, in raw switching speed, the brain ends up being a million times faster at what it does. And at the very basis of this lies the neural network of synapses that constitutes the human brain.

**(Refer Slide Time: 13:12)**

# Artificial Neural Networks



- Many neuron-like processing elements
  - Input & output units receive and broadcast signals to the environment, respectively
  - Internal units called hidden units since they are not in contact with external environment
  - Units connected by weighted links (synapses)
- A parallel computation system because
  - Signals travel independently on weighted channels and units can update their state in parallel.
  - However, most ANNs can be simulated in serial computers

Now artificial neural networks are where we put many neuron-like processing elements and try to create this structure. We have input and output units that receive and broadcast signals to the environment respectively. Internal units called hidden units are there which are in contact with only units that are hidden and are not in contact with external environment.

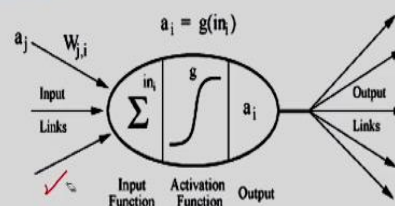
And all of these units are connected by weighted links to recreate the very essence of the synaptic connections that we have seen in the human brain. A parallel computation system is what we end up in, because signals travel independently on weighted channels and units can update their states in parallel. But what makes it exciting is that most artificial neural nets can be simulated in serial computers.

**(Refer Slide Time: 14:19)**

## Single Computing Element



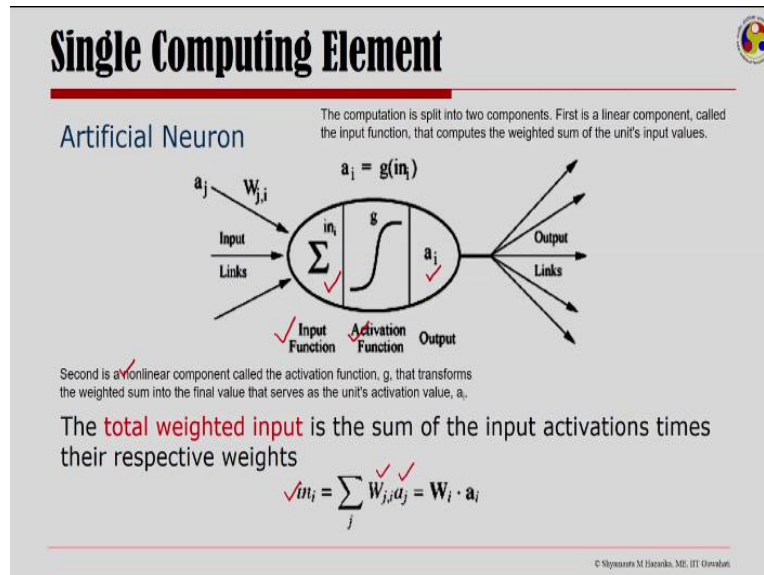
Artificial Neuron



- Receives n-inputs
- Multiplies each input by its weight
- Applies activation function to the sum of results
- Outputs result

So here is the single computing unit, which is referred to as an artificial neuron. So we have links coming to it as if these are the dendrites. And then going out as if that is the axon and then we have here an activation function that plays the role of the synaptic junction. So it receives n inputs. Now these inputs get multiplied by its weight. And then an activation function is applied to the sum of the results and then the output is produced.

**(Refer Slide Time: 15:06)**



The computation can be seen in two components. The first is a linear component, which is the input function that actually computes the weighted sum of the unit's input values. And then there is a second component, which is a nonlinear component. And it is called the activation function. The activation function transforms the weighted sum into the final value that serves as the units activation value AI.

Now the total weighted input is the sum of the input activation times their respective weights. So the input is the sum of the input activations times the weights.

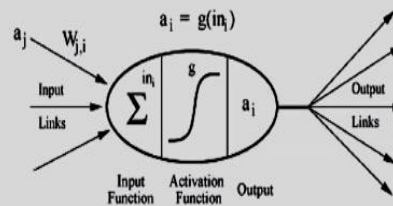
**(Refer Slide Time: 16:07)**



# Single Computing Element



## Artificial Neuron



The **elementary computation step** in each unit computes the new activation value for the unit by applying the activation function,  $g$ , to the result of the input function:

$$a_i = g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$

© Shyamam M Hazrati, ME, IIT Guwahati

And the elementary computation step in each unit computes the new activation value by applying the activation function  $g$  and this is mapped to the output  $a_i$ .

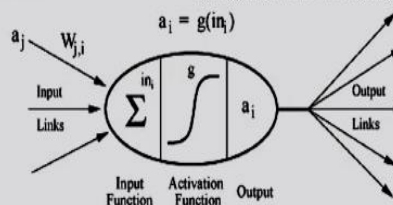
(Refer Slide Time: 16:23)

# Single Computing Element



## Artificial Neuron

✓ Allows for a simpler learning element because it need only worry about adjusting weights, rather than adjusting both weights and thresholds.



✓ Replace the **threshold with an extra input weight**. Add an extra input; activation fixed at -1. Weight serves the function of threshold at  $t$ .

$$a_i = \text{step}_t\left(\sum_{j=1}^n W_{j,i} a_j\right) = \text{step}_0\left(\sum_{j=0}^n W_{j,i} a_j\right) \text{ where } W_{0,i} = t \text{ and } a_0 = -1$$

© Shyamam M Hazrati, ME, IIT Guwahati

These artificial neuron allows for a simple learning element, because it need only worry about adjusting weights. For the threshold also can be included as an extra input step. So even if we said that there must be a minimum threshold in which the synaptic connection becomes either excitatory or if it is less than that, then it is inhibitory.

Here we replace the threshold with an extra input weight and add an extra input activation, which is fixed at -1. So the weight serves the function of the threshold at  $t$ . So basically, you have the threshold factored in here. And when I have this, I land up

with having even a more simpler learning element. Because now one need only worry about adjusting weights rather than adjusting weights and thresholds as well.

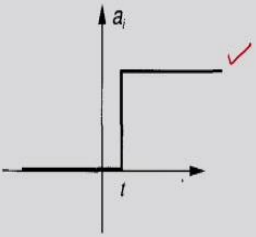
**(Refer Slide Time: 17:39)**

## Common Activation Functions

Different models are obtained by using different mathematical functions for  $g$ . Three common choices are the step, sign, and sigmoid functions.

Step function

The step function has a threshold  $t$  such that it outputs a 1 when the input is greater than its threshold, and outputs a 0 otherwise.



The biological motivation is that a 1 represents the firing of a pulse down the axon, and a 0 represents no firing. The threshold represents the minimum total weighted input necessary to cause the neuron to fire.

© Shyamam M Hazucha, M.E., IIT Guwahati

So let us take a moment and look at the common activation functions. Different models of artificial neural nets are obtained by using different mathematical functions for  $g$ , the activation function. Three common choices are step, sign, and the sigmoid function. The step function, which is shown here has a threshold  $t$ . And if the input is greater than this threshold, then the output is 1. Otherwise, the output is 0.

So that is the step function. Now the biological motivation of the step function is that a 1 represents the firing of a pulse down the axon and a 0 represents no firing. The threshold somehow represents the minimum total weighted input necessary to cause the neuron to fire. And that is what makes it either pass on the signal to the next group or stay inert.

**(Refer Slide Time: 19:02)**

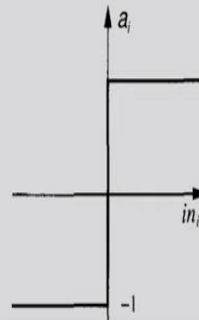
## Common Activation Functions



Different models are obtained by using different mathematical functions for  $g$ . Three common choices are the step, sign, and sigmoid functions.

### Sign function

Threshold function outputs  $\checkmark 1$  when input is positive and 0 otherwise



The sign function is when it outputs 1 when the input is positive, and 0 otherwise.

(Refer Slide Time: 19:11)

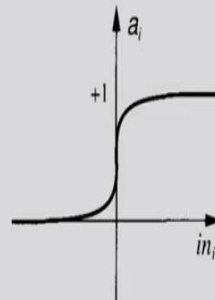
## Common Activation Functions



Different models are obtained by using different mathematical functions for  $g$ . Three common choices are the step, sign, and sigmoid functions.

### Sigmoid function

$$\text{Sigmoid}(x) = \frac{1}{(1 + e^{-x})}$$



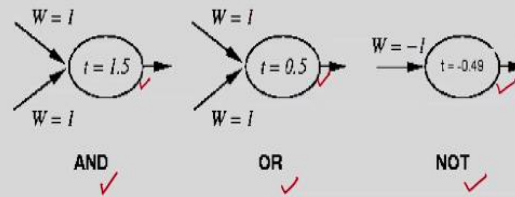
The sigmoid function is what is shown here.

(Refer Slide Time: 19:16)

# Basic Boolean Functions



One of the original motivations for the design of individual units (McCulloch and Pitts, 1943) was their ability to represent basic Boolean functions



- Simple neurons with can act as logic gates
  - Appropriate choice of activation function, threshold, and weights

© Shyamanta M. Dey, MS, III, Ghent

Now one of the original motivations for the design of such individual units, which is referred to as the McCulloch and Pitts model was their ability to represent basic Boolean functions. Simple neurons can act as logic gates. Some appropriate choice of activation function, thresholds and weights can make it realize that we can create different basic Boolean functions using the very simple computational element that we have shown.

Here, we have shown three Boolean functions, the and, or, and the not. A little introspection and looking at how you add up these  $W$ 's and see the threshold whether it crosses that or not. And you will have an output here or not, will make you realize that given these different threshold values and the different weights as shown, we can have these three basic Boolean functions. Now these are units with a step function for the activation.

**(Refer Slide Time: 20:41)**

# Network Structures



There are a variety of kinds of network structure, each of which results in very different computational properties

- In principle, **networks can be arbitrarily connected**
  - occasionally done to represent specific structures
    - Semantic networks
    - Logical sentences
  - Makes learning rather difficult
- **Layered structures**
  - Networks are arranged into layers
  - Interconnections mostly between two layers
  - Some networks may have feedback connections

Let us now quickly look at what are the different network structures that can be possible with these simple computational units. There are a variety of kinds of network structures each of which results in very different computational properties. In principle, networks can be arbitrarily connected. Now occasionally, networks are connected in a way to represent specific structures.

Like we may be interested in recreating certain semantic networks or certain logical sentences. But however, this makes learning rather difficult. Most times the network structures are layered structures. So the networks are arranged into layers and interconnections are mostly between two layers. Some networks may have feedback connections as well.

**(Refer Slide Time: 21:50)**

# Network Structures

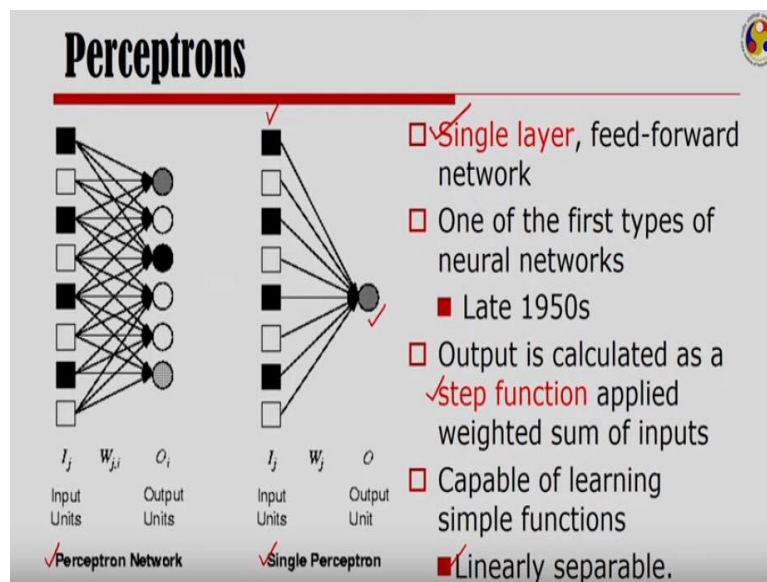


- **Feed-forward**
  - In a feed-forward network, links are unidirectional, and there are no cycles.
  - A feed-forward network is a directed acyclic graph.
  - **Layered feed-forward network** - each unit is linked only to units in the next layer; there are no links between units in the same layer, no links backward to a previous layer, and no links that skip a layer.
- **Recurrent Network**
  - In a recurrent network, the links can form arbitrary topologies.

So we have what are called the feed-forward networks. In a feed-forward network, the links are unidirectional and there are no cycles. So in a way the feed-forward network is a directed acyclic graph. The layered feed forward network is one in which each unit is linked to only to units in the next layer. And there are no links between units in the same layer.

Neither do we have links backward to a previous layer and no links that skip a layer. Recurrent networks on the other hand, are those in which the links can form arbitrary topologies.

**(Refer Slide Time: 22:40)**



So here, we show one of the most simplest of the feed-forward network which is a single layer feed-forward network. This is one of the first types of neural networks to be developed in the late 1950s. This single layered feed-forward network is called a perceptron network and a single perceptron is one where we have a number of input units all coming up to one output.

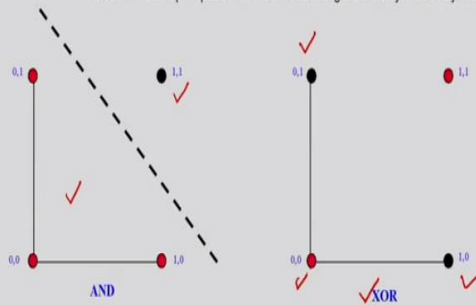
So output is calculated as a step function applied to the weighted sum of the inputs. The single layer feed-forward network referred to as the perceptron network is capable of learning simple functions and only linearly separable functions. Now recall our discussion on linear separability.

**(Refer Slide Time: 23:53)**

# Perceptrons and Linear Separability



The fact that a perceptron can only represent linearly separable functions follows directly from the function computed by a perceptron. A perceptron outputs a 1 only if  $W \cdot I > 0$ . This means that the entire input space is divided in two along a boundary defined by  $W \cdot I = 0$



Perceptrons can deal with linearly separable functions; some simple functions are *not* linearly separable XOR function

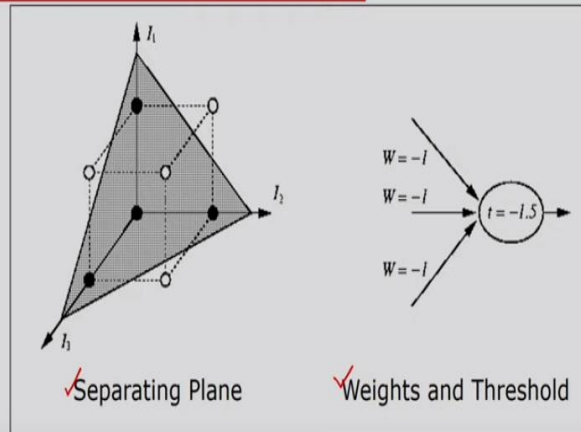
Linear separability is one where you could separate items using a decision boundary like this. So the fact that a perceptron can only represent linearly separable function actually follows from the function computed by a perceptron. The perceptron is a step function, the perceptron outputs a 1 only if I have the weight and the input greater than zero.

This means that the entire input space is divided into two along a boundary, which is defined as weight into input equal to zero. So perceptrons can deal with linearly separable functions. Some functions which are not linearly separable, such as the XOR function is something on which you cannot use a perceptron. Here, we have the and function. So 0 and 1 is 0, 0 and 0 is 0, 1 and 0 is 0, and 1 and 1 is 1.

So these two sides are linearly separable. Whereas if you do an XOR, you will see that here you have a zero, you have a 1 here, and you have a 1 here. So it is not possible for a linear decision boundary to separate them out.

**(Refer Slide Time: 25:42)**

## Perceptrons and Linear Separability



✓ Separating Plane

✓ Weights and Threshold

Linear separability can be extended to more than two dimensions; more difficult to visualize

Linear separability actually can be extended to more than two dimensions. But then it is more difficult to visualize. Nevertheless, we could have separating planes and we could create weights and thresholds in such a way that we have linear separability beyond what I have shown in the two dimensional space.

(Refer Slide Time: 26:09)

## Perceptrons and Learning



- Perceptrons can learn from examples through a simple learning rule
  - Calculate the error of a unit  $Err_i$  as the difference between the correct output  $T_i$  and the calculated output  $O_i$ 

$$Err_i = T_i - O_i$$
  - Adjust the weight  $W_j$  of the input  $I_j$  such that the error decreases
 
$$W_{ij} := W_{ij} + \alpha * I_{ij} * Err_{ij}$$
    - $\alpha$  is the learning rate
    - This is a gradient descent search through the weight space
- Great enthusiasm in the late 50s and early 60s;
  - Minsky & Papert in 1969 analyzed the class of representable functions and found the linear separability problem

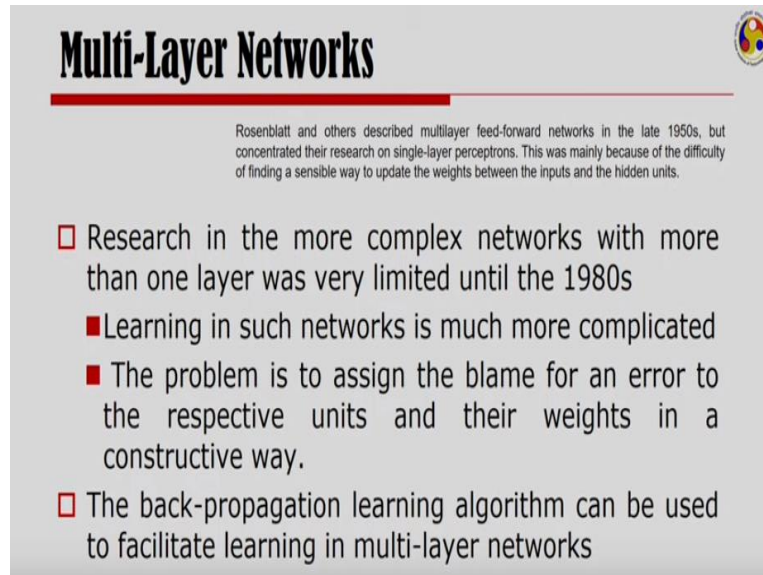
© Sreyasram M Haseenka, M.E., IIT Guwahati

Perceptrons can learn from examples through a very simple learning rule. It calculates the error of an unit as the difference between the correct output and the calculated output that is the error of a particular unit. And then all it needs is to adjust the width such that the error decreases and this is nothing but a gradient descent search through the weight space.



So there was great enthusiasm in late 50s and early 60s with the very introduction of the perceptron. However, Minsky and Papert in 1969 analyzed the class of representable functions and found the linear separability problem. And then people looked beyond perceptrons.

**(Refer Slide Time: 27:08)**



**Multi-Layer Networks**

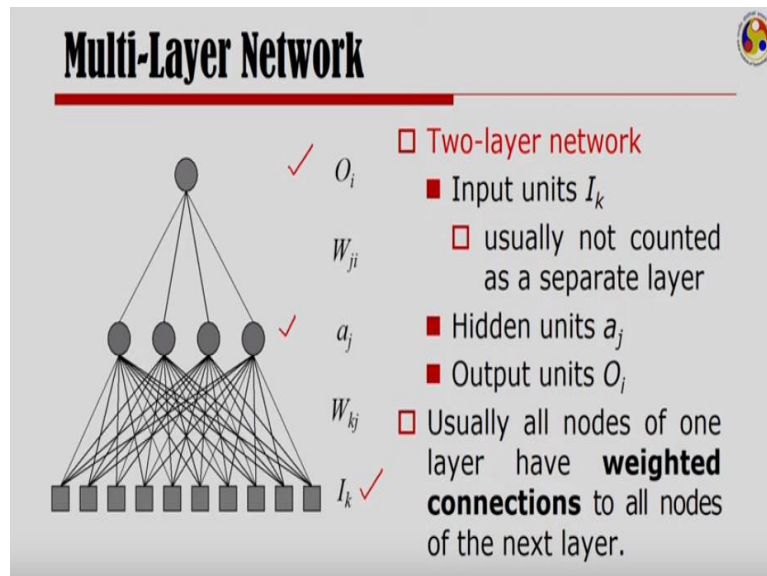
Rosenblatt and others described multilayer feed-forward networks in the late 1950s, but concentrated their research on single-layer perceptrons. This was mainly because of the difficulty of finding a sensible way to update the weights between the inputs and the hidden units.

- Research in the more complex networks with more than one layer was very limited until the 1980s
  - Learning in such networks is much more complicated
  - The problem is to assign the blame for an error to the respective units and their weights in a constructive way.
- The back-propagation learning algorithm can be used to facilitate learning in multi-layer networks

Rosenblatt and others, actually described the multilayered feed-forward networks in the late 1950s, but concentrated their research on single layered perceptron. This was mainly because of the difficulty of finding a sensible way to update the weights between the inputs and the hidden units. Research in more complex networks with more than one layer was very limited until the 1980s.

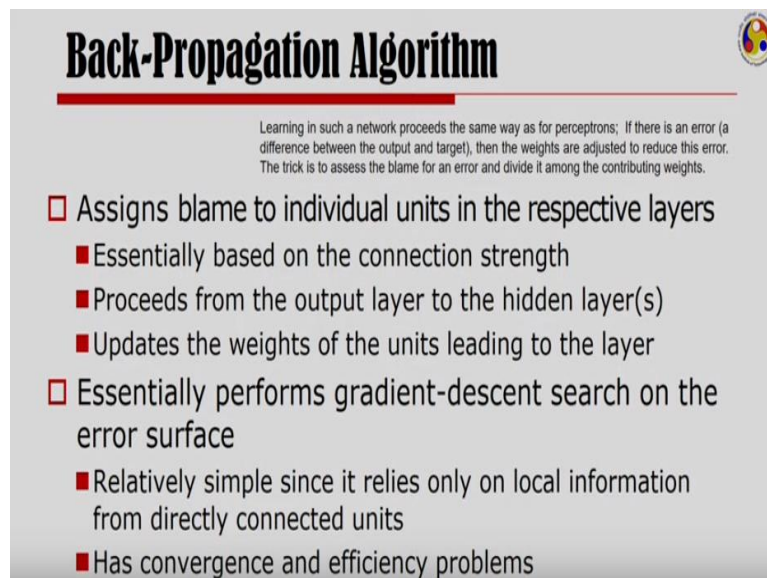
Because learning in such network is much more complicated. And the problem is to assign here the blame for an error to the respective units and their weights in a constructive way. The back propagation learning algorithm was one that facilitated learning in multilayer networks.

**(Refer Slide Time: 28:09)**



So here I have a two-layer network, which is of the input units here, the input units here and there are the hidden units and finally the output unit. Now the input unit is not counted as a separate layer. And therefore, this is referred to as a two-layered network. Usually, all nodes of one layer have weighted connections to all nodes of the next layer as shown here.

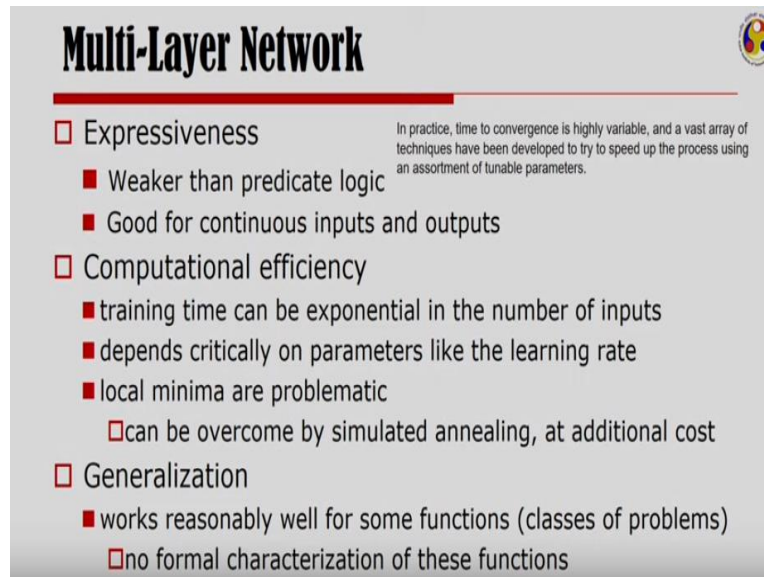
(Refer Slide Time: 28:49)



The back propagation algorithm, which allows learning in such networks proceeds the same way as for perceptrons. If there is an error, that is a difference between the output and the target, then the weights are adjusted to reduce this error. Now the trick here is to somehow assess the blame for an error and divided among the contributing weights. So you assign blame to individual units in the respective layer.

This is essentially based on the connection strength. So you proceed from the output layer to the hidden layer and update the weights of the units leading to the layer. You essentially perform here what is called the gradient-descent search on the error surface. And this is relatively simple, since it relies only on local information from directly connected units. However, such learning has convergence and efficiency problems.

**(Refer Slide Time: 30:03)**



**Multi-Layer Network**

- Expressiveness
  - Weaker than predicate logic
  - Good for continuous inputs and outputs
- Computational efficiency
  - training time can be exponential in the number of inputs
  - depends critically on parameters like the learning rate
  - local minima are problematic
    - can be overcome by simulated annealing, at additional cost
- Generalization
  - works reasonably well for some functions (classes of problems)
  - no formal characterization of these functions

In practice, time to convergence is highly variable, and a vast array of techniques have been developed to try to speed up the process using an assortment of tunable parameters.

Now multi-layer networks beyond 1980s have come in a big way after the slump in research in neural networks between late 60s and early 80s. Let us try to understand how does these multi-layer network match up with other forms of representation and reasoning that we have covered in this course or that is available for looking at problems with an artificial intelligence.

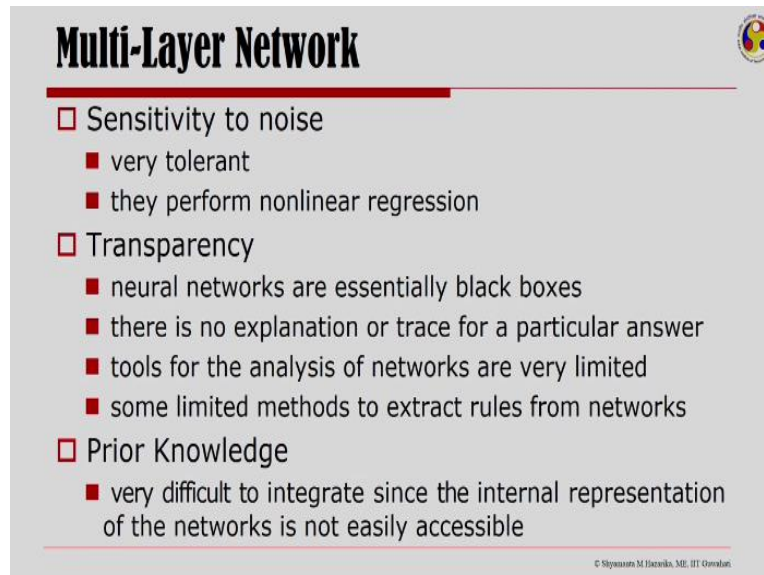
So multi-layer networks in terms of its expressiveness are actually weaker than predicate logic. They are good for continuous inputs and outputs. In practice, they have actually surpassed a lot of other representation formalism or for that matter learning methods. However, if you look at the computational efficiency, the training time can be exponential in the number of inputs.

So depends critically on parameters like the learning rate and local minimas are known to be problematic. Now these can be overcome by certain methods like simulated annealing at additional cost. Recall our discussion on simulated annealing when we had introduced problem solving by search. In practice time to convergence

is highly variable and a vast area of techniques have been developed to try to speed up the process using an assortment of tunable parameters.

In terms of generalization, it is seen that multilayered networks works reasonably well for some functions or classes of problems. However, there are no formal characterization of these functions.

**(Refer Slide Time: 32:38)**



**Multi-Layer Network**

- Sensitivity to noise
  - very tolerant
  - they perform nonlinear regression
- Transparency
  - neural networks are essentially black boxes
  - there is no explanation or trace for a particular answer
  - tools for the analysis of networks are very limited
  - some limited methods to extract rules from networks
- Prior Knowledge
  - very difficult to integrate since the internal representation of the networks is not easily accessible

© Shyamprasad M. Harekrisna, M.E., III, Gwalior

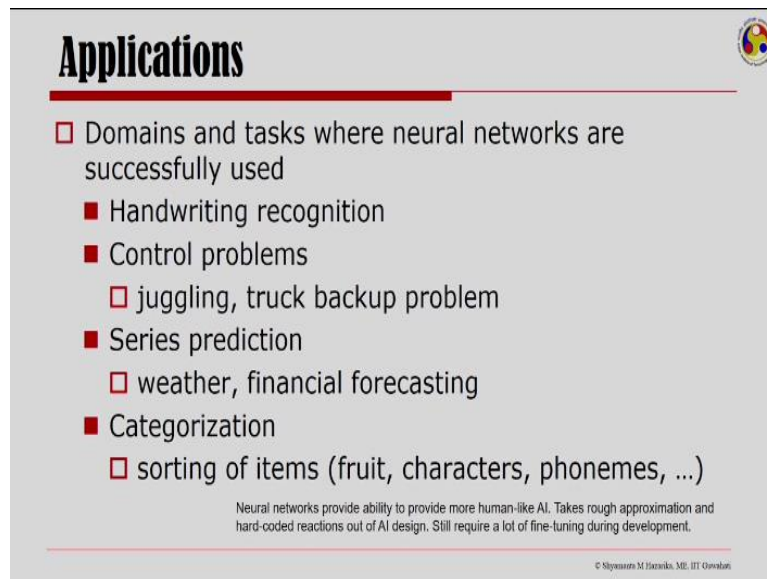
One very interesting thing that makes artificial neural nets popular is that they are very tolerant to noise. In terms of sensitivity to noise, they are perhaps one of the best and they perform nonlinear regression. In terms of transparency, that is knowing what is happening and how it is happening, possibly neural networks are essentially black boxes. There is no explanation or trace for a particular answer.

Tools for analysis of neural networks are very limited. Of course, we should point out here that some limited methods are there to extract rules from networks or to get to the decision tree for a given network. But then generally neural networks are known to be non transparent. In terms of integrating prior knowledge into the systems, it is very difficult to integrate prior knowledge into artificial neural networks since the internal representation of the network is not easily accessible.

In fact, if you look at the multilayered neural network, the network works by taking a group of inputs, weighing them, passing on to the next layer, use some weights to adjust themselves and then finally arriving at an output. So there is hardly any way

that we can think of integrating prior knowledge in such internal structures of the network.

**(Refer Slide Time: 34:42)**



**Applications**

- Domains and tasks where neural networks are successfully used
  - Handwriting recognition
  - Control problems
    - juggling, truck backup problem
  - Series prediction
    - weather, financial forecasting
  - Categorization
    - sorting of items (fruit, characters, phonemes, ...)

Neural networks provide ability to provide more human-like AI. Takes rough approximation and hard-coded reactions out of AI design. Still require a lot of fine-tuning during development.

© Seymour M. Harnik, M.E., IIT Guwahati

In terms of applications, domains and task where neural networks are successfully used are across many areas of our daily life. Handwriting recognition, lot of work has been done in terms of recognition of handwritten characters using artificial neural networks. A number of control problems such as the juggling problem, the track backup problem have been solved using artificial neural nets.

Perhaps one of the largest use of artificial neural networks have been in the area of weather and financial forecasting. One area that ANN has widely been used is in categorization or sorting of items. Now what we have covered here today is a very basic introduction to artificial neural networks.

Rather I would say we have tried to show you the simile of how the neural structure of the brain the very simplest computational element the neuron could have an artificial counterpart and then we could arrive at the perceptron and we have just made a mention of how multiple layered networks then can be created. And we could have learning using the back propagation algorithm.

Neural networks actually provide more human like AI. Takes rough approximation and hard coded reactions out of AI design. However, they require a lot of fine tuning during development. One needs to be very clear at this point that artificial neural

networks the very approach of computation using an ANN and the approach of AI that I have taken in most of the lectures until today, are distinctly different.

One could be referred to as symbolic AI. And the other is more to do with what is called connectionist AI. Thank you very much.