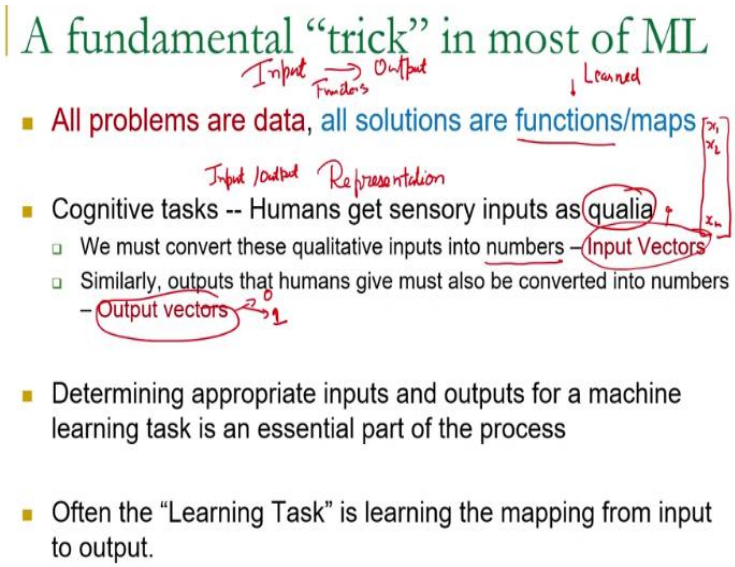
Inverse Methods in Heat Transfer
Prof. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology – Madras

Lecture - 45
Supervised Machine Learning as an Inverse Problem

Welcome back. This is week eight of inverse methods in heat transfer. This week we are looking at a brief introduction to machine learning and how it is relevant to inverse methods in heat transfer. so far in the last video, I showed you what supervised learning was, what unsupervised learning was and what the general machine learning idea is. In this video we will dive in a little bit further into the detail of what actually makes machine learning specifically a certain type of supervised learning work, which is what we will be reusing again and again in the next few weeks and we will also see that effectively it turns out that this approach is identical to the approach that we have been taking so far in the last seven weeks with inverse problems in general. So, this is a strong connection between inverse methods and machine learning.

(Refer Slide Time: 01:12)



So, there is a trick effectively, this is also the trick that we use in inverse methods, that works in most of machine learning. the idea is this any problem that I give you is treated as if it is an inverse problem or as if it is a data problem and every solution that I offer to it is effectively a learned function. so, this function that we look at is a learned function. what do we mean by this? we will see some further details of this next week. but let us say I am looking at an example of being given an image and trying to recognize what it is? as a human being a human being

sees what is known as qualia. okay qualia mean we have certain qualitative idea about what an image is, what a sound is.

so, when we look at our mother, we look at our child we look at our spouse or we look at our friend we do not immediately see friend we actually see a sensory input there is some feeling of seeing and this feeling of seeing is what we wish to convert into a number which tells us whether this is our mother, whether it is our brother, whether it is our spouse or whatever it is. this thing that human beings get input as qualia has to be converted into a language that the machine understands now this input is converted into numbers okay. so that is the first trick that happens within machine learning now this technique or this trick is really not required in most inverse problems.

Most inverse problems we generally directly deal with numbers. especially the things that I gave you so far, I was asking you well if the temperatures are this and this then what is the heat flux okay. so, this we are directly dealing with numbers but in certain cases let us say you have an MRI scan you have an image now the image appears a certain way to the doctor but it appears in an entirely different way to the machine which I will show you in the next few slides. but to the machine everything you give whether it is a sound signal, whether it is an image, whether it is a text piece of text even something as clear as what is written here is only a bunch of numbers to the image and this is what we call an input vector. so, remember a vector here in the language that we are using is simply a bunch of numbers.

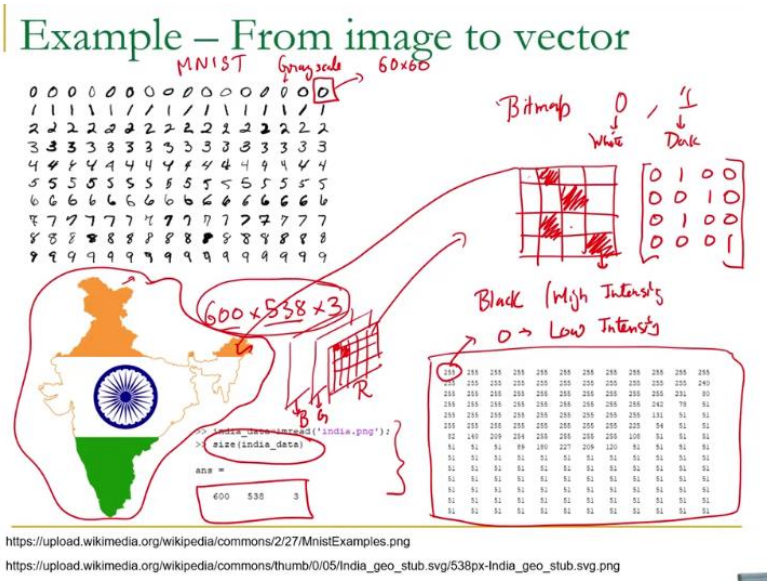
So, if I show the slide this slide is stored internally in a bunch of zeros and ones within the machine and that can be represented effectively as a bunch of numbers okay. so, this is called an input vector the x vector or what we see is an input vector. Similarly, outputs that we generate, so let us say you have an email that email goes into this pipeline and somehow gets converted to a bunch of numbers and we want to say whether it is spam or not, so we cannot simply say spam or not spam but what we want to say is an output vector. so, we can say something like zero for spam or one for not spam or if you want to say whether this person has covid or does not have covid, we say zero for not covid and one for covid.

So, we will see other such representations. This thing is called the idea of representation okay input and output representation. this can be a particularly powerful thing for machine learning unfortunately we want again get into it within this course but it is an important thing to

remember the fact that what we see or what we experience in real life is somehow converted into numbers which are ultimately posed in the inverse problem as input numbers and output numbers okay. now this itself is an essential part of the process and if people do it cleverly especially for tasks like language, you will have to do this really cleverly how do you represent a set of emails as a number or how do you represent an emotion as a number.

All these things for outputs so such things are a clever part of the machine learning algorithm design part. but like I said it is a representational idea okay. but usually when we say we are learning all we are learning so this is where the trick happens, I post the learning problem as if I am simply learning a function so I have an input which could be an image which could be an email or whatever or a sound which you want to finally recognize and this has to be turned to an output and we know that things that turn to from inputs to outputs are essentially functions. so, all we are trying to learn effectively the learning task is learning that function.

(Refer Slide Time: 05:58)



https://upload.wikimedia.org/wikipedia/commons/2/27/MnistExamples.png

https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/India_geo_stub.svg/538px-India_geo_stub.svg.png

So here let me show you an example of how we can do representation. so here is a bunch of numbers, let us say this is what is called from the MNIST data set. I will not get into it in greater detail here but these are some black and white or grayscale images. so each one of these images if I cut this image of zero out, this will usually be something like a sixty cross sixty or twenty seven cross twenty seven image but let me just take I for some reason I like this number sixty cross sixty so I tend to use this a lot so let us assume there is an image and there are sixty pixels on each side so if you take your camera you will say so three mp camera means three megapixels means three into ten power six pixels. so let us look at the detail of what happens within each

one of these pixels. so here is an image of India unlike this which is a grayscale image. this is a color image okay.

Now when it appears to our eyes, it appears as one integrated color image whereas if you actually try to see the size, I am not sure if it is visible this is kind of small I took this data within MATLAB and I tried to find out what the size of the data is and what you will see here is these numbers six hundred cross five thirty eight cross three. so as far as the machine is concerned it basically stores three arrays and those three arrays of numbers, so you can see the huge number of digits that is represented by the single image and these numbers each have a meaning. that meaning is kind of shown here again whether it is visible or not I will show more about this in the next slide, but essentially these are for a color image these are three channels r, g and b.

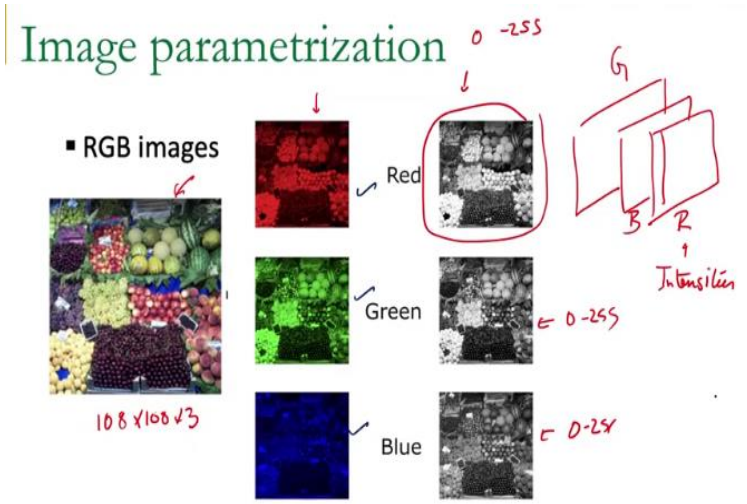
Now the way this works is that, there is a grid and within that grid you store these numbers. so, if you hit two fifty-five, let us say in this case it is white, but let us say this is black or really dark, high intensity and let us say when it is zero, then it is low intensity. okay this is not usually the standard way things are done but let us say that is the way things are done now what will happen is let me give a simple example of what is known as a bitmap in image instead of zero and two fifty five, if we if I simply have two values which is zero and one and let us say zero means white and one means dark if I have a four cross 4 four image, then something of this sort we will take an example of this again next week. you can now store this as numbers.

So, what will this number? This will simply be an array which will say,

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

So, black is one and white is zero in in the notation or the representation I have chosen. now you can think of scaling this up to an image of this sort.

(Refer Slide Time: 09:40)



What happens here of course is you have six hundred cross five thirty-eight numbers and each one of them as shown here, are in three different channels. so let us take this image let us say just as an example that this is a hundred cross hundred eight cross hundred eight cross three image. these three means there are 3 three channels, there is a red channel, there is a blue channel and there is a green channel okay. now the red channel simply talks about intensities in red.

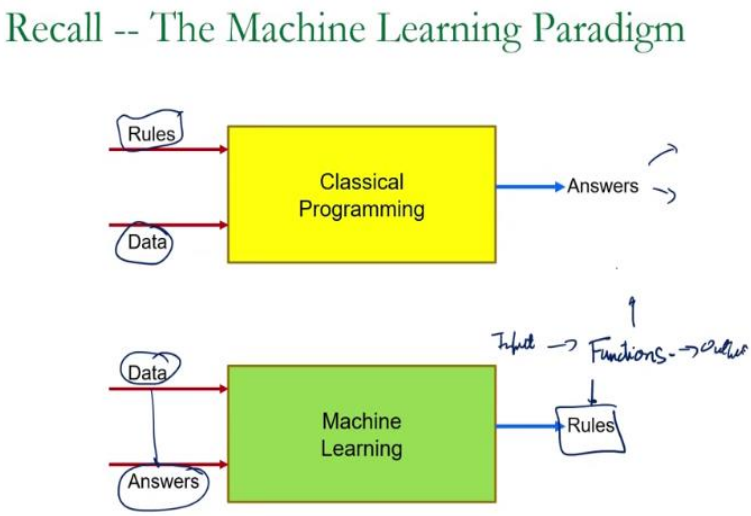
So, you can see here this image itself even though within our eye for example there are three cones, there is a red cone, blue cone, green cone. if in case somebody is color blind, these things tend to work a little bit more differently. but in case you do not have a color blindness problem, you will have these three cones working fine. but within red all it is a bunch of numbers between zero to two fifty-five, in case you have an eight-bit representation. similarly, between zero green this will be zero to two fifty-five and this will also be zero to two fifty-five. so instead of having just like in the previous slide we had this bitmap which was zero or one imagine now you have numbers continuous numbers between zero and one and you have what is known as Grayscale.

That is like the image here. so not only are you allowed zero that is white and black, you are also allowed shades of gray okay so you could have shades of gray, so a light shade, a darker shade, a milder shade so you could have numbers like zero zero point two one point six so similarly here we scale it up between zero to two fifty five it is easier to work in binary sort of sort of two fifty six being two power eight so it is easier for you to work with that with images rather than work between zero and one. but anyway, you can think of this as if zero is white and 1 one is complete black and you can think of scaling. now what happens in our eye of course is

that when you see an image it combines these three intensities that is this red intensity, this green intensity and this blue intensity and we get a composite color,

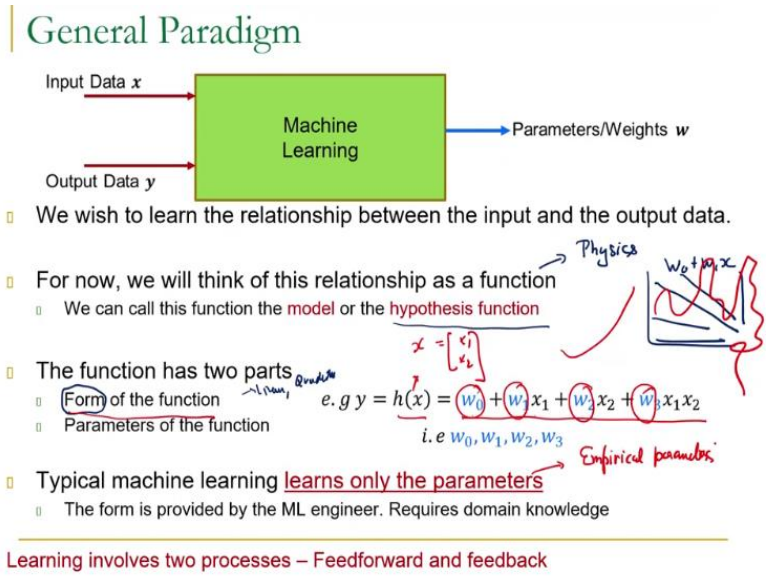
But as far as the machine is concerned it sees only numbers like the ones that I showed here. now when I look at these numbers, I cannot see an image and similarly when the machine sees an image it cannot or on when this machine sees numbers it cannot see the image either. but when I see the image, I cannot see the numbers and when it sees the numbers it cannot see the image. so, because of this combination we need a translation between the machine and ourselves and which is what machine learning tends to do automatically. but as far as the image is concerned, we give just the numbers to it. so, I hope this portion is clear.

(Refer Slide Time: 12:41)



Now once we accept the fact that the data that the machine takes is always numbers. remember that in the previous video I had talked about the machine learning paradigm. I had said that in classical programs in case you want to say okay given an image find out is this laminar flow or turbulent flow. then you will have to give some rules. if the numbers in the input combine this way, then say laminar otherwise say turbulent or when the words in email appear this way say spam or say not spam, whereas when we come to machine learning we simply say for these numbers the flow was laminar, for these numbers the flow was turbulent and you have to learn these rules and these rules as you will see in the next slide are functions. why are these functions? because functions can take one set of numbers and give out another set of numbers.

(Refer Slide Time: 13:43)



So, the idea is very simple and in case you remember what I talked about in the general inverse process, I will talk about this shortly once more. you should be able to see the connections between what I am showing here and what we have been doing for the last seven weeks. so, what we want to do is to learn the relationship between input data and output data okay. like I said image and the class of the image sound and where that sound comes from etcetera. for now, let us think of the relationship as a function. now in typical inverse problems this function was derived from physics. but right now, we are assuming we do not even have access to the physics of the problem okay.

So that is why we are going to just take a hypothesis okay. so, I call this the hypothesis earlier also which is why we had this h or y hat etcetera sitting earlier. but any typical function has two parts, that is what we call the form of the function okay. what is the form of the function? the form of the function says is this linear, is this quadratic. so why do I call it form of the function? if I have a function looking like this and I have any arbitrary parameter, let us say w_0 and w_1, so let us say this is the function $w_0 + w_1 x$ now the only thing that I can do by changing w_0 and w_1 is change the shape like this. but I cannot get an entirely new shape of this sort okay. so that is not possible. so, a linear function cannot get a quadratic shape. because of this our job is to fix the form first okay.

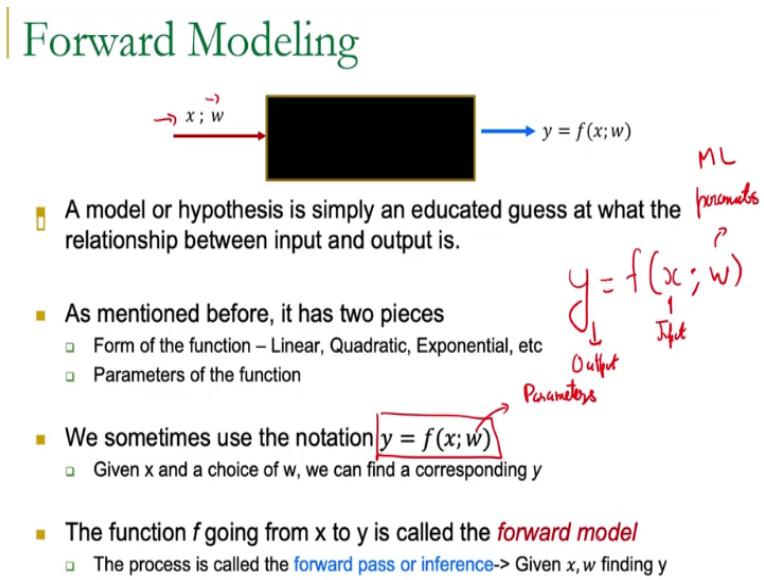
You have to decide on how many degrees of freedom this function is going to have okay, is it going to wiggle a lot or is it going to be straight etcetera. how do we make this choice this we will come to later on but for now, at least remember the general paradigm that, the form of the function is just deciding whether it is linear nonlinear whether there is a sine function whether

there is a cosine function etcetera okay. so we say $h(x)$, let us say x has two variables $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, so deciding on the form of the function is something like saying $w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2$ on the other hand the parameters of the function are this now changing the parameters is not going to change the form just because I change w_0 and w_1 my linear is suddenly not going to become nonlinear okay it will still stay linear however in order to fully specify the function, I have to give the knobs or the weights w_0 and w_1 also okay.

Now when we look at typical machine learning and here is where the strong intersection with the inverse problems happens you are only learning parameters in inverse problems you can learn some things like heat flux, you can just like we did the last week or you can learn conductivity stuff of that sort. these parameters could simply be empirical parameters. the form on the other hand is provided by the machine learning engineer and of course this requires domain knowledge, in case you are doing from physics or even in other cases you can be a little bit subtle, but that might be beyond this course.

But typically, this learning involves like we have been seeing just with inverse problems it involves feedback and feedforward.

(Refer Slide Time: 17:24)

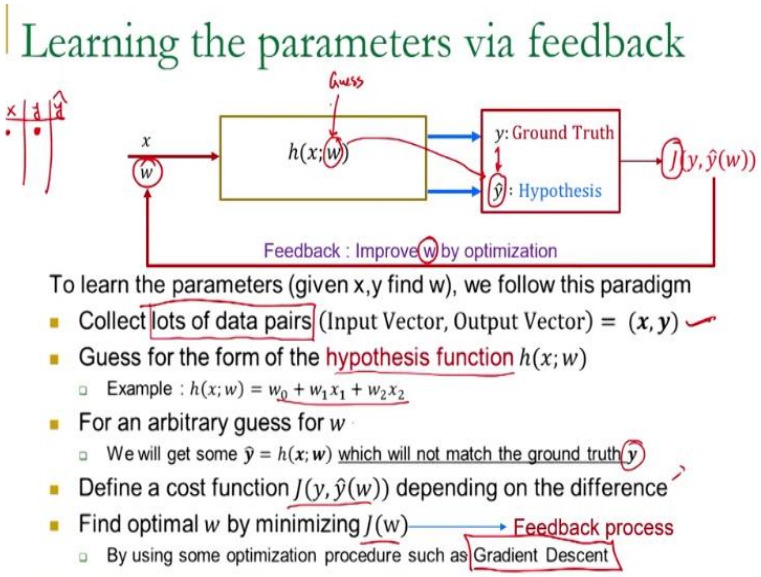


So, the forward modeling is just like what we have been doing so far okay. so, we know use this notation $y = f(x; w)$, w are the parameters. I used this earlier even during probabilistic modeling. I will write this a little bit bigger, so function of x, so the output is a function of the input with some unknown parameters. these parameters are in the machine learning parameters,

when you are given x and you are given w, as we did earlier, let us say with nonlinear regression, you give a guess for a parameter and give an input you find out what the corresponding output for that.

So once you give the x and the w you go forward you do the forward model and you go to w and usually we have been doing this forward model through physics okay.

(Refer Slide Time: 18:32)



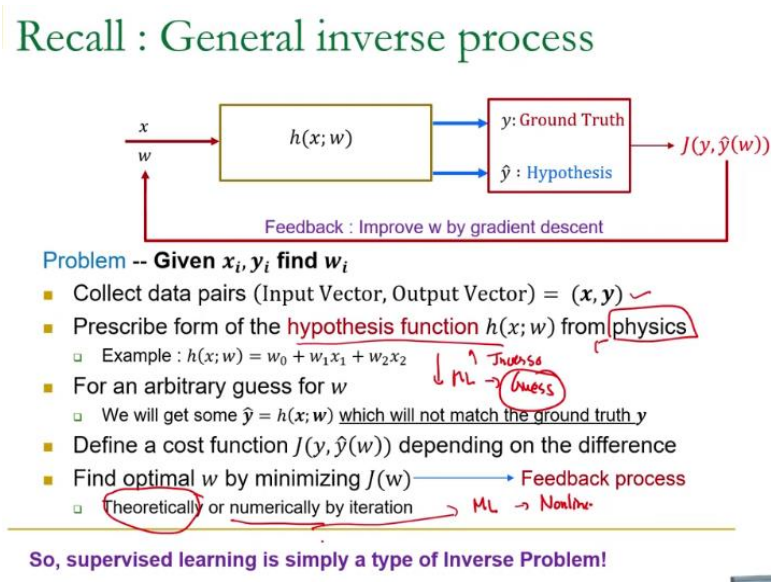
How do we learn the same parameters via feedback, we have done this a whole lot of times. but let us see this again in the context of machine learning. you have a data set you give some x, there is some truth y, because you know this parameter, you are guessing for this parameter to start with, you can guess a corresponding y hat this y hat and y will have a gap which we call J and we improve J by optimization, we improve w sorry by optimization okay. so, as I have written here you collect a lot of data pairs. now here is the catch the key difference between machine learning and some of the techniques that we use for inverse problems will come as we come to the next week because typically machine learning involves a lot of data points whereas inverse problems at least the simple problems that we have been doing even in real life measurements your number of measurements will actually not be too many okay.

So many of the techniques in machine learning have been optimized for large data sets and we are including this within this course just so that as the data that we actually collect becomes more and more massive. for example, weather data, radar data, other assimilation data this becomes huge you start moving towards from very general inverse techniques to very specific machine learning techniques so we will see some of those techniques next week.

Anyway, coming back to this point, you collect a lot of data points you have to guess for the form of the hypothesis function, for example linear, quadratic etcetera. then for an arbitrary guess for w, you take a guess, just like we did with nonlinear optimization in gauss newton etcetera you guess a value for w and the guess you take will generally not match the ground truth okay. then you define cost function depending on the difference. we saw one cost function which was simply least square. we will look at another cost function in the next week when you have to do classification and stuff like that and then you simply do a feedback process. you start minimizing J via gradient descent okay. so, we will concentrate on gradient descent and some of its variants in the next week.

But typically, in machine learning the industry standard is some version of gradient descent. gradient descent has many variants and many flavors so to speak. so, which we will see next week but the general process is simply you keep on improving J by doing gradient descent okay.

(Refer Slide Time: 21:20)



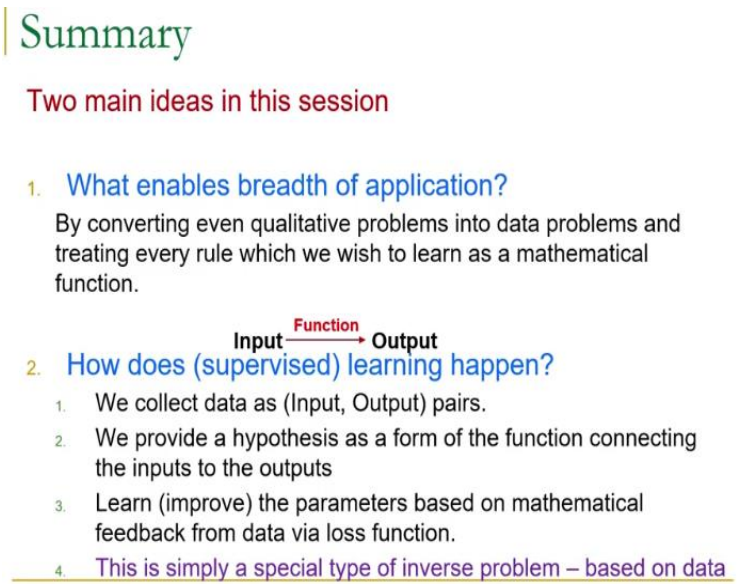
Now if you might remember I had almost exactly the same slide, when we were in, I forget when it was maybe week three or week four when I was talking about the general inverse process in fact it is the same thing you collect data pairs, I did not say lots of data points because inverses dependent on the measurement that we are doing. you prescribe the hypothesis function however this comes from physics; this is for general inverse.

If you are doing ml, you have to guess this form this hypothesis function. how do we do a systematic guess, we will come to that again okay now for so we are guessing for two things

the parameters as well as the form. so, in case of physics based inverse methods we are not guessing for the form. for the form we just simply just take a form from the physics of the problem. however, when we come to physics informed neural networks you will see an intersection between these two for an interesting idea which comes right between these two. now for an arbitrary guess again the end result will not matter define cost function optimize okay. now the other option which we had instead of gradient descent was we could do this optimization theoretically, in the case of a general inverse problem whereas in the case of machine learning you can only do it numerically okay.

So, ML usually will be always nonlinear okay generally this will be generally nonlinear and you can numerically obtain solutions only through iteration. but my reason for showing this slide was to show you that what I had shown you as the general supervised learning feedback process is exactly what we get in inverse. so basically, supervised learning can be thought of as a type of inverse problem okay. so, this is the important thing to remember.

(Refer Slide Time: 23:24)



So, there are two main ideas here the reason why ml is able to deal with a large number of applications is that even qualitative problems can be turned into data problems okay. basically, by treating every rule which we wish to learn as if it is a mathematical function.

And how does supervised learning happen, we collect input output data pairs, we provide a hypothesis as a form of the function connecting the inputs to the outputs and when we say we learn we are typically learning only the parameters. so, as I have mentioned here this is simply a special type of inverse problem based on data okay. so, you simply take an input put a function

get an output. now the plan earlier was to move on and show you how to apply this inverse process for linear regression using gradient descent. I did not demonstrate this in the earlier week and I also wanted to talk about some special features that come out when we especially start thinking about the scale of machine learning.

However, this week will be kept very sweet, very short and very simple you had a harder earlier week. so, the next week we will move on to doing both linear regression and what is known as logistic regression and you will see the entire process of inverse methods applied again to slightly maybe both familiar as well as unfamiliar problems. so, I will see you in the next week thank you.