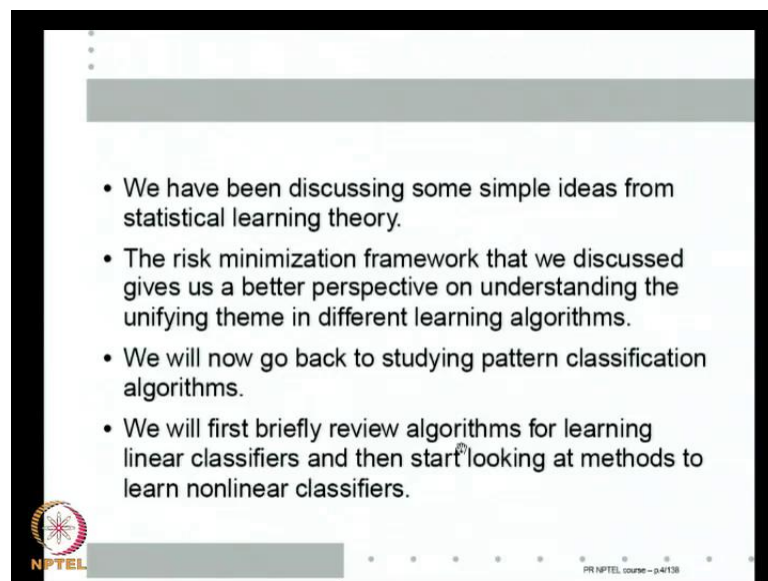


Pattern Recognition
Prof. P. S. Sastry
Department of Electronics and Communication Engineering
Indian Institute of Science, Bangalore

Lecture - 26
Overview of Artificial Neural Networks

Hello and welcome to the next lecture in this course in Pattern Recognition, for the last few classes we had been discussing ideas on statistical learning theory, and how one gives performance guarantees for learning algorithms or how one can bound the generalization error of a learning system. So, specifically we have seen you know the VC theory, so just to recap in two sentences, we looked at some ideas from statistical learning theory.

(Refer Slide Time: 00:48)



And the basic thing that we have introduced is the, so called risk minimization framework, whereby we essentially have a hypothesis space which represents some class of functions from the feature space to reals. And we evaluate each function or each hypothesis in terms of risk which is expectation of loss, so we seen that the risk minimization frame work gives a very good perspective on learning from examples, and it allows us to look at all the various learning algorithms under single unifying theme.

So, now that we have this perspective, now we go back and look at learning algorithms again, how to learn pattern classifiers some examples and also regression functions, and

examples, but this time we will be discussing things a little bit more in the risk minimization frame work. So, we have already discussed before we discussed statistical learning theory, we already studied some methods for learning linear classifiers and linear regression models. So, I will very quickly review because in some time I will very quickly review the algorithm for learning linear models, and then move on to learning non-linear models.

(Refer Slide Time: 02:12)

Linear Models

- In the two class case, the linear classifier is given by

$$h(X) = \text{sign}(W^T X + w_0)$$
- We have seen that we can also think of $h(X)$ as

$$h(X) = \text{sign}(W^T \Phi(X) + w_0),$$
 where $\Phi(X) = [\phi_1(X), \dots, \phi_m(X)]^T$ as long as ϕ_i are fixed (possibly non-linear) functions.

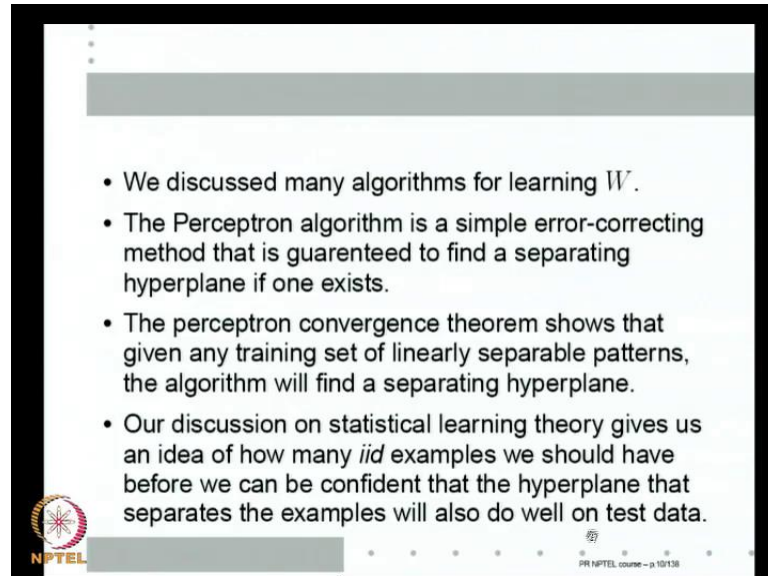
NPTEL © NPTEL course - p.0118

So, in the two class case a linear classifier as we now is given by sign of W transpose X plus W naught, so we have to learn this W and W naught to learn a linear classifier this is a two class case, we also see in the multiclass case, but let us not worry about it is just a review. Specifically, we also said this is important for us, that we can think of a linear classifier also as W transpose phi X plus W naught where phi X is some sort of m functions of X phi 1 X up to phi m X.

So, this also such a model can also be learnt using the standard linear model learning techniques as long as these phi's are fixed, so if the phi's are prefixed functions then this is still a linear model, because they are linear in the parameters to be learnt. Phi's can be non-linear functions of X, but does not matter as long as they are fixed the only things we are learning are the parameters, and the this W transpose phi X plus W naught it is still linear in it is parameters. So, linear models will still work for example, we seen how

we can think of simple linear least squares method for learning any degree polynomial function.

(Refer Slide Time: 03:24)



There are many algorithms we are learning the w , the first and the simplest we considered is the perceptron algorithm that is the simple error correcting type algorithm. And it guarantees to find a separating hyperplane if one exists we are still discussing two class case, actually we showed that is we proved a perceptron convergence theorem, that shows that given any training set of linearly separable patterns. Starting with any arbitrary initial W if you run the perceptron algorithm, we ultimately stop after some finite iterations and output is separating hyperplane.

Now, actually this might be a good example of asking what is it that the statistical learning theory gives you, what the perceptron convergence theorem says that give me any training set of linearly separable patterns, I will find a separating hyperplane, that separates the training examples. So, if I give only two patterns also it will find a separating hyperplane, if I give 20 patterns also it will find a separating hyperplane, if I give 200 patterns also it will find a separating hyperplane.

So, let us assume the class linearly separable, so all examples will also be linearly separable, but the separating hyperplane, it finds on 2 patterns may not be same as what it finds on 200 patterns. Obviously, if you give very few examples I may learn a


separating hyperplane that separates the examples, but may not separate the pattern classes.

So, the statistical learning theory is that we have studied tells us how many iid examples we should have, before we can be confident that the hyperplane that separates the examples, we will also do l n test and all. For example, in this particular case we know the family of hyperplanes in our d the d dimensional equivalent space as we see dimension d plus 1.

So, as a thumb rule we need at least 10 times the examples before a hyperplane that separates the examples is also very likely to separate the pattern classes, so that is the two (()) we first need an algorithm that does well on the examples. So, for example, the perceptron algorithm does well and as shown by the perceptron convergence theorem, on the examples it does well in the sense it separates the examples, then we have enough iid examples.

Enough depends on the d c dimension of the hypothesis space we are considering and in this case the hypothesis space is the set of hyperplanes and we know the d c dimension, we have seen how to calculate the d c dimension further. So, given the d c dimension the statistical theory tells us, that if I have sufficient number of examples then a hyperplane that does well on examples iid examples will also do well on the test data.

(Refer Slide Time: 06:02)



- We have also seen the least-squares method where we find W to minimize

$$J(W) = \frac{1}{n} \sum_i (W^T X_i - y_i)^2$$

where, for simplicity of notation, we have assumed augmented feature vectors.

- In our risk minimization framework, \mathcal{H} is parametrized by W , we take $h(X) = W^T X$ and minimize empirical risk under squared-error loss function.

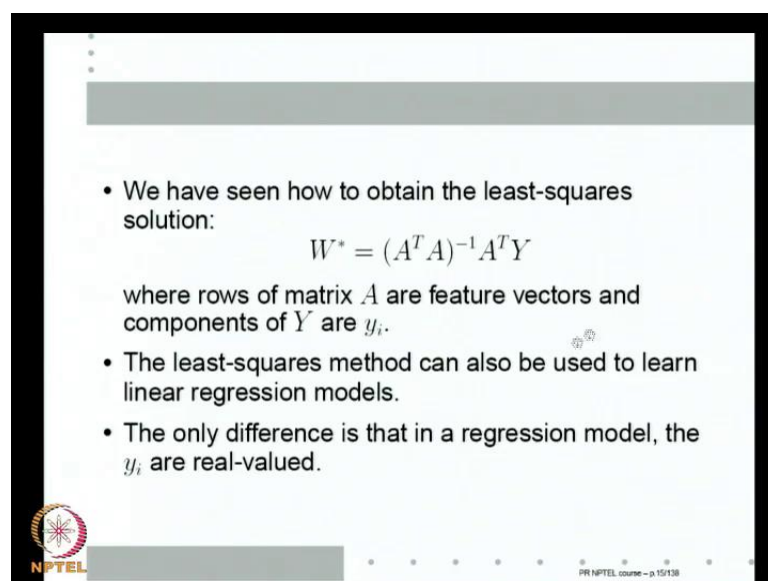
FR NPTEL course - p 12/138

The other main algorithm we considered for learning linear models is the least squares method, least squares method simply minimizes this function of W . Here, for simplicity of notation we have gone back to the augmented feature vector, I hope all of you remember that instead of writing $W^T X + W^T \mathbf{1}$, we can simply write as $W^T X$ by having $W^T \mathbf{1}$ as the first component of W and putting a first component of 1 in each X , thus called the augmented feature space representation.

Now, of course, in the least squares algorithm we considered earlier this 1 by n was not there, actually we put a half to cancel these two when we differentiate, but any constant here does not change the minimizer of W and minimizer of J . So, we can think that least squares method actually minimizes that why do we write it like this, this is very much in our risk minimization framework.

Now, our hypothesis space is a set of all function that is completely represents $W^T X$, so it is parameterized by the vector W and what we are minimizing, now this is nothing, but the empirical risk for the squared error loss function. So, this is the output of my function this is $h(X^T W)$, h_i this is y_i , so my loss function is $(X_i^T W - y_i)^2$, so I am using the squared error loss function. Then J is the empirical risk and a squared error loss function, so what least squares method is doing it is doing empirical risk minimization and squared error loss function that the hypothesis space of linear functions.

(Refer Slide Time: 07:42)





• We have seen how to obtain the least-squares solution:

$$W^* = (A^T A)^{-1} A^T Y$$

where rows of matrix A are feature vectors and components of Y are y_i .

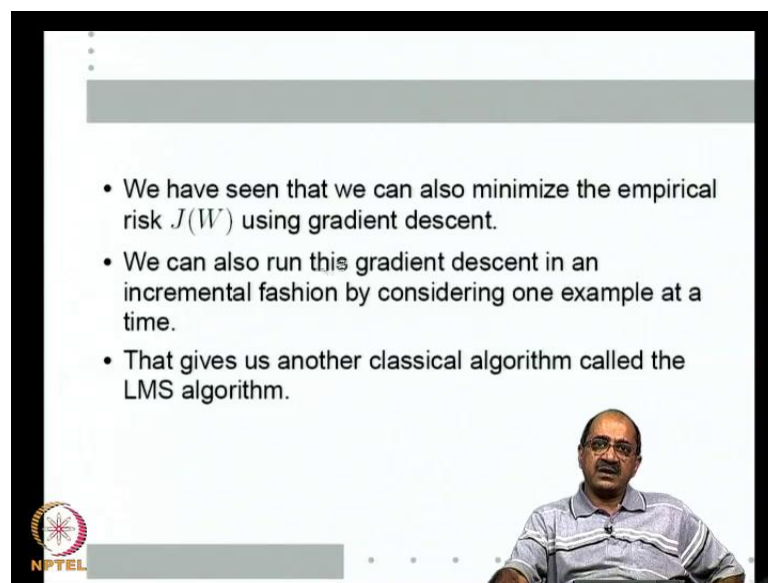
• The least-squares method can also be used to learn linear regression models.

• The only difference is that in a regression model, the y_i are real-valued.

 NPTEL  PR NPTEL course - 0 15138

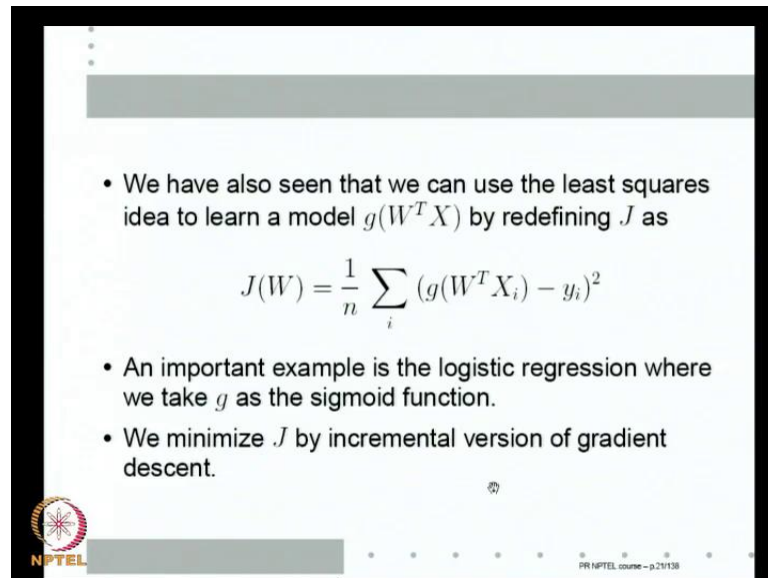
We have seen how to obtain the least square solution that turns out to be $A^T A^{-1} A^T Y$, where A is the matrix whose rows are the feature vectors and Y is the vector whose components are y_i . And the least squares method can as easily learn linear regression models, the only difference between the classification idea and the regression idea is that, when you are learning a regression function the y_i 's are real value. If you are doing the two class classification y_i 's are binary valued, if I learning a general function y_i are real value, except for that we still will be minimizing the same squared error and this is the least square solution.

(Refer Slide Time: 08:20)



We also seen that we can minimize the empirical risk which is same as $J W$ using gradient descent, and we also seen that we can run this gradient descent in incremental fashion, one example at a time rather considering the entire gradient. And that gives us one more classical algorithm what we call the L M S algorithm, and the added line just like perceptron that is also a very classical model up to linear element and that is nothing, but minimizing the empirical risk, in gradient descent using an incremental version we seen that also.

(Refer Slide Time: 08:58)



- We have also seen that we can use the least squares idea to learn a model $g(W^T X)$ by redefining J as

$$J(W) = \frac{1}{n} \sum_i (g(W^T X_i) - y_i)^2$$

- An important example is the logistic regression where we take g as the sigmoid function.
- We minimize J by incremental version of gradient descent.

And we also seen that least squares method can be learnt to can be extended to learn a model g of W transpose X , for in some function g rather than J 's W transpose X . What is the utility of this if I am learning any function of X here in as much as this empirical risk is a good approximation with expectation, if this is expectation of some g of X minus y whole square.

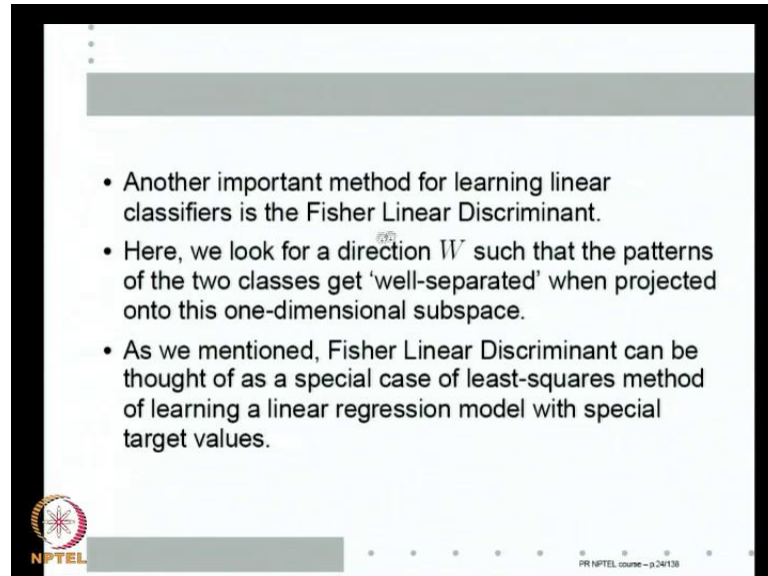
Then the best function is the conditional expectation, so if I mean a classification problem with y i's represented as 10, then we saw that this is the best function the conditional expectation is nothing, but the posterior probability. So, minimizing this it is trying to learn a learn a posterior probability of function learn a proper function of X , that gives me the posterior probability of class one for X and we are modelling the posterior probability by g of W transpose X .

Earlier if you just choosing W transpose X we are trying to model the posterior probability by linear function, which is not a very nice model, so sometimes if you choose a right g in g of W transpose X i is a nice model class of posterior probability and by linear least squares method. We can learn the proper posterior probability function and hence the proper classifier, as we seen an important example of this is the logistic regression, where g is taken to be the sigmoid function.

So, in that case of course, because of the g here we can no longer use the simple linear algebra method of getting a close form expression for w , but we can still minimize this

using gradient descent. So, generally when we use logistic regression, we minimize J by gradient descent or an incremental version of gradient descent.

(Refer Slide Time: 10:36)

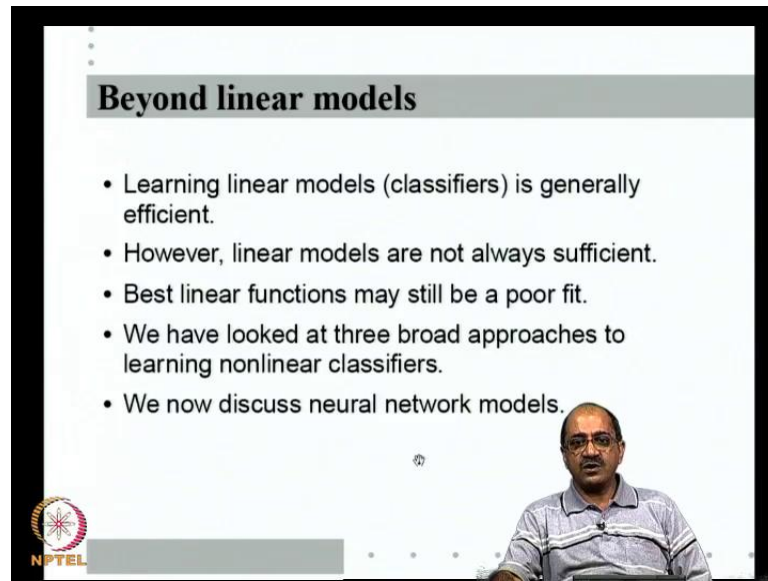


Another important method that we consider for learning linear classifiers is the fisher linear discriminant, where the idea is that we look for a direction W . Such that when the patterns of the two classes are projected onto this one dimensional space represented by W , then the two classes are well separated, well separate in the sense the distance between means is large relative to the individual variances.

Starting with such a objective function, we seen how to obtain the fisher linear discriminant and we also mentioned that such a linear discriminant, can be thought of as a special case of linear least squares method of learning a regression function with special target values. Normally, when I am learning a two class classifiers think of the target values are binary we shall think of them as binary.

I mean they are still true values, but if I take some special values then as we seen fisher linear discriminant can be thought of as a special case of linear least squares, so it is essentially a linear least squares method. So, these are all the set of methods that we have at our disposal for learning linear models, many of them are very efficient many of them are reasonably good and depending on the problem depending on what you want to learn we can use any of them.

(Refer Slide Time: 11:54)



Beyond linear models

- Learning linear models (classifiers) is generally efficient.
- However, linear models are not always sufficient.
- Best linear functions may still be a poor fit.
- We have looked at three broad approaches to learning nonlinear classifiers.
- We now discuss neural network models.

The slide includes an NPTEL logo in the bottom left corner and a video inset in the bottom right corner showing a man speaking.

But, the problem is all of them learn only linear models, because learning linear models is generally efficient these are all nice algorithms for example, linear least squares has a nice close form expression, it just involves one generalized inverse of a matrix. However, linear models may not be sufficient, if I have data X_i and y_i and want to model the function relation between y_i and X_i . y as some function of X a linear function may not be the best fit for my data, so the best linear function may still be a poor fit right then what do we do.

We discussed this right in the beginning of the course when I was giving you a general overview over the courses and there we have kind of intuitively identified at least three broad approaches to learning non-linear classifiers. What are they, the first one is say just like we have linear least squares on linear functions, linear functions are very simple I can always write them as $W^T x$.

So, we need a similar way of parameterizing a sufficiently good class of function, the problem with non-linear functions is non-linear function is any function that is not linear, linear functions have a very nice structure. So, linear functions we know how to parameterize, but non-linear functions are a very heterogeneous class, so if we can find some good way of parameterizing a sufficiently large class of non-linear functions.

Then once again possibly we can use a squared error loss function and minimize empirical risk over that family of functions to learn a good non-linear function. So, one

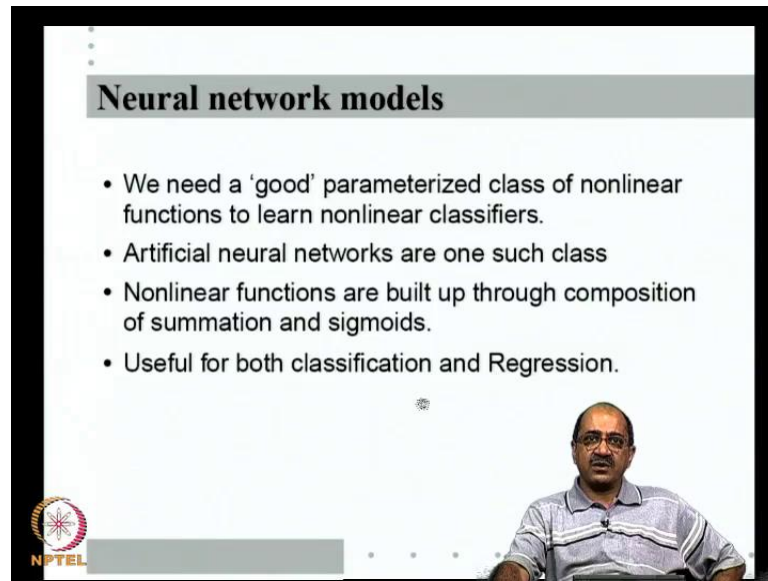
approach is to look for methods whereby I can get a good parameterized class of non-linear functions, and over that parameter over that class of functions ask are there some efficient ways of minimizing empirical risk let us say under squared error loss function.

The second approach is to say next best to linear is piece wise linear, so try and intuitively breakdown the feature space, not intuitively find some algorithm to breakdown the feature space. So, that within each part of the feature space I can learn a linear model, so I have to learn the partitioning of the feature space and the appropriate linear model together, they are not independent problems.

And this way of looking for piecewise linear models is another approach to learning non-linear models, best exemplified by this, so called decision tree classifiers, which we may or may not consider in this course. And the third approach is the kernel the kernel function based approach where the basic intuitive idea is that, I first non-linearly transform my feature space into some other high dimensional space, and there learn the learn a linear function.

As, we have seen what is non-linear in the original feature space can be linear in some other high dimensional space. So, if I can find an efficient way of transforming my original feature space to a high dimensional space and learn a linear function there, then I have learnt the non-linear function towards a feature space these are the three broad approaches. So, first we are going to consider the first approach, learn using a good parameterized class of non-linear functions, and that is called the neural network models, so that is what we are going to discuss next.

(Refer Slide Time: 15:30)



Neural network models

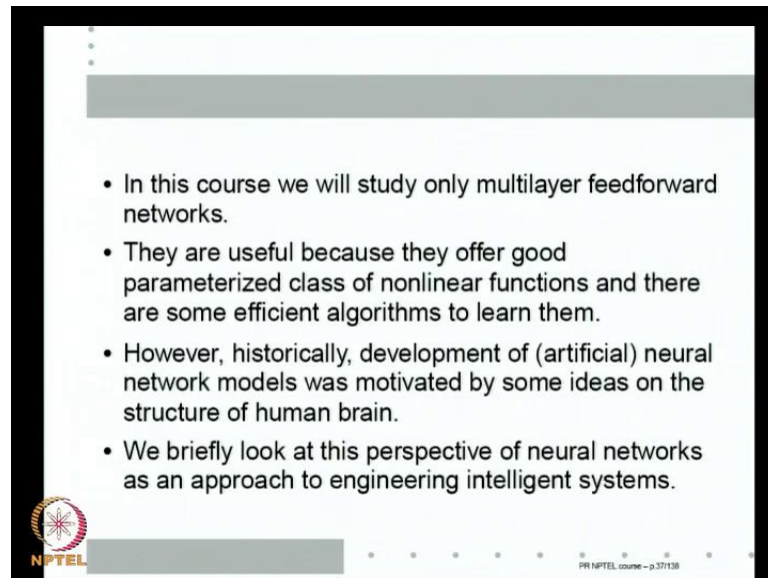
- We need a 'good' parameterized class of nonlinear functions to learn nonlinear classifiers.
- Artificial neural networks are one such class
- Nonlinear functions are built up through composition of summation and sigmoids.
- Useful for both classification and Regression.

The slide includes an NPTEL logo in the bottom left corner and a video inset in the bottom right corner showing a man speaking.

So, what the neural network models give us basically, as I said we need a good parameterized class of non-linear functions to learn non-linear classifiers. So, we are looking at what is a good parameterized class of non-linear functions and the, so called artificial neural network models are one such class. Here, non-linear functions are built up through function composition of the functions of finding linear sums and passing it through a non-linear function such as a sigmoid function, so I keep composing this operation of finding linear sums and passing through a non-linear function again and again.

And that is how I build up many non-linear functions we will see all the details, you this is this models are useful both for classification regression as a matter of fact that general models on non-linear functions. So, essentially what they do is learning a function given examples X_i, y_i how do y_i represent X_i as a function of X y as a function of X is why these models are for as we seen even though classifier is a binary valued function still a function so we can still use a general function learning method to learn classifiers also. It is in that sense actually this is a regression function learning model, which is also useful for classification.

(Refer Slide Time: 16:52)



- In this course we will study only multilayer feedforward networks.
- They are useful because they offer good parameterized class of nonlinear functions and there are some efficient algorithms to learn them.
- However, historically, development of (artificial) neural network models was motivated by some ideas on the structure of human brain.
- We briefly look at this perspective of neural networks as an approach to engineering intelligent systems.

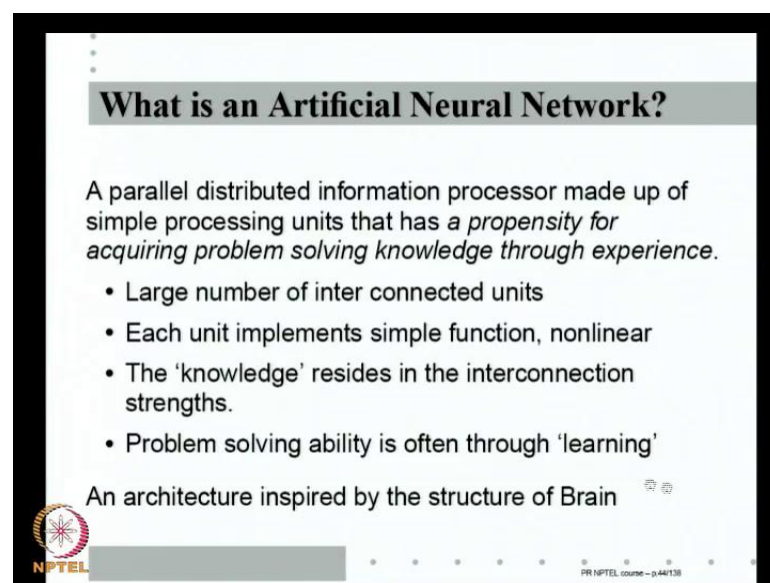
Artificial neural network models are pretty varied they are they are they are they are rich class of models, for this course we will study only what are known as multilayer feed forward networks as these provide the kind of parameterized class of non-linear functions that we are interested in. They offer a good parametric class on non-linear function and there are some efficient algorithms to learn them, so essentially they are useful for pattern recognition regression.

So, that is the only kind of neural networks that we consider, but having said that the neural network models themselves are developed historically, some other motivation the original motivation not just to find a good parameters class of non-linear functions. While, that is there actually the entire field neural network developed, through somewhat of a different stand point basically the development was motivated by some ideas on the structure of human brain.

That is the reason why these are called neural networks as you know what we have in our brain are called neuron, that is the basic computing element in the brain and these artificial neurons are somehow trying to model the structure of human brain at least that part of the structure of human brain which may be useful for computation. So, historically that developed as an approach towards engineering intelligent machines, while, that view point may not be strictly relevant to us.

Since, you are studying these models it is always good to look at that, so what I am going to do in this class is I will briefly present you this perspective of how neural network models have come up as an approach towards engineering intelligent systems. So, this is a kind of general introduction to the entire set of model that go under the under the rubric of artificial neural network models. As, I said we would not be studying all these models there is this sub class of models there, we will be only studying one particular kind of models which are useful for pattern recognition ultimately, but in the introduction we will look at the neural network models as an approach to engineering intelligent systems.

(Refer Slide Time: 19:34)



What is an Artificial Neural Network?

A parallel distributed information processor made up of simple processing units that has a *propensity for acquiring problem solving knowledge through experience*.

- Large number of inter connected units
- Each unit implements simple function, nonlinear
- The 'knowledge' resides in the interconnection strengths.
- Problem solving ability is often through 'learning'

An architecture inspired by the structure of Brain

NPTEL

© NPTEL course - 044138

So, from this point stand point we can define an artificial neural network as follows we will start with a general textbook based definition, we think of neural networks as a as a interesting computation paradigm. Like as opposed to the standard digital computer stored program digital computer method that we know of computing, which possibly whose generic abstract model is a turing machine.

As, opposed to that we think of a slightly different way of looking at computation, so one way you can define artificial neural networks is a parallel distributed information processor. It is a information processor say it is a kind of computing machine, which is both parallel and distributed. And it is made up of very simple processing units it is parallel in the sense there are many processing units, but each processing units not very

complicated it does not have for example, huge stored program capability it can only represent very simple input output function, but there are many such processing units.

So, it is a parallel processor made up of large number of simple processing units and as a whole it has propensity to acquire knowledge through experience, so basically that is what is useful for us because as we seen all of machine learning is about learning from examples, almost a machine learning is about learning from examples. So, a computing paradigm which puts learning as the constrone of all problem solving is important to us.

So, that is that is how the original artificial neural networks are thought of parallel distributed information processing processor, contains large number of simple units and has a has a propensity for acquiring problem solving knowledge through experience. So, what is it characterized by large number of interconnected units, each unit implements a very simple function may be a non-linear, but very simple it does not have much of stored program capability a simple input output relationship is all it has.

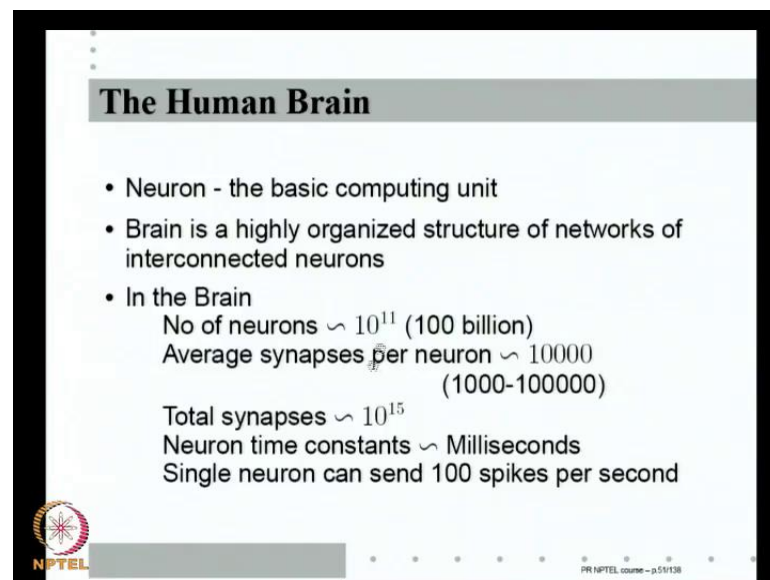
The units are interconnected as we shall learn them each of the interconnection has some weight or strength some parameter associate with the so they are called interconnection strengths. And much of the knowledge resides in the interconnection strengths by knowledge, we mean it is ability to solve a problem comes because of the interconnection strengths not because of the individual processors input output function.

Process input output function is very simple, but it is the interconnection strengths that can program or make the machine learn to solve a particular problem, so the problem solving ability is often through learning. So, it is quite distinct from the way normal digital computing machines are looked at, where the knowledge resides in the programs. So, to say by executing a precise sequence of operations and the particular precise sequence is where the knowledge resides.

That is how a a generic digital computing machines solves a problem, because here we have large number of units, they are interconnected each interconnection has a strength each unit itself is very simple, but this interconnection is what gives it the ability to solve a particular problem or represent a particular function. This is say somewhat of a different architecture, it does not look like the standard computing architecture the reason why such architecture were at all thought about is because they are inspired by the structure of the brain.

While, we may still may not know fully exactly how our brain gives us all the capabilities that we have, lot is known about the structure of the brain and it is quite distinct from the standard stored program computer architecture that we have. What is the structure of the brain?

(Refer Slide Time: 23:19)



The Human Brain

- Neuron - the basic computing unit
- Brain is a highly organized structure of networks of interconnected neurons
- In the Brain
 - No of neurons $\sim 10^{11}$ (100 billion)
 - Average synapses per neuron ~ 10000
(1000-100000)
 - Total synapses $\sim 10^{15}$
 - Neuron time constants \sim Milliseconds
 - Single neuron can send 100 spikes per second

NPTEL
PR NPTEL course - 051118

In the brain in the human brains neurons are the basic computing units, our brain is nothing, but a highly organized structure of interconnected networks of neurons. Now, what does each neuron do of course, there are many different kinds of neuron, but for our purposes we can ideally visualize a neuron like this. Neuron is a cell a biological cell, it is a cell body and it has input and output terminal, so to say, so there is one process thing sticking out of the neuron, which is it is output terminals, so to say it is called an axon.

And it has many inputs coming they are called dendrites, so on the dendrites axons of other neurons come and make connections, these connection points are called synapses. So, whenever a neuron fights let us say essentially when the when the the neuron at it is resting is at roughly minus 40 milli volts from the surrounding medium, and for some reason if the neurons potential increases above some threshold. Then the neuron fights what is called an axon potential, it is essentially of of a travelling wave of ionic current which is let out through it is section.

So, if the potential goes above some threshold briefly the neuron fights is called a spike, which is a kind of a ionic current that travels down the axon, when it travel down the

axon the axon then branches out or what is called arborizes, and makes connections with lot of input terminals sort of other neurons. That generates of other neurons is such a connection is called a synapse, at the synapse because of this axon fights ionic current part of that ionic current comes and injects some potential into the receiving neuron at the synapse.

Now, the synapses are on the dendrites and dendrites make a big tree structure on one side on the neuron, so different synapses can be at different distances from the cell body, the soma and synapses can be of different types. So, because of that the amount of charge injected at a synapse will have different affects on the surface potential of the cell body of the receiving neuron. Now, so each of these injected potentials contribute a a part to the surface potential of the receiving neuron cell body.

So, we can think that the incoming signal is multiplied by some connection weight of the synapse, and that is what gives us the the add the addition to the surface potential of the receiving neuron. There are many, many synapses all of them are summated to get the surface potential, if it goes enough then once again the neuron fights and this is how it goes on is an highly interconnected structure of neuron and all computing goes through like this.

So, each neuron is simply you know it receives lot of inputs through the synapses essentially at each synapse because of the inputs from other other neurons, some charges injected into the receiving neuron. And the net effect of all the charge injections at various synapses and it is input terminals, they are called dendrites is a increase in surface potential of the some of the receiving neuron. And if it goes above it is threshold then it fights an action potential on it is own, which now travel through it is axon and through the axon arborization to all the downstream neurons family and so on.

Now, this is the generic structure of how simple neurons function, now the the the the real complexity is in numbers, in the human brain there of the order of about 10 power eleven neuron. That is about hundred billion most of these estimates most of the estimates, so say give or take a 10 power roughly, on the average there are about 10 thousand synapses per neurons, at a low of thousand to a high of hundred thousand. So, of the order of about 10 thousand synapses per neuron that is the amount of find in a neuron has and similar amount of find over to a neuron has.

And so, total synapses taking 10 thousand per neuron and 10 power 11 neuron is about 10 power 15 synapses in the brain. A neuron time constants are in the milli seconds, so as a matter of fact once neuron fires an action potential it takes some time before it comes back to its resting potential till then it is not sensitive to any other input it receives. So if we think of this firing of action potential as the switching, then switching time constants are neurons are in milli second range, which means a single neuron can send up to about a hundred spikes per second, these are the numbers.

So, essentially the entire computing is now to be mediated by various neurons sending spikes, when they receive inputs and it is coordinated activity is what gives rise to all the intelligent behaviour we know, because it is a highly organized structure of networks in neurons. It is not just any arbitrary network, but not bothering about that just looking at the basic structural connectivity.

(Refer Slide Time: 29:01)

A rough estimate of processing power:

- One arithmetic operation per synapse
- 10^4 operations per neuron per spike
- 10^6 operations per neuron per sec
- 10^{17} operations per sec!!

(A gigaflop is 10^9 operations, teraflop is 10^{12} operations!)

Massive parallelism can deliver massive computing power, if we know how to manage it

NPTEL

We can do some simple rough estimates, let us say each synapse is to be viewed as analogous to doing one arithmetic operation. Then we can do 10 power 4 operations per neuron per spike, that many synapses because may not be that much, but let us just put an arbitrary estimate. We can send about hundred spikes per second, so we can do 10 power 6 operation per neuron per second, that is about 10 power 7 operations per second Indian type brain.

Obviously, things cannot work at this throughput pace, it does not because there is so many of few neurons are meant to keep you keep your metabolism going, keep your heart going, you know keep your body posture going and so on. But, and you know there are many neurons in your you know eye and retina and so on. So, there are lot of sensitive neurons lot of motor neurons, which are needed for the organism to function, but still this is staggering amount of operations possible.

If if we know if the if all the neurons are working in concept, so this is something that all of us know massive parallelism can deliver massive computing power. If only we know how to manage it of course, one does not know how to manage you know a 100 billion process to act together, but having said this then one can as kit seems to be a a a very simple structure. Each neuron has very simple input output behaviour and somehow the entire power seems to be the way in which they are interconnected and made to work in concept.

(Refer Slide Time: 30:47)

Digital computers:

- Precise design, highly constrained, not very adaptive or fault tolerant, Centralized control, deterministic, basic switching times $\sim 10^{-9}$ sec

Natural neural networks:

- massively parallel, highly adaptive and fault tolerant, self configuring, self repairing, noisy, stochastic, basic switching time $\sim 10^{-3}$ sec
- Most capabilities of Brain are LEARNT.

So, should not we try and think of computing system architectures, which somehow mimic this, so here is a dramatic way of saying this if we look at digital computers, they are very precised design of course, you use very sophisticated software to design the chip layout. They are highly constraint not very adaptive or fault tolerant if something goes wrong in the chip that is the end of it, mostly centralized control mostly deterministic and our basic switching times of the order of nanoseconds.

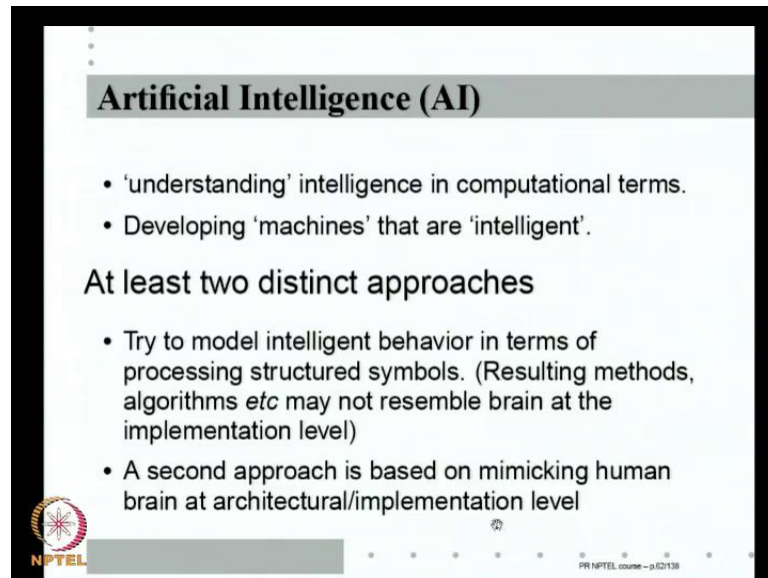
That is the natural neural networks are massively parallel, they are highly adaptive and fault tolerant neurons keep dying we do not have many bone marrows, where new neurons are made at the current knowledge, we think if at all just a very small fraction of neuron degeneration grows most of the neurons that we were born with only dying. But, in spite of it we have lot of adaptation and lot of fault tolerance of course, they are self configuring self repairing noisy stochastic, and have basic switching times of only 10 power minus 3 seconds the switching times are milli seconds.

And what is important is that it is not pre-programmed most of what a person of 5 or 10 year old can do, almost 99 percent of the capabilities of 5 or 10 year old person are not there at birth. When you are born with your neurons you cannot even do very simple things like sitting up or standing and walking, so most capabilities of our brain are all learnt. So, not only it has very strange architecture, but it essentially relies on learning, given that neurons input output structure is very simple and already well constrained.

It looks like the only thing that we can learn not synapses may be we form new synapses or we modify existing synapses. So, essentially synaptic strength might be what changes when we learn, so essentially is the interconnection strengths that translate into having learnt a particular capability. So, whatever knowledge resides in the brain must be that of interconnection strengths, this is actually what was first propounded as a theory or a hypothesis by a famous psychologist called Huber.

When he wrote a book in late 40's called organization of behaviour the basic premise is very simple because neuron functions seem to be very straightforward. There is not much scope there for doing something drastically different, but the behaviour of a 2 year old human being to a 5 year old human being is so different. That there must be something that is changed in the brain and most probably what is changed are all the synaptic strengths, so all learning is in terms of changing synaptic strengths.

(Refer Slide Time: 33:51)



Artificial Intelligence (AI)

- 'understanding' intelligence in computational terms.
- Developing 'machines' that are 'intelligent'.

At least two distinct approaches

- Try to model intelligent behavior in terms of processing structured symbols. (Resulting methods, algorithms *etc* may not resemble brain at the implementation level)
- A second approach is based on mimicking human brain at architectural/implementation level

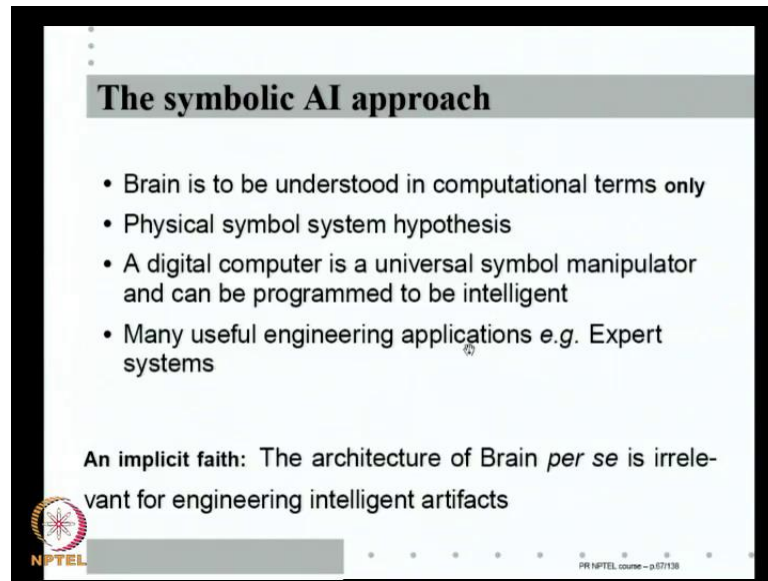
NPTEL PR NPTEL course - p.12/138

So, now stepping back a little bit if you think of the feel of artificial intelligence of course, artificial intelligence today means one particular set of techniques, but let us not worry about it. Let us think of it in the sense of what the name suggests, its grand goal is to understand intelligence in computation terms. And develop machines that are intelligent everything we put in codes, so that let us not get into any philosophical arguments.

Let us not define what a machine is and what intelligence is, but that is irrelevant, but within the context of what all of us understand, artificial intelligence support understanding an intelligent behaviour in computational terms and developing machines that are intelligent. Historically, today what we consider that there were 2 distinct approaches that people followed, one may be called the symbolic approach, where the idea is to develop models to for a intelligent in terms of processing symbols.

The idea is that the resulting methods or algorithms may not resemble, how brain does any thing as a matter of fact you may not even understand how brain does something, but if you understand what computation is involved in a particular intelligent behaviour. Then we can simply try and create it through some symbol manipulation, the second is to say well we do not know how the brain actually achieves its function, but we know lot about the brains architecture. So, lets us try and mimic the architecture of the brain, and then ask does it lead us to a kind of machine that has some interesting behaviour.

(Refer Slide Time: 35:31)



The symbolic AI approach

- Brain is to be understood in computational terms only
- Physical symbol system hypothesis
- A digital computer is a universal symbol manipulator and can be programmed to be intelligent
- Many useful engineering applications e.g. Expert systems

An implicit faith: The architecture of Brain *per se* is irrelevant for engineering intelligent artifacts

NPTEL

PR NPTEL course - p.07138

The first approach can be called the symbolic AI approach, the basic idea there is that the brain is to be understood in computational terms, so may be brain is to be understood only in computational terms that is if I look at any intelligent behaviour speech natural language understanding, common sense. All of it is some kind of a computation I may not fully understand what kind of computation it is, but nonetheless I can think of it as just a computation and any computation can be done on a turing machine.

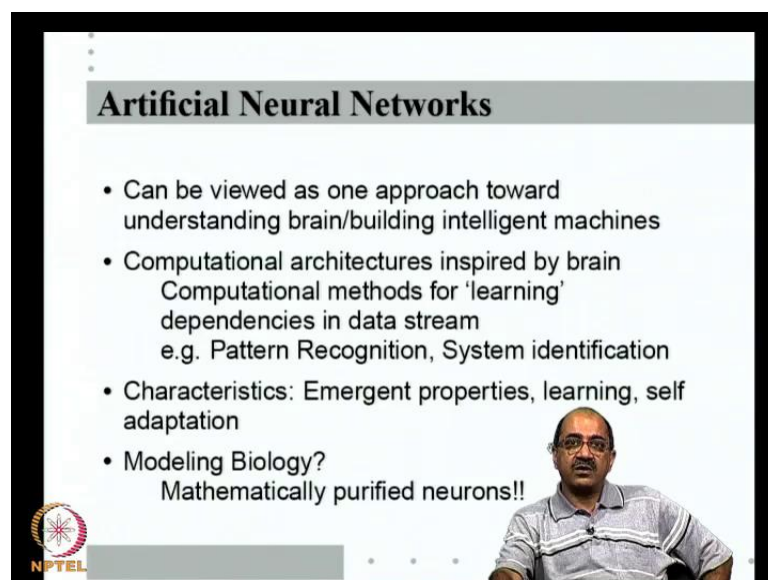
So, essentially any machine that is capable of carrying what abstract computation is capable of exhibiting intelligent behaviour, that is the basic idea of the symbolic approach. This is often referred into what is called a physical symbol system hypothesis, what the hypothesis says is that any any physically realisable system is capable of creating, storing, modifying, abstract, symbols is in principle capable of intelligent behaviour.

So, essentially intelligence says that you you symbolically represent the world outside, through some symbols inside and define proper operation of the symbols to represent. You know whatever inferences you you want to make or whatever changes in the outside world that you want to track and so on. So, any such system which is capable of creating, storing and modifying abstract symbols is capable of exhibiting the intelligent behaviour. Since, a digital computer is a universal symbol manipulator and it it can in principle be programmed to be intelligent.

That a digital computers architecture is vastly different from the brains architecture is really irrelevant, because brain is one way of representing one physical system that can represent abstract symbols may be. Digital computer is another kind of physical system that can represent a abstract symbols, so as long as the system has the capability to manipulate abstract symbols, it is capable of intelligent behaviour. So, we have to just figure out how to understand any intelligent behaviour in computational term, so that we can program it in a digital computer this is the basic symbolic A I approach.

Of course, it has given rise to lot of interesting application expert systems, chess playing programmes all of them come out of this symbolic A I approach, but something that characterizes the symbolic A I approach is the the the the the faith, the dogma, the theory, the hypothesis, whatever we want to call it. Which, says that the architecture of brain per se is irrelevant for engineering intelligent artifacts, that the specific architecture if the brain is some accident of the evolution. So, it is not necessary that such an architecture is needed for intelligent behaviour, and this is this is the faith and symbolically A I approach. Where, we are discussing it only to context of the neural networks, we are not saying right or wrong or which is better and all those those are unnecessary philosophical debates, but this is one approach.

(Refer Slide Time: 38:43)



Artificial Neural Networks

- Can be viewed as one approach toward understanding brain/building intelligent machines
- Computational architectures inspired by brain
Computational methods for 'learning' dependencies in data stream
e.g. Pattern Recognition, System identification
- Characteristics: Emergent properties, learning, self adaptation
- Modeling Biology?
Mathematically purified neurons!!

NPTEL

The slide features a small inset image of a man with glasses and a mustache, wearing a light-colored polo shirt, sitting in front of a white background. The NPTEL logo is visible in the bottom left corner of the slide.

The other approach which we are more interested in is the artificial neural network approach, which is another approach towards understanding brain or building intelligent

machines. An approach characterized by mimicking the brain at an architecture level and I am trying to see within that architecture, can we think of some simple learning algorithms, which will allow us to solve some problems.

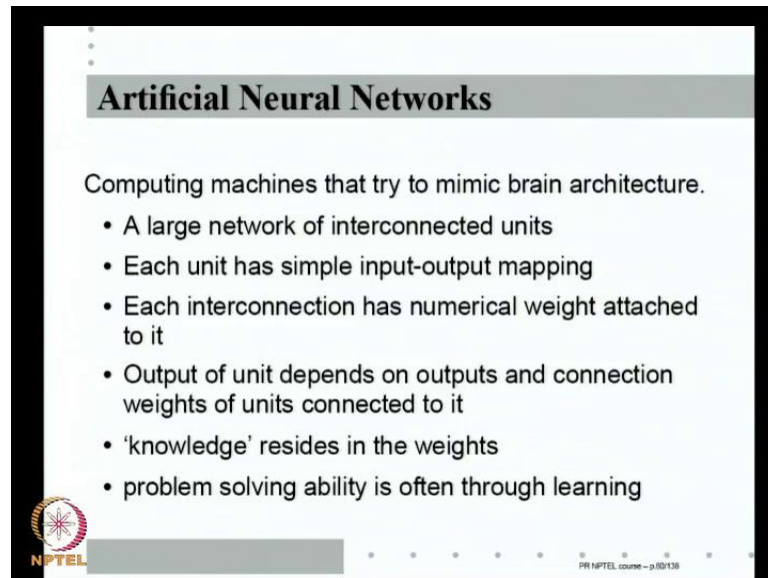
So, it represents a computational architecture inspired by the structure of human brain or structure of mammalian brain or structure of any intelligent organisms brain, so it gives rise to computational methods for learning dependencies in the data streams. There are many interesting they are just like expert systems on the symbolic A I set, there are many interesting application game on this system identification for pattern recognition so on.

There are some interesting methods of learning using the structures, the characteristics of the structures is that the emergent properties, what is there are some properties of the entire structure which cannot easily be assigned to any subunit. So, the the the network as a whole has some interesting properties, which it cannot be said that it is synapse that has the property or the neuron that has the property.

The other interesting aspect of this models is because they are built up very simple processing units, and every thing resides in the connection strengths, learning is the only way to program them. So, we have to somehow find the weights and the only way we find the weights are through learning, whether artificial neural networks really gave rise to understanding biology is a is a difficult question. I will come back to it in a in a in a few minutes again, when we finish the discussion, but most of the model is that we are considering here, can be thought of as mathematically purified neurons.

They are not sufficiently realistic in terms of real biological neurons, they can be thought of as very simple mathematical model. So, we can think of them as models or mathematically purified neurons and in that sense this the this specific models, we considered may not have much to do with modelling biology.

(Refer Slide Time: 41:04)



Artificial Neural Networks

Computing machines that try to mimic brain architecture.

- A large network of interconnected units
- Each unit has simple input-output mapping
- Each interconnection has numerical weight attached to it
- Output of unit depends on outputs and connection weights of units connected to it
- 'knowledge' resides in the weights
- problem solving ability is often through learning

NPTEL PR NPTEL course - p.03/138

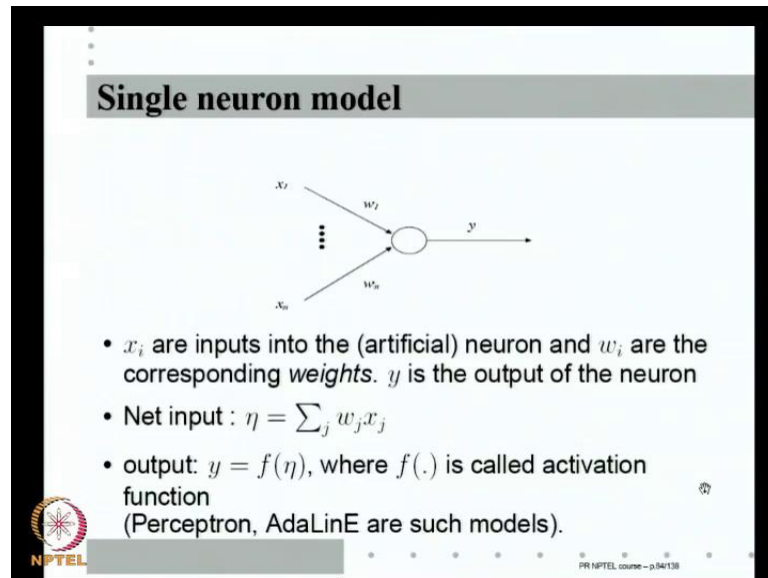
So, we can now sum up the discussion artificial neural networks are computing machine that try to mimic brain architecture, so what do they have large number of interconnected units. Each units like a neuron that is why they are called artificial neurons, sometimes people omit the word artificial, but at least in the beginning we view the word artificial. Each unit has simple input output mapping like your neuron firing an action potential, so essentially it takes some sum of some weighted sum of it is inputs and passes it through some function to get it is output.

So, each input is a very simple input output mapping, and they are interconnected and each interconnection has a numerical weight attached to it it is like the synaptic strength, in the in the real neural networks, and the output of a unit depends on the outputs and connection weights of units connected to it. Each unit receives inputs from many other units which are interconnected network of units and the output of any unit depends on both the outputs of the units, that send their outputs to you as well as the weights that connect those outputs to this unit.

The knowledge resides in the weights, essentially if you want the system to do anything the only thing you can change are the weights. The the the functional relationship of how output depends on the weights is prefixed, but you can change the values of the weights, so the knowledge essentially resides in the weights that is the kind of structure artificial neural networks give rise to.

And the problem solving ability hence is often through learning, so because of large number of weights and it is impossible to analytically decide how the weight should be or impossible to get close form expressions for how weight should be much like in what we are considering, so far in pattern recognition, the problem solving ability often comes through learning.

(Refer Slide Time: 43:06)



These models we have already seen actually, we can think of a single neuron model like this, there are many inputs say X_1 to X_n . Each input is connected with a neuron through some synapse, so that interconnection is a weight, so called \hat{W} and y is the output of the neuron. So, what does the neuron do it calculates it is net input as a weighted sum of inputs, so this is input coming to the neuron, so multiply that with the weight and sum over all the inputs that we call the net input.

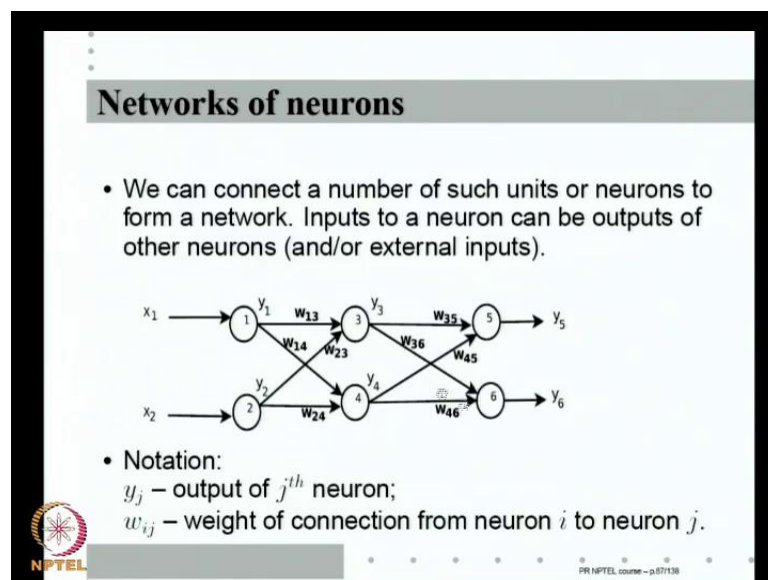
And you calculate your output as some function of the net input where f is called the activation function, this is this can be a simple model of a neuron, we have seen such models already in pattern. Perceptron is a very simple model, when we studied perceptron are actually said that it comes from some simple neuron models. So, essentially if I take this activation function to be a step function, this exactly over the perceptron does, it takes the weighted sum and thresholds it at some point.

So, if the if the net input is greater than some threshold output is 1, net input is less than the threshold output is 0. We can think of this as these are the incoming charge injections

into the neuron, these are the synaptic strengths, so at the soma the net input represents the total increase in potential and if it is above some threshold you find neuron action potential. So, but viewed simply as a mathematical model within the structure, if I take f to be a step function I get a perceptron model.

If I take f to be a linear function, we get that adapt to linear element model which is what we said is the original L M S algorithm is based on. So, these models as I said already are used in pattern recognition at that time we did not talk of them as a neural network models, but these are what we have considered as our linear models can be simply thought of as single neuron models, but really the reason why these neural networks are more powerful is that you can interconnect them.

(Refer Slide Time: 45:19)



We can connect a number of such units to form a network, so when we when we say network what means is output of one neuron can become input to another neuron like this. So, if I take neuron 3 it takes inputs from neuron 1 and 2, calculates it is output we call it y_3 and that itself now forms inputs in neurons 5 and 6. So, in this diagram the notation is y_j is the output of the j -th neuron and w_{ij} is the weight of connection from neuron i to neuron j , do not read anything into the bold characters.

The weights are in bold only because you know the diagram is crowded and if I did not input in bold it was becoming very difficult to notice them, that is the only reason the weights are put in bold, so in all my diagrams a symbol being symbol being in bold

means nothing. So, again, so y_i is the output of j -th neuron y_j is the output of j -th neuron and W_{ij} that is our notation, so W_{13} connects neuron 1 to neuron 3 W_{36} connects neuron 3 to neuron 6 and so on. So, essentially a if you take this neuron, it gets some inputs from other neurons calculates output this output itself, now forms input to some other neurons and so on, so there is a network like this.

(Refer Slide Time: 46:47)

- Each neuron computes weighted sum of inputs and passes it through its activation function, to compute output
- For example, output of neuron 5 is

$$y_5 = f_5(w_{35} y_3 + w_{45} y_4)$$

NPTEL logo and footer: PR NPTEL course - p.09138

How does this network compute, each neuron computes weighted sum of inputs and passes it through an activation function to compute it is own output.

(Refer Slide Time: 46:55)

Networks of neurons

- We can connect a number of such units or neurons to form a network. Inputs to a neuron can be outputs of other neurons (and/or external inputs).

Diagram description: A neural network with 6 neurons arranged in two rows. Neurons 1 and 2 are on the left, 3 and 4 in the middle, and 5 and 6 on the right. External inputs x_1 and x_2 point to neurons 1 and 2 respectively. Weights w_{ij} connect neurons: w_{13} (1 to 3), w_{14} (1 to 4), w_{23} (2 to 3), w_{24} (2 to 4), w_{35} (3 to 5), w_{36} (3 to 6), w_{45} (4 to 5), and w_{46} (4 to 6). Outputs y_1 through y_6 are shown for each neuron.

- Notation:
 y_j – output of j^{th} neuron;
 w_{ij} – weight of connection from neuron i to neuron j .

NPTEL logo and footer: PR NPTEL course - p.07138

So, for example, if we go to neuron 5, how does neuron 5 calculate its output, it takes output of neuron 3 multiplies with w_{35} output of neuron 4, which is y_4 multiplies with w_{45} and passes it through its own activation function.

(Refer Slide Time: 47:15)

- Each neuron computes weighted sum of inputs and passes it through its activation function, to compute output
- For example, output of neuron 5 is

$$y_5 = f_5(w_{35} y_3 + w_{45} y_4)$$

$$= f_5(w_{35} f_3(w_{13} y_1 + w_{23} y_2) + w_{45} f_4(w_{14} y_1 + w_{24} y_2))$$
- By convention, we take $y_1 = x_1$ and $y_2 = x_2$.

NPTEL logo and footer: PR NPTEL course - p 91138

So, y_5 is f_5 the activation function of the neuron 5 of w_{35} into y_3 plus w_{45} into y_4 , now how are y_3 and y_4 obtained.

(Refer Slide Time: 47:28)

Networks of neurons

- We can connect a number of such units or neurons to form a network. Inputs to a neuron can be outputs of other neurons (and/or external inputs).

Diagram description: A directed graph with 6 nodes (neurons) labeled 1 to 6. Node 1 receives input x_1 and produces output y_1 . Node 2 receives input x_2 and produces output y_2 . Node 3 receives inputs from nodes 1 and 2, producing output y_3 . Node 4 receives inputs from nodes 1 and 2, producing output y_4 . Node 5 receives inputs from nodes 3 and 4, producing output y_5 . Node 6 receives inputs from nodes 3 and 4, producing output y_6 . Weights are labeled on the edges: w_{13} (1 to 3), w_{14} (1 to 4), w_{23} (2 to 3), w_{24} (2 to 4), w_{35} (3 to 5), w_{36} (3 to 6), w_{45} (4 to 5), w_{46} (4 to 6).

- Notation:
 y_j – output of j^{th} neuron;
 w_{ij} – weight of connection from neuron i to neuron j .

NPTEL logo and footer: PR NPTEL course - p 87138

They themselves are obtained in the same way, y_3 is obtained by multiplying y_1 with W_{13} y_2 with W_{23} and passing it through the activation function of y_3 and similarly, y_4 .

(Refer Slide Time: 47:48)

- Each neuron computes weighted sum of inputs and passes it through its activation function, to compute output
- For example, output of neuron 5 is

$$y_5 = f_5(w_{35}y_3 + w_{45}y_4)$$

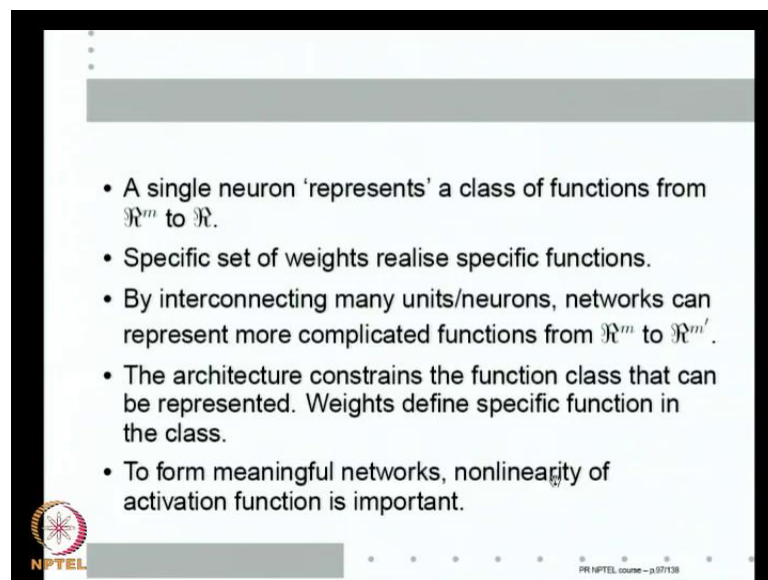
$$= f_5(w_{35}f_3(w_{13}y_1 + w_{23}y_2) + w_{45}f_4(w_{14}y_1 + w_{24}y_2))$$
- By convention, we take $y_1 = x_1$ and $y_2 = x_2$.
- Here, x_1, x_2 are inputs and y_5, y_6 are outputs.

So, if I expand that this y_3 and y_4 can be expanded to y_3 becomes f_3 of $W_{13}y_1 + W_{23}y_2$, and y_4 becomes f_4 of $W_{14}y_1 + W_{24}y_2$, so as you can see the this output is now a function of all the weights and here the input is y_1 and y_2 . And these are obtained through composition, so I have linear sum and some activation function, now once again a linear sum over such things and an activation function. So, I have used composition of linear sums and non-linear activation functions to build more and more complicated functions.

(Refer Slide Time: 47:28) By convention in networks like this, we take these as the input neurons we will look at the full notation later on, but essentially for the input neurons it is output will be input, so it is just a find over device. So, we will take by convention y_1 is equal to X_1 , y_2 is equal to X_2 , so the expression we wrote for y_5 directly gives y_5 as a function of the inputs, inputs X_1 and X_2 .

(Refer Slide Time: 47:48) So, that is the if I take y_1 is equal to X_1 , y_2 is equal to X_2 y_5 is a function of $X_1 X_2$, it is a kind of non-linear function because the $f_3 f_4 f_5$ are all non-linear, the w 's are all the parameters of the function. So, $X_1 X_2$ are inputs $y_5 y_6$ are outputs, so that particular network model represents some function, from r_2 to r_2 .

(Refer Slide Time: 49:27)



So, a single neuron as we have seen because it has only one output represents a class of functions from some m inputs to a single real output r_m to r . Specific set of weights realise specific functions right you have perceptron can represent any hyper plane, which hyperplane it represents depends on what weights we put on a perceptron. By interconnecting many neurons we get more complicated functions now from r_m to r_m prime, we have seen an r_2 to r_2 function network.

So, we can get more complicated model that can represent functions from r_m to r_m prime, the architecture of the network will of course, constrain the function class that can be represented. We will look at it later what kind of architecture can represent what function classes, but the architectures in some sense is reasonable to assume that

constrains the class of function that can be represented whereas, specific weights define a specific function within that class.

(Refer Slide Time: 50:30)

- Each neuron computes weighted sum of inputs and passes it through its activation function, to compute output
- For example, output of neuron 5 is

$$y_5 = f_5 (w_{35} y_3 + w_{45} y_4)$$

$$= f_5 (w_{35} f_3 (w_{13} y_1 + w_{23} y_2) + w_{45} f_4 (w_{14} y_1 + w_{24} y_2))$$
- By convention, we take $y_1 = x_1$ and $y_2 = x_2$.
- Here, x_1, x_2 are inputs and y_5, y_6 are outputs.

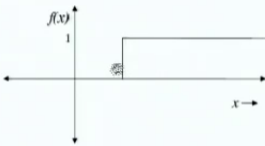
And finally, to form meaningful networks nonlinearity of the activation function is important, look at this function if suppose f_3 , f_4 and f_5 were all linear functions. Then f_3 of this is some linear function of y_1 , y_2 , which is X_1 , X_2 , f_4 of this is some linear function of y_1 , y_2 . So, I can write the whole thing as a linear function of X_1 , X_2 , now if f_5 is also linear the whole thing is a linear function of $(())$.

So, essentially it represents no more than what a perceptron can even, if f_5 is non-linear if f_3 and f_4 are linear, what is inside here is simply a linear sum a linear function of X_1 , X_2 on the weights. So, essentially it is representation capabilities these are no different from what a single neuron can represent, so nonlinearity of the activation function is very important, typical activation functions used in this models are things that we have already seen one is what is called a hard limiter.

(Refer Slide Time: 51:16)

Typical activation functions

1. Hard limiter:


$$f(x) = \begin{cases} 1 & \text{if } x > \tau \\ 0 & \text{otherwise} \end{cases}$$

- We can keep the τ to be zero and add one more input line to the neuron. An example of a single neuron with this activation function is Perceptron.

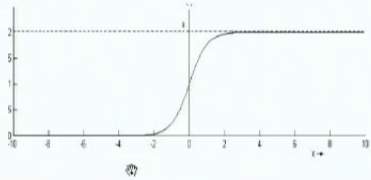
NPTEL PR NPTEL course - p.09/138

Essentially, if the net input the the argument to the activation function is greater than some threshold, then output is 1 otherwise it is 0. Of course, as we have already seen in the perceptron case I can always take tau to be 0 by pushing tau to the input side we will see that later on. Another activation function that we seen is the sigmoidal activation function, so the hard limiter is not differentiable, sometimes it may give us problems, that we shall see later on.

(Refer Slide Time: 52:00)

Activation functions (cont).....

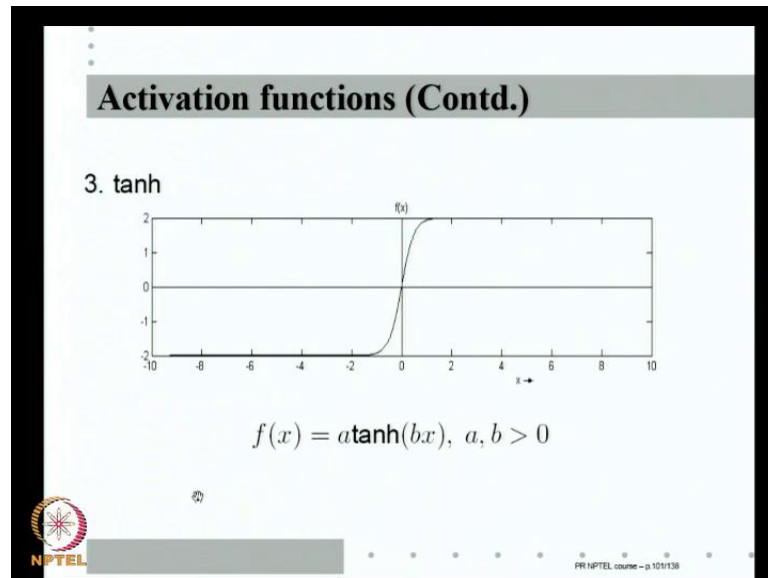
2. Sigmoid function:


$$f(x) = \frac{a}{1 + \exp(-bx)}, \quad a, b > 0$$

NPTEL PR NPTEL course - p.100/138

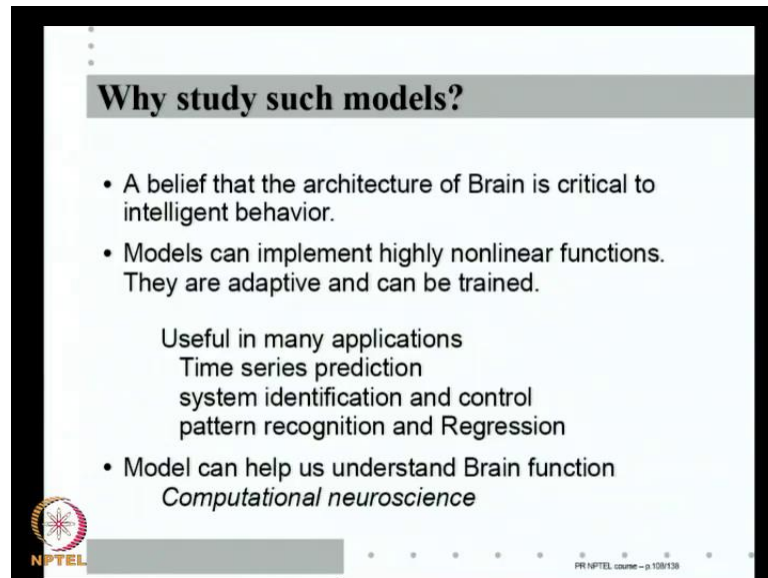
So, we can make that into a nice smooth differentiable function that is a sigmoid. This is $\frac{1}{1 + e^{-bx}}$, I can always scale it to a , here a is actually 2 this is the rough shape of it, and the coefficient b determines this slope of it near 0. The sigmoidal function takes only from 0 to 1, so it cannot take both positive and negative values.

(Refer Slide Time: 52:25)



If I need to take both positive and negative values, I can take hyperbolic tangent which has roughly the same shape as sigmoid, but goes from positive to negative. It actually goes from minus 1 to 1, but I can put any amplitude there, and also I can put a parameter b to control the slope near 0, so these are the three most often used activation functions neurons.

(Refer Slide Time: 52:49)



Why study such models?

- A belief that the architecture of Brain is critical to intelligent behavior.
- Models can implement highly nonlinear functions. They are adaptive and can be trained.

Useful in many applications
Time series prediction
system identification and control
pattern recognition and Regression

- Model can help us understand Brain function
Computational neuroscience

NPTEL PRR NPTEL course - p 100/138

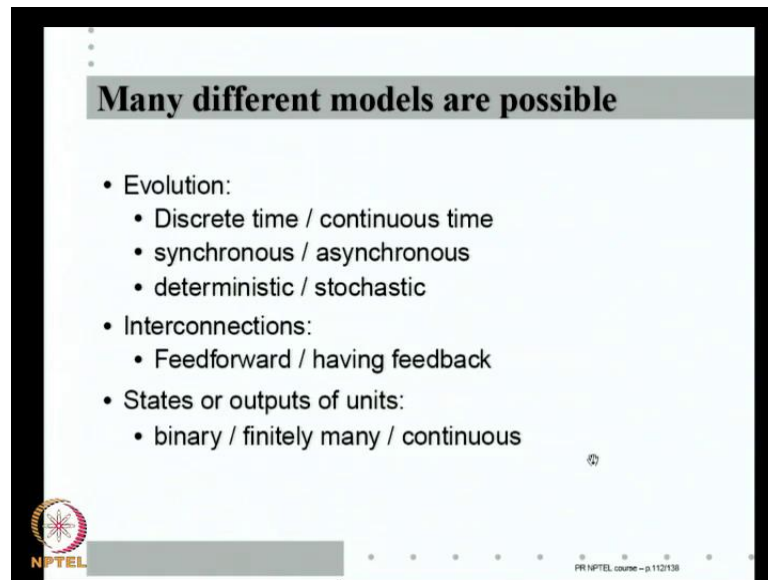
Now, why does one study such models, one we have already seen that believe that the architecture of the brain is somehow critical to intelligent behaviour, so if I mimic the architecture and sees what models of this architecture can do it may tell us something about how intelligent behaviour may come about the models. As we have seen implement highly non-linear functions, they are adaptive and can be trained by learning the weights.

So, it gives us a good class of non-linear functions, that is useful in many applications time series prediction, system identification control and most importantly for us pattern recognition regression. There can be a third goal models can help us understand brain function, so actually I can put models like this then tweak them to be realistic to some specific neurons in terms of how many synapses they have maybe, I can put some non-linear synapses.

And then try to see what the model predicts and then go back to experiments see, if it is if it can be observed in real neural tissue then once again tweak the model and so on. As a matter of fact there is a field, now name computational neuroscience, theoretical neuroscience which actually does this, and they do much more sophisticated models of neurons than what we considered. They actually take care of lot of nonlinearities lot of models for how the potentials are transmitted through the neuron through the dendrites and so on. So, there are methods of using such models as a way of understanding brain

function, so of course, we are not interested in any of these as we have already said we are just interested in these models as some nice parameters class of functions for doing pattern recognition regression.

(Refer Slide Time: 54:36)



Many different models are possible

- Evolution:
 - Discrete time / continuous time
 - synchronous / asynchronous
 - deterministic / stochastic
- Interconnections:
 - Feedforward / having feedback
- States or outputs of units:
 - binary / finitely many / continuous

NPTEL logo and footer text: PR NPTEL course - p112138

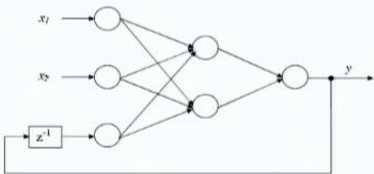
Of course, as you the the network that we considered is only one simple model is can be called a feed forward model because there are no feedback connections. Let me let me come to feedback later, but let us ask what are all the possible models, we can think of one is whether the models work in discrete time continuous time or all the units change, compute their outputs together in a clock fashion like a digital computer or a everybody does at some random time of their choice.

By the activation function deterministic stochastic, this is one kind of choices we have, we have choices in terms of architecture where the interconnections are only feed forward or also having feedback, whether the outputs of units are binary. They take only finitely many values they are continuous functions there are many different combinations and all combinations are combinations are possible.

(Refer Slide Time: 55:34)

Recurrent networks

- The network we saw earlier has no feedback.
- Here is an example of a network with feedback



- Can model a dynamical system:

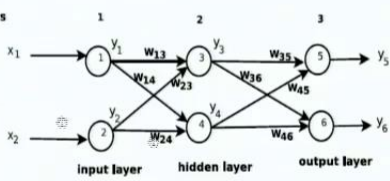
$$y(k) = f(y(k-1), x_1(k), x_2(k))$$

NPTEL PR NPTEL course - p 115/138

So, for example, here is a simple network is if the rest of it looks like a previous network we have concerned the outputs go only in one direction, but output of some unit can come back. So, we can think of this as using z inverse as a usual unit symbol, what it means is the output y at at time k is some function represented by the network of some input $X_1 k$ another input $X_2 k$ and y of k minus 1, so this can model a dynamical system. So, many of these models are used for non-linear dynamic system modelling also.

(Refer Slide Time: 56:12)

- We will consider only feedforward networks which provide a general class of nonlinear functions.
- These can always be organized as a layered network.



- This network represents a class of functions from \mathbb{R}^2 to \mathbb{R}^2 .

NPTEL PR NPTEL course - p 116/138

Of course, as I said we considered only feed forward networks, which provide a general class of non-linear functions. By feed forward this is the network model that we considered last time, so each each neuron calculates it is output and sends it to downstream neuron, there is no feedback connections, because there is no feedback connections we can always organise them as layers. Such that a neuron in one layer gets it is input only from the immediate previous layer sends it is outputs only to neurons in the next layer.

As we can always do this a little thought will tell you, because if suppose neuron 3 has to also get neuron 2's output, I can simply put a dummy unit here, which only gets neurons two's output and then passes it on to neuron 5. So, the by by adding such dummy units if I need I can think of this network always having a layered structure, so I call this layer 1, I call this layer 2, I call this layer 3 any neuron in any layer gets inputs from the immediate previous layer, and sends it is output only to the immediate succeeding layer.

We call the layer one as the input layer and that takes inputs from the external system, last layer as the output layer which sends it is outputs to the external world, everything in between is a hidden layers, so this we call hidden layers later on I will tell you why the name hidden. So, this particular network represents some class of functions from \mathbb{R}^2 to \mathbb{R}^2 , it has 2 inputs and 2 outputs, so if I put this entire thing into a black box, it is it has 2 inputs, 2 outputs, it represents some function from \mathbb{R}^2 to \mathbb{R}^2 .

(Refer Slide Time: 57:44)

- Each unit can also have a 'bias' input.
- This is shown for a single unit below.

The diagram shows a circular neuron. On the left, there are vertical arrows representing inputs labeled x_1, \dots, x_d . These arrows point to a central circle. The arrows from x_1 and x_d are labeled with weights w_1 and w_d respectively. Below the circle, there is an upward-pointing arrow labeled $+1$ with a weight w_0 next to it. An arrow labeled y points out from the right side of the circle.

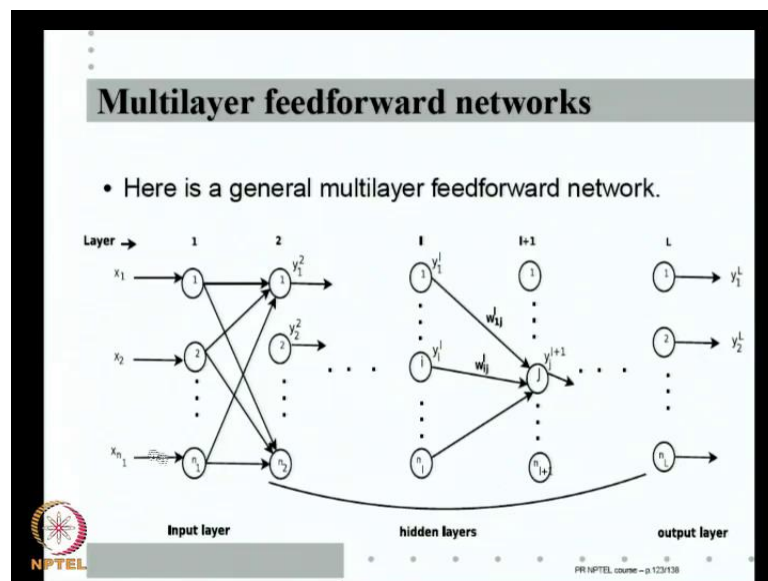
- One can always think of bias as an extra input

$$y = f\left(\sum_{i=1}^d w_i x_i + w_0\right) = f\left(\sum_{i=0}^d w_i x_i\right), x_0 = +1$$

PR NPTEL course - p 122/138

Each unit can also have a bias input, so earlier we only considered only this, but we can have y as a function of $W_i X_i$ plus some biased W_{naught} . We can always think of bias as a extra input because I can make this summation going from instead of 1 to d , I can go it from 0 to d , and call W_0 as the weight and an extra connection which is permanently connected to an input that is permanently kept at plus. So, I think of X_0 as a extra input which is permanently plus 1 and I can represent it as a bias input.

(Refer Slide Time: 58:24)



So, in general we can have feed forward networks like this. So, what we are going to study next class is now we will forget all about intelligent machines and everything, we will just look at these as a non-linear class of functions. So, there are layers, so each neuron completes its output and sends through a succeeding layer, so using and we can have many number of layers many number of things. So, using such models we will ask how we can represent and learn non-linear functions that are useful in pattern recognition regression.

Thank you.