

Pattern Recognition and Application

[View Slides](#) [View Expressions](#)

Prof. Prabir Kumar Biswas
IIT KHARAGPUR

INDEX

<u>S.No.</u>	<u>Topic</u>	<u>Page No.</u>
<i>WEEK- 1</i>		
1	Introduction	1
2	Feature Extraction – I	8
3	Feature Extraction – II	15
4	Feature Extraction – III	18
5	Bayes Decision Theory	20
6	Bayes Decision Theory	22
<i>WEEK- 2</i>		
7	Normal Density and Discriminant Function	26
8	Normal Density and Discriminant Function (Cond.)	37
9	Bayes Decision Theory – Binary Features	45
10	Maximum Likelihood Estimation	48
11	Probability Density Estimation	49
12	Probability Density Estimation (Contd.)	54
<i>WEEK- 3</i>		
13	Probability Density Estimation (Contd.)	59
14	Probability Density Estimation (Contd.)	64
15	Probability Density Estimation (Contd.)	69
16	Dimensionality Problem	74
17	Multiple Discriminant Analysis	79
18	Multiple Discriminant Analysis	82
<i>WEEK- 4</i>		
19	Multiple Discriminant Analysis (Tutorial)	87
20	Perceptron Criterion	92
21	Perceptron Criterion (Contd.)	99
22	Linear Machine	103
23	Liner Discriminator	110
24	Neural Networks for Pattern Recognition (Contd.)	118

WEEK-5

25	Neural Networks for Pattern Recognition (Contd.)	124
26	Neural Networks for Pattern Recognition (Contd.)	130
27	RBF Neural Network	135
28	Lecture No. – 28	141
29	Support Vector Machine	148
30	Hyper box Classifier	153

WEEK- 6

31	Hyper box Classifier (Contd.)	161
32	Fuzzy Min Max Neutral Network for Pattern Recognition	166
33	Reflex Fuzzy Min Max Neural Network	173
34	Clustering	178

WEEK- 7

35	Clustering (contd.)	182
36	Clustering Using Minimal Spanning Tree	191

WEEK- 8

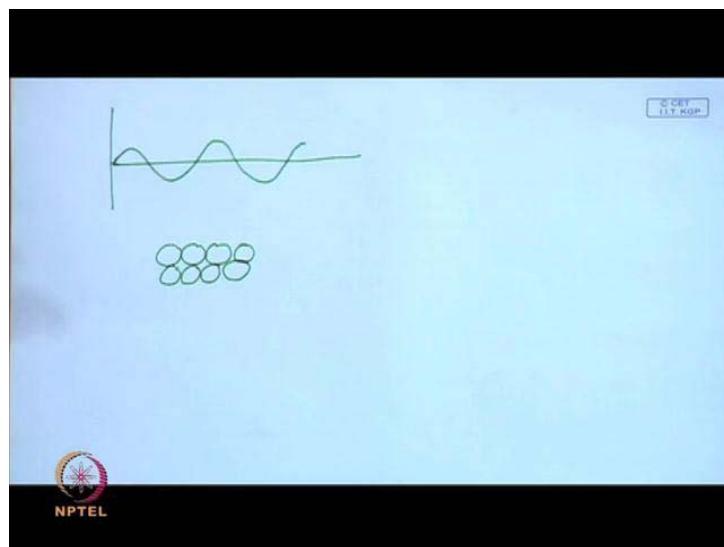
37	Temporal Pattern Recognition	196
38	Hidden Markov Model	202
39	Hidden Markov Model	207
40	Hidden Markov Model	220

Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 1
Introduction

Hello. Welcome to this course on pattern recognition and applications. So, when you talk about the problem of pattern recognition, let us try to see what is meant by pattern recognition or specifically what is meant by a pattern?

(Refer Slide Time: 00:45)



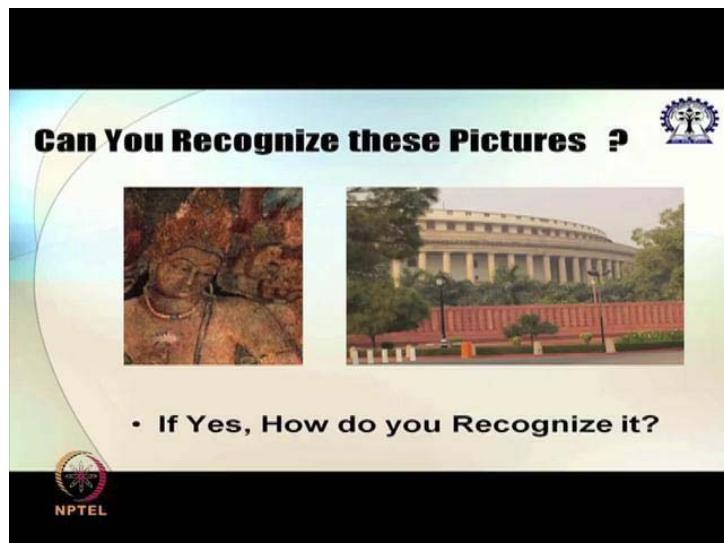
So, if I draw a simple diagram something like this. This is nothing but a pattern or if I draw diagrams like this, this is also a pattern. So, the problem of pattern recognition is given some signal whether it is one dimensional signal or a two dimensional signal. We want the machine to recognize that particular signal, so that is what the problem of pattern recognition is. Now, let us look at the origin of the pattern recognition problems; pattern recognition, though we talk of problem recognition problems these days, but the problem of pattern recognition is not that new. In fact, it started with in the early years with the effort of understanding intelligence. So, what is meant by intelligence?

We can crudely define and intelligence as the ability to comprehend or to understand and profit from experience. So, what do I mean by that ability to comprehend and to understand and profit from the experience? Let us take a very simple example, suppose I

have a sharp object, so if I take the tip of this pen, so I all of you can see this pen. If I take the tip of this pen, the tip of this pen is quite sharp or even if it is sharper than this. Say for example, the tip of a pin, which is very sharp and we know that if we touch or if we hit the tip of the pin in that case this hurts our finger. So, that is what is our experience and how do I profit from this experience? The profit is that whenever I come across such a sharp end I will not try to touch it hard, because I know that it will hurt me.

Similarly, if I take the example of fire, you know if I put my finger in the fire, the fire will burn my finger. So, that is the experience and how I profit from this experience is that, whenever there is fire I will not put my finger into the fire. So, that is what is my experience. So, as I said that the crude way of defining intelligence is to comprehend or to understand and profit from the experience. That is what is intelligence or we can say, also say that it is the capability and to acquire and apply knowledge. So, we should be able to acquire knowledge and I should be able to apply the knowledge that I have acquired. Now, let me show you two different pictures.

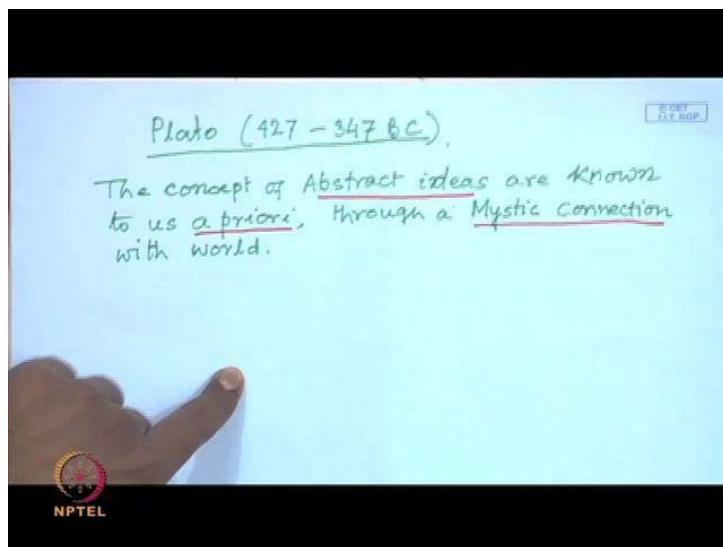
(Refer Slide Time: 04:14)



So, find that I have two pictures in the slide. On the left I have one picture and on the right I have another picture. So, if I ask you that can you recognize these two pictures, if you have seen it earlier, we will obviously say that, ok this left hand side the picture, which is on the left it is nothing but a painting. To be more specific, if I know I say that it is wall painting in Ajanta caves.

Similarly, if you look at the picture on the right, one will immediately say that this is picture of a building. But, some of us would have seen this building or we have seen this picture before. We will say that this is the picture of Parliament house in New Delhi. So some of us will be able to recognize it and the question is, if we are able to recognize these pictures then how do we recognize these pictures? Now, if I try to find out the answer to this particular question when I am able to recognize these pictures, then how do I recognize these pictures? The answer is not very new, in fact we have to go back almost 2005 years ago when Plato, who tried to give the answer to this, how do we recognize objects or how do we recognize pictures or how do we recognize patterns?

(Refer Slide Time: 05:57)



So, it is Plato. In early days, I mean it was in 427 to 347 BC, so you find that this is as early as more than 2005 years ago. And that time Plato said that the concept of abstract ideas are known to us a priori through a mystic connection to the world. So, if I write it the concept of abstract ideas are. So, you look at these two terms, one is abstract ideas, a priori and addition this is another term, which is mystic connection with the world. So, again if you look at these two pictures, we said that if we know we can say that this left hand is wall painting in Ajanta caves and the right picture is the picture of Parliament House in New Delhi.

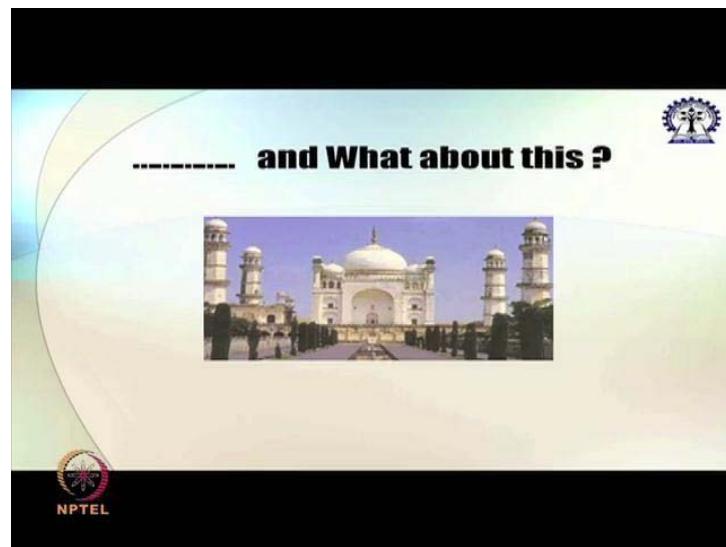
So, how do you know it in many cases, we have seen the pictures of wall painting in Ajanta caves in our school level history books, similarly some of us might have seen the Parliament House in New Delhi, or some of us might have seen the images or the photographs of the Parliament House in New Delhi. So, that means this information or what I say is, in this case it is an abstract idea, because I cannot specifically say that why do I recognize this to be a

painting in Ajanta caves and why do I recognize or how do I recognize this to be the image of Parliament House.

So, this concept is actual and abstract concept and I can identify this to be a painting in Ajanta caves, or the other one to be the image of Parliament House in New Delhi, because this information or this knowledge is already available to me a priori. So, what I am doing is I am making use of this a priori knowledge to recognize this two particular pictures. So, that is why these two terms that abstract ideas and a priori, these are very important and not only this as I recognize the painting to be a painting in Ajanta caves.

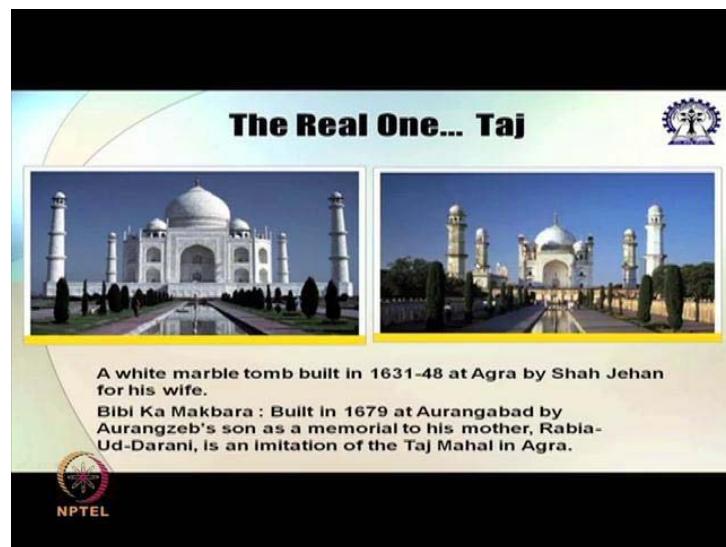
That means, this information is embedded or the image of the picture is embedded in my brain in some way. And what is that way that is not yet fully understood? So, that is why it is said that through a mystic connection with the world, so these abstract ideas of a priori knowledge are stored in our brain in a very very mystic way. It is mystery that how do we recognize or how do we store that information or later on how do we retrieve that information to recognize certain things. Plato has also concluded that the ability to think is, in a priori knowledge of the concepts. So, this is a priori term is told by Plato in many ways. Now, let us look at another picture.

(Refer Slide Time: 10:49)



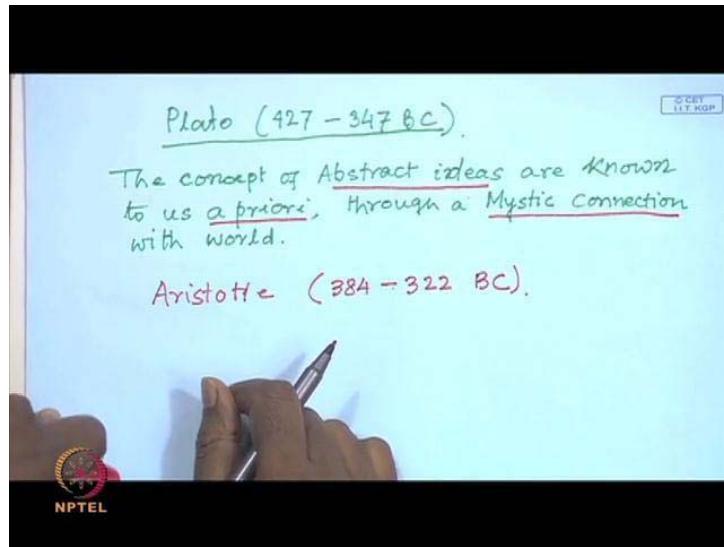
So, picture like this right? So, if we ask you that, what is this picture? You know that, most of us have seen that images of a monument something like this, which is Taj Mahal. So, if I look at this picture the immediate reaction will be, this is the picture of Taj Mahal.

(Refer Slide Time: 11:21)



Now, if I show you the next two pictures, where you find that the picture on the right is nothing but the picture that we have seen earlier and the picture on the left that is something different. So, when I show you these two, you will immediately say that picture on the left is the picture of the Taj Mahal, picture on the right is not really the picture of Taj Mahal, but its a monument, which is very much similar to Taj Mahal. In fact, this monument is what is known as Bibi ka Makbara and it is situated in Aurangabad and which was built by Aurangzeb's son. But, they are very much alike. So, you find this a priori knowledge, which has been proved by Plato is not sufficient we should be able to adapt our knowledge. That means the learning we do have or the knowledge, acquiring knowledge which we employe that much be incremental in nature.

(Refer Slide Time: 12:30)



So, as a result Plato's student Aristotle, which was in 384 to 322 BC, you see it is before Christ, it is almost 2005 years ago. So, Aristotle who was actually Plato's student he did not fully agree the concept of a priori knowledge, which was given by Plato. So, what Aristotle said that it is not only the a priori knowledge, which is very important, but also the ability to learn or to adapt to the changing world, that is also equally important. That means we should be able to adapt to the changing environment and our learning process must be adaptive. That is, we should be able to learn new and new things and we should be able to modify the knowledge that we have already acquired.

So, what is the problem of pattern recognition in that case? So, as we have seen earlier, say these two figures, they may represent some signal. They may represent certain structure and so on. So, the problem of pattern recognition is to identify the underlying structure within a data. So, if I have a given set of data I want to identify the structures within that data and what are the structures that I want to identify? It is the structures, which are known to me a priori, so that is what the pattern recognition problem is.

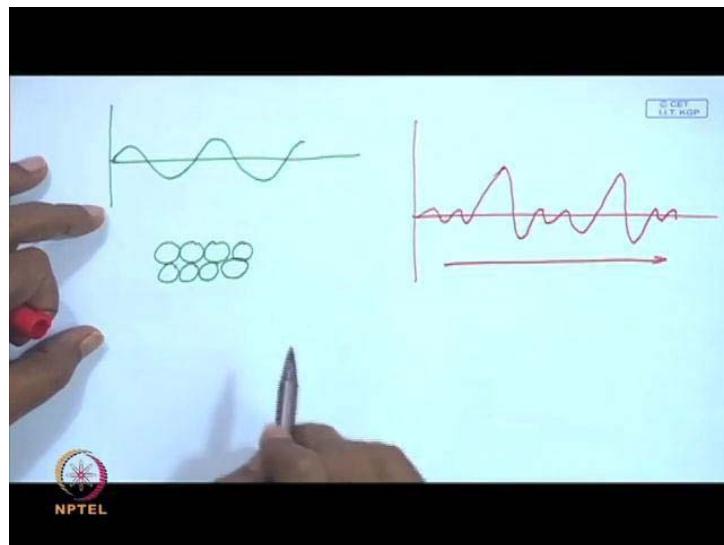
So, we can define purely the pattern recognition to be start of structure in the data, so that is what is why patterning recognition. Now, why do I want this pattern recognition, what is the importance of pattern recognition? I want this pattern recognition, so that I

can impart certain power to a machine, so that the machine can work in the same way in which I can do work.

So, I want that the machines should be equally intelligent as a human being and whatever work that we do, a machine or a computer should be able to do the same things. So, what are the different approaches of pattern recognition? So, in this introductory lecture, I will briefly tell about or I will introduce the pattern recognition problem. Then I will talk about what are the approaches of the pattern recognition problems or to solve the pattern recognition problems during this course of pattern recognition and applications.

Now, coming to the application side, the pattern recognition problem or the pattern recognition techniques have applications in various domains. For example, take for example, the medical signal analysis. So, each of us have seen that whenever a doctor or medical practitioner have any doubt about the functioning of the heart, whether our heart is functioning properly, they prescribe us to go for ECG checkup.

(Refer Slide Time: 16:33)



All of you know that when you get an ECG pattern, the ECG pattern appears something like this and it is a repeated pattern, so this is nothing but a structure. Ok? So, what the doctors do is, they look at these patterns and by looking at these patterns, they try to judge whether our heart function is proper or there is some abnormality in our heart function. So, this is just one of the example, where the pattern recognition is very useful.

Similarly, we can have the application of the pattern recognition in speech recognition or in speaker recognition. I can easily recognize the voice of Hemant Kumar and easily recognize the voice of Lata Mangeshkar, I can easily recognize the voice of Asha Bhosle and all those different people. But, all of them are singing, so which is nothing but an acoustic signal, because it is sound wave, it is nothing but acoustic symbol. But, how do you from those acoustic signals, how do you recognize the voice of different persons? So, that is what is speaker recognition problem?

Similarly, I can have speech recognition problem. So, you might have heard about tools like speech to word conversion that speech to word conversion tool is nothing but whatever I am speaking, there is a machine which captures the voice and which is converted into electrical signal processes. That electrical signal extracts certain properties, or called features which I will come to a bit later.

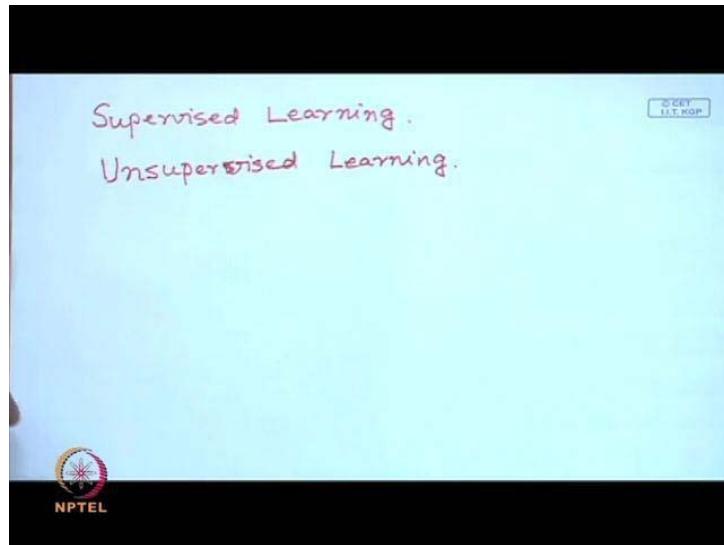
And based on those features the machine tries to extract that which word I have uttered. It is a same concept, which is also used whenever we go for speaker recognition, because whenever I speak or I try to sing a song though I am not a singer. Whenever Hemant Kumar tries to sing a song, there is clear distinction between my voice and the voice of Hemant Kumar, so we can have signal processing techniques to extract those properties or those features from the speech signal.

And using those features, we can try, you should be able to recognize whether the voice is from Hemant Kumar or it is a voice of somebody else. So, similarly, in case of machine mission if I have an automated assembly shop, for we want that every operation has to be done by a robot in an automatic way. So, the robot must be capable of seeing what is there in the shop floor that means that the robot must be attached with a vision system, may be a digital camera. So, digital camera takes images of the shop floor, from those images it locates different objects.

And locating different objects is not sufficient, it should also be able to recognize which object is what and based on that it can pick up certain objects, and picks that object in some specific position. So, there also your pattern recognition or object recognition, because objects are also nothing but patterns. So, pattern recognition or object recognition is very very useful. So, this problem of the domain recognition problems is quite wide, the pattern recognition is applied in various applications, whether we try to

recognize the patterns into one-dimensional signals or we try to recognize patterns in two dimensional signal, like images. Now, when I go for such pattern recognition the pattern recognition can be done in one of the two ways.

(Refer Slide Time: 21:05)



One of the way is called a supervised learning and the other approach is unsupervised, though the problem domains are slightly different. So now, let me just illustrate what is meant by supervised learning.

(Refer Slide Time: 21:50)



Let us take few objects. So, this is a set of objects all of you know, what these are and I have another set of objects something like this. So, if I take the images of these 2 sets of objects they are nothing but patterns. And these patterns are to be known a priori coming, to what Plato said that I had to have a priori knowledge, or a priori information of the object that I have.

And how do I have that a priori information? All of you know that this is nothing but an USB drive sometimes called pen drive, which is used to store data. So, I know that these class of objects or these class of patterns are nothing but USB drives or pen drives which are used for storage of data or storage of program or taking back up from a computer.

Similarly, here I have another set of objects or another set of patterns and I know that these are pens with which I can write. So, these are 2 sets of patterns and the knowledge of patterns. I have a priori and how do I acquire this knowledge? Acquiring this knowledge is not a one-day process, this knowledge has been acquired over the time starting from my childhood when I started writing a, b, c, d, my mother used to take a pen and she used to tell me this you can write and this is how you can write. So, you find that starting from our childhood, we have started learning patterns that is what is a priori knowledge and then I grown up a little bit then I started using these. This USB drives and then there also my teacher or my friends, who have used it before, must have told me that these are USB drives or pen drives with which you can take your data backup or with which you can take a program back up. Then I have started using this USB drives.

So, along with what Plato said that, abstract concepts are known a priori and then what Aristotle said that these concepts that we are learning, the learning must be incremental. That is in the childhood, I have learned about pens when I grown up I learnt about this USB drives. I did not use USB drives when I was child, so I am acquiring this incremental knowledge. So, you can easily see that the pattern recognition problems that we are talking about today that this is not really a recent problem. The problems were have thought of those are analyzed by the philosophers as early as 2005 years ago.

Now, let us come to our problem that when I have these two sets of objects. Now, if I put these objects I know that the similarity of these objects with these sets of objects is more than the similarity of this object with this set of objects. So, as a result I will recognize this object, which is now an unknown object to be a pen not to be a USB drive. So, this is what I mean by supervised learning. So, in case of supervised learning I have to have a priori knowledge and when I get this a priori knowledge, this a priori knowledge is acquired

through experience, through observation, through instruction. It may be many ways and during classification, if I am shown an unknown object then that unknown object has to be put into known classes, which are known a priori based on the similarity measure.

So, when I try to classify this object to a pen effectively, what I am trying to do is, I am trying to find out the similarity between these object and these two sets of objects, and I compute that the similarity of these object with these set of objects is more than the similarity of these object, with these set of objects. So, I recognize this object to be a pen not as a USB drive and this is what is supervised learning.

So in case of supervised learning, I must have a priori knowledge and that a priori knowledge may be acquired through experience, through observation, through instruction, it can be many ways, okay? And I apply that a priori knowledge to recognize or to classify an unknown object or an unknown pattern, so this is what our supervised learning pattern. Now, the other category that I said, which is unsupervised learning, in case of unsupervised learning nothing is told a priori.

So, what I have is, I have a mixture of all these objects right the problem is something like this, I have a mixture of all these different objects. And from this mixture I have to separate the objects into two or more groups. So, what I will do is I will try to find out the objects, which are similar and put them into one group and objects which are not similar to the objects belonging to that group will be put in other groups.

So, over here I will simply put these objects into one group because they appear to be similar, and I will put these objects into another group, because they appear to be similar. So, what I had initially is a set of objects for which I do not have a priori knowledge and what I am doing now is, I find out every pair of objects which are similar and the objects which are similar are put in one group. Objects which are dissimilar are put in different groups.

So, doing that if take any two objects from a group they must be similar. That means, they have high degree of similarity, but if I take an object from one group and another object from another group, they must be dissimilar or the degree of similarity may be very poor.

So, this is the one for which I do not apply any prior knowledge is what is known as unsupervised learning, that is what I said. So, whenever we talk about such pattern recognition problems, the pattern recognition problems are categorized into two different types, one is supervised learning other is unsupervised learning. In case of supervised learning, we have a priori knowledge of pattern or we have a priori knowledge of objects.

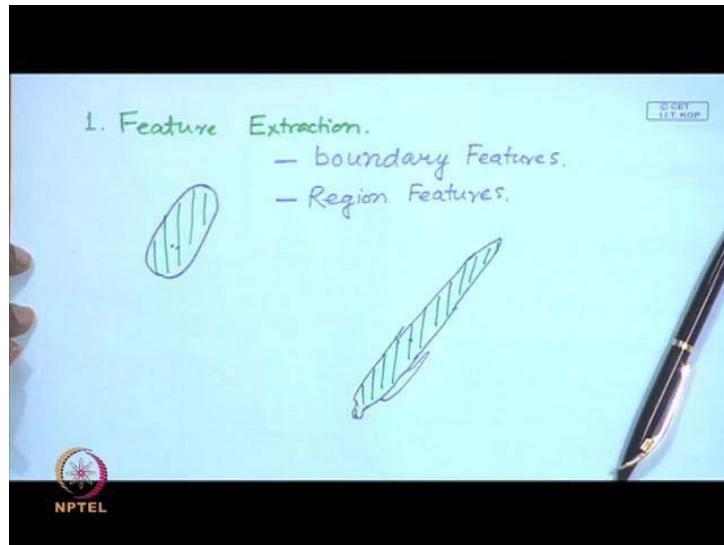
And using that a priori knowledge I want to recognize, or I try to recognize or classify an unknown pattern or an unknown object.

In case of unsupervised learning, I have a mixture of objects, I do not have a priori knowledge, but from that mixture of products what I do is, I partition them into different groups where objects within the same group and similar are similar and objects belonging to different groups are dissimilar. So, I have to have some sort of similarity measure, now again I come back to what Plato said, that the a priori knowledge is embedded in our brain in some mystic way. It is really a mystery that how the information is actually embedded or stored in our brain and how do we retrieve that information to recognize an object.

Now, you think of our purposes that I want to impart the same level of expertise or the same level of intelligence to a computer or to a computing machine. So, that the computer can walk the same way in which, I work or the computer should be able to say that this is one object this is another object. Though as a human being this is pretty obvious to me, I will simply say that, these two are not same, they are different, this is a pen, whereas this a USB drive. But, our challenge is that how do we want or how do we expect the computer to do the same thing, the computer to solve the same problem.

So, all of you who have knowledge of computer programming or writing algorithms, you know that if I want the computer to do the same operation, then I must have some description of the objects or the description of the patterns. So, this object should have some description, these objects should have similar description. And the description or the instance of the description of this object, and the instance of the description of this object they should be different and this is this description what I am calling as feature. So, what are the different types of features that we can have? So, in this course initially we talk about the feature extraction. So, our first topic of this pattern recognition and application lecture series will be feature extraction.

(Refer Slide Time: 32:26)

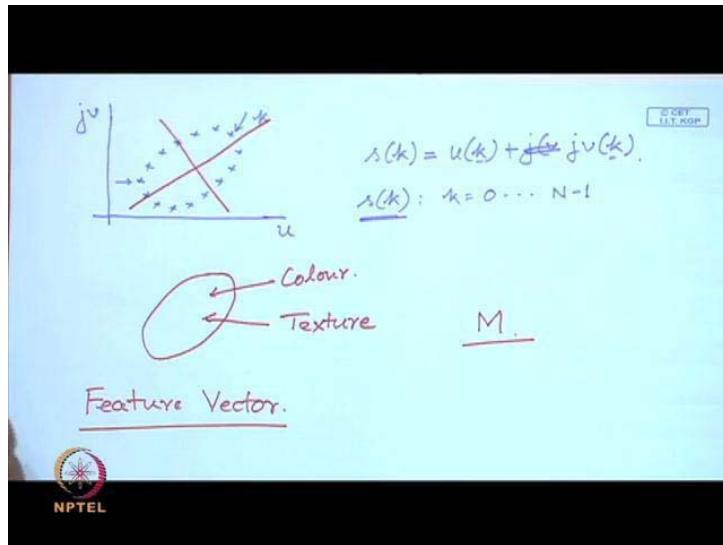


So, what is this feature extraction? Say, for example, if I take this device this USB drive and if I take this pen, you find that if I draw the boundary of these two defined objects. The boundary of this object will be something like this. Similarly, the boundary of this object will be something like this, right? So, you find that there is clear difference in the boundary, so I can have the features, which are derived, which are extracted from the boundary. Not only boundary, both these objects have a region, which is bounded by the boundary. So, this is the bounding region of the USB drive and this is the bounding region of the pen, so I can have features from these bounding regions, also.

So accordingly, when I talk about features, the features are usually of two types, one type of feature is known as boundary features and the other one is known as region features. So, boundary feature is the feature or the set of features, which can be derived, which can be extracted from the boundary of the object. And it has to be argued that whenever it comes to a shape of an object most of the information is actually available on to the boundary.

And whenever we cannot discriminate among different objects or the different patterns using boundary information only, then we go for region features which are obtained from the region enclosed from the boundary. So, what are the boundary based features that we can extract from it? What I can do is, I can have this boundary in the digital domain, is nothing but a set of points.

(Refer Slide Time: 35:57)



So, what I have is, if I draw it in a two dimension or objects are actually three-dimensional, but I can illustrate that with the help of two dimensional figures. So, this boundary is actually nothing but a set of points in a two dimensional space, something like this. So, when I have a set of features like this, I can assume this horizontal axis to be my real axis and the vertical axis to be an imaginary axis. So, given this, every k^{th} point in this boundary can be represented by a complex number say, s_k which is nothing but $s_k = u_k + jv_k$.

And I have all these different points for different values of k , so what I actually have if I start scanning this boundary from any of the initial point. What I have is a set of my complex numbers is s_k , where k varies from 0 to $N - 1$. If I have capital N number of such boundary points, so you find that it is nothing but a sequence of such complex numbers. Then one of the ways in which the feature can be extracted is that I have a sequence of complex numbers. What I simply do is take the Fast Fourier transform, or Discrete Fourier transform of this sequence of complex numbers, when I take the Discrete Fourier transformation.

As I have capital N number of samples, complex samples, I have N number of Discrete Fourier transformed coefficients. And the magnitudes of those N number of Discrete Fourier coefficients itself can describe this particular boundary. So, that becomes a set of feature, so there are various other ways this is just one example, there are various other ways in which the feature can be extracted. One of the ways can be that when I have such a boundary, I may like to find out the moment of this shape around say principle axis. I

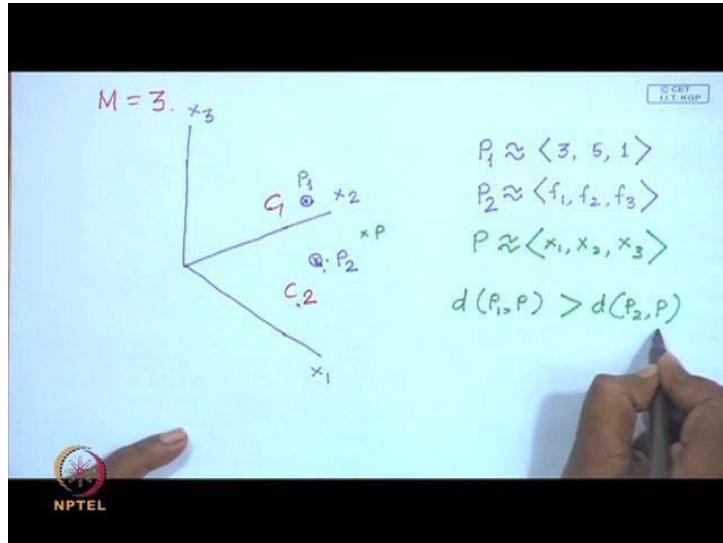
will find out the moment of this shape around this axis, and I find out the moment of this shape around another axis, which is orthogonal to the principle axis.

So, I get two components, so one is moment around the principle axis, the other one is moment around an axis, which is orthogonal to the principle axis and passing towards the center of gravity of the particular shape, so you get another set of feature. This is what is shape feature, so I can have boundary feature, I can have shape feature coming to the regional features, the bounded region will have certain color, it can also have certain texture. So, I can extract the color property which I can call color feature. I can find out; I can extract the texture property which I can call texture feature.

So, this particular object now has different types of properties or set of properties are computed from the boundary, set of properties are computed using the shaping formation like moments, or set of from the color or set of properties from the texture and so on. So, all these different features actually gave me different numerical values and if I put these numerical values in a certain order, each of the numerical values gives me some information of this object. And when all this numerical values are put in a certain order what I get is a vector, right? So, every individual component is feature and all the features take together in a particular order gives me what is called a feature vector.

So, find now that the objects or the patterns are represented by the corresponding vector. So, whenever I have such a vector representation of an object or a pattern my classification becomes very, very simple. How does it become simple? Because, if the components in my feature vector are say capital M, that is every pattern is represented by an M dimensional vector, effectively what I am doing is, I am transforming the pattern or the object from its special domain or time domain to another space, which is a feature space and because this feature vectors are M dimensional. So, what I have defined is an M dimensional feature space, for simplicity, because I will be drawing in the paper.

(Refer Slide Time: 42:09)



If I assume the value of M to be 3, so I assume $M = 3$, that means that I have three-dimensional feature vectors. For every pattern I have one three-dimensional feature vector. Then what is my advantage is? Advantage is, whenever I have such feature vectors of say dimension three, what I am effectively doing is I am representing my pattern in a three-dimensional space, which is nothing but feature space. So, this is my feature 1, let me call it x_1 , this is the dimension of x_2 , this is the dimension x_3 .

So, if pattern P_1 will be represented by a vector, say something like this $P_1 \approx \langle 3, 5, 1 \rangle$, which is the feature vector corresponding to pattern P_1 and the moment I have this three-dimensional feature vector. This three-dimensional feature vector is nothing but a point in my three-dimensional space which may be says somewhere here, so this is the feature vector corresponding to my pattern P_1 .

Similarly, I can have a point P_2 , which is again represented by a feature vector say $P_2 \approx \langle f_1, f_2, f_3 \rangle$, but this f_1 will have certain numerical value f_2 will have certain numerical value, f_3 will also have certain numerical value. And this pattern P_2 once it is represented by such a feature vector, this pattern P_2 will also be a point in my three-dimensional feature space. Now, you look the advantages that once these patterns are represented in the three-dimensional feature space, I can measure the similarity or dissimilarity between these two patterns P_1 and P_2 .

Simply by taking the distance between these two feature factors, if the patterns P_1 and P_2 , they are identical. That means, this P_2 and this point P_1 , they should be coincident that if f_1 should be equal to 3, f_2 should be equal to 5 and f_3 should be equal to 1. In other words, the distance between P_1 and P_2 should be equal to 0, if P_1 and P_2 are same pattern.

If P_1 and P_2 they are similar pattern they may not be exactly identical in that case. The distance between the corresponding features of vectors, when I compute the distance should be very small. If P_1 and P_2 they are widely different, there is no similarity between P_1 and P_2 , in that case the distance between the corresponding feature vectors in the three-dimensional feature space will be very large.

So, if the distance between the feature vectors is small or negligible, we can say that those patterns are similar or they are the same patterns, slight difference in the feature values may be due to measurement error, may be due to fabrication error, but the patterns are same if there is no error, then the feature of vectors must be identical. So, I have different sources of error, I can have measurement error, I can have fabrication error. All this different types of errors may come in which does not make the features of vectors identical, but the feature of vector should be very close that is the distance between the feature of vectors are very small.

So, I can have a measure of similarity, I can decide whether the patterns are similar or the patterns are not similar from the distance between the corresponding feature of vectors. If the distance is very small then they are similar, if the distance is large then they are not similar, so this is my simple rule. Now, let us assume that this point P_1 and this point P_2 , they are patterns of two different classes, they are patterns of two different classes and these two patterns are known as priori. So, this point P_1 may come from a class of C_1 and this pattern C_2 may come from class C_2 .

Now, my job will be that given an unknown pattern say P , but this P is also represented by a similar such feature vector something like this say $P \approx \langle x_1, x_2, x_3 \rangle$. So, my job is that I have this a priori in P_1 and have this a priori in P_2 . I know P_1 belongs to C_1 , P_2 belongs to C_2 . So, my job is that given now unknown pattern P , I have to recognize this pattern or I have to classify this pattern.

So, one simple classification rule can be that, because I have these two known patterns P_1 and P_2 . I find out the distance from P_1 to P , I find out the distance from P_2 to P , so what I have to do is, I have to find out distance between P_1 and P and I have to find out distance from P_2 to P . So, if I find that the distance from P_1 to P is greater than distance from P_2 to P that means

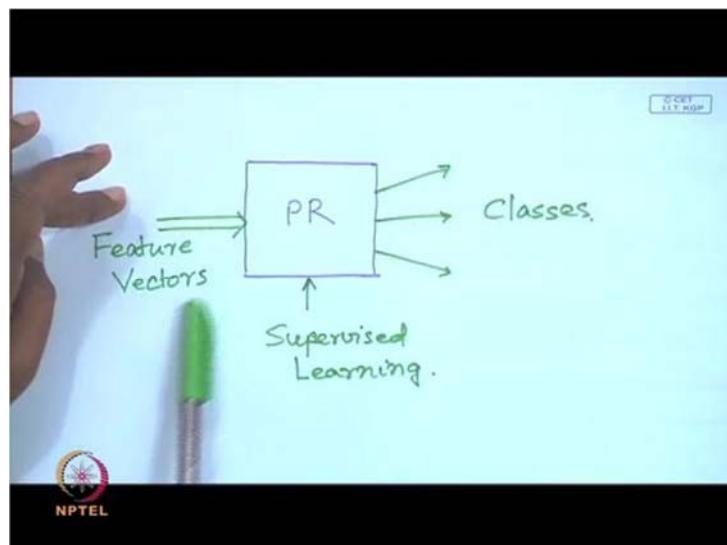
P_1 this unknown pattern P is more similar to P_2 than to P_1 , because the distance between P_1 and o is more than the distance between P_2 and P . So, this unknown pattern P is more similar to P_2 than it is similar to P_1 . So, my simple classification rule can be that I will classify or I will recognize this unknown pattern to be a pattern belonging to class C_2 .

It will not be pattern belonging to class C_1 , so during our course of lectures, initial few lectures we will devote for different types of features and different types of features extraction techniques. And then next, we will talk about different types of classification problems. Now, when I, the moment I have come for the classification, we said that we want to represent every pattern by a corresponding feature vector.

So, once a pattern is represented by a feature vector, whether it is temporal pattern like speech signal or the signal coming out of any other source, or it is a spatial signal like image or video. Video is of course, a spatiotemporal signal, because I have different number of frames, the frames are played at a certain rate, 30 frames per second and 25 frames per second and so on. So, images are spatial signal, speech signals are totally temporal signals, videos are spatiotemporal signals.

I have special variation, variation along the space I have also variation along the time, so those are actually spatiotemporal signals. Now, whatever type of signals or whatever type of patterns I have once, I convert the pattern to a feature vector. My classification remains the same, because what I have to do is depending upon my application domain. I have to train my classifier so that it can work in that particular application domain. So what I can do is, I can put this pattern recognition problem in a block diagram, something like this.

(Refer Slide Time: 51:09)



So, I have pattern recognition or a pattern recognizer, let me call it as PR, input to this pattern recognition system, will be my feature vectors. So, these are feature vectors and output of the pattern recognition system will be the decision about different classes of this input feature vector. When I have this pattern recognition system, this pattern recognition system must be trained or learned by a supervised learning approach.

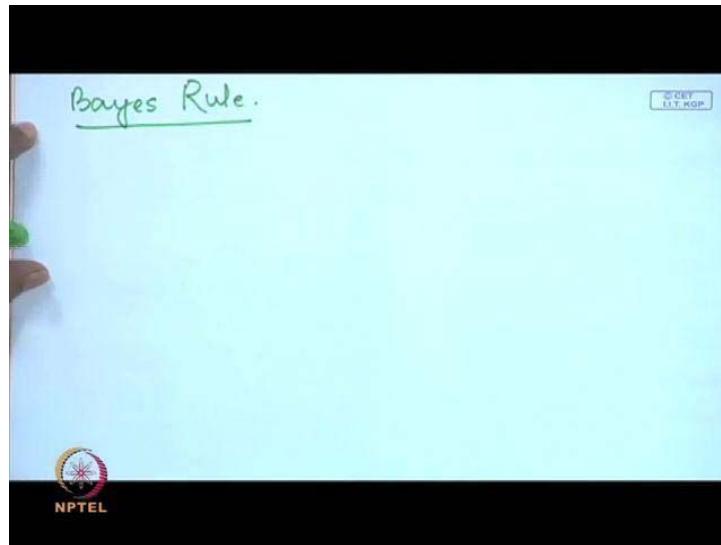
So, for supervised learning what I have to do is, I have to input a set of feature vectors or different sets of feature vectors. For different sets, I must know what class, from which class that particular set of feature vector has come, so that is what is supervised and based on that information I have to train this pattern recognition system. Once the pattern recognition system is trained then given an unknown feature vector. I should be able to say, or this pattern recognition system must be able to recognize, to which class this feature of vector should be classified or it should belong.

Now, given this sort of a block diagram. You find that this sort of pattern recognition system can now be applied in any domain. It can be used for speech recognition, it can be used for speaker recognition, it can be used for object recognition, it can be used for image classification. Now, what I mean by image classification, I can have images of different types; I can have indoor images, I can have images of natural scene, I can have images of a shop floor. If my pattern recognition is properly trained, after properly finding out the feature vectors, then this pattern recognition system can be used to recognize or to classify patterns from any domain, right?

Of course, the pattern recognition system for speech recognition is different from the pattern recognition system for object recognition because that domain knowledge is different. My representation of the feature vectors will be different. So, in this particular course what we will talk about is, we initially we talked about the different techniques to extract the feature of vectors.

What are different types of feature of vectors that can be used, what are different techniques to extract the feature of vectors. And then we will go for different pattern recognition systems or different types of classifiers. Some of the classifiers make use of the statistical properties of the signals or the feature vectors are used using statistical property.

(Refer Slide Time: 54:55)



Then we can have probabilistic models of different classes, making use of mean and variants of the feature vectors, which we will talk about, when we talk about something called Bayes rule, which is used in statistical classification or statistical classifier. So, there again we can have two different forms of classifier one form of classifier is called a parametric classification technique. Another form classification is called a non-parametric classification technique. So, in case of parametric classification, whenever I have a statistical model or probabilistic model of a particular class, we will assume that the probability density function will have a parametric form.

In the sense say for example, a Gaussian probability density function that has two different parameters, one is mean other one is variance. When I have feature vectors of multiple dimensions, I will have mean vector and covariance matrix so which are the parameters of probability density function. If I assume the probability density function to be Gaussian, but in most of the cases it may so happen that the probability density function, which is exhibited by the sample, they do not follow any particular parametric form.

So, I will have different types of non-parametric classification techniques, so you will have parametric classification techniques. We will have different types of non-parametric classification techniques, then we will also have other types of classification techniques. Say for example, neural network many of you might have heard of it neural network is an attempt to build machines which can imitate the functioning of our human brain. So, we will also

discuss about different types of neural networks, which can be used for pattern recognition or pattern classification.

We will talk about another type of pattern recognition or pattern classification, which is called hyper box classifier, then we will also try to combine hyper box classifier with the fuzzy measure. Later on we will combine our hyper box classifier, fuzzy measure and neural network to have certain model called funny mean max neural network. So, that is another type of tools fuzzy mean max neural network that can also be used for pattern recognition or pattern classification.

We will also talk about something called support vector machine, so what does support machine vector does is, it defines a plane in my feature space. So, using that plane I can divide the space into two half spaces. So, in one of the half space I will have patterns belonging to one class, in the other half space, we will have patterns belonging to some other class.

So, the support vector machine actually tells you that given a set of training features of the training feature vectors, where to put this plane or hyper plane, or how to sub divide my feature space into half spaces so that my classification, when I go for classification I can attain minimum error of classification. So, all these different types of techniques for pattern classification we will talk about. Then later on we will also talk about classification of temporal patterns.

Say, for example, if I simply put my hand like this or my palm like this it's just a palm, but it is not sufficient. But, if I wave like this, this has certain meaning, if I put it like this it has some other meaning. So, simply palm, showing a palm is not sufficient. It is also important that how the palm moves over time, because that can state a meaning so it is a temporal pattern. So, you will also talk about how to recognize or how to classify such temporal patterns. So, these are the different techniques that we will be talking about during this course, on pattern recognition and applications. So, in the next class we shall start talking about the feature extraction techniques.

Thank you.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 10
Maximum Likelihood Estimation

Good morning. So, today we are going to discuss about the parameter estimation of the probability density functions, and the particular type of estimation technique that we will discuss is called maximum likelihood estimation. So, till now what we have discussed is, we have talked about the discriminant functions for different classes and we have seen that, the type of discriminant function which is very popular is nothing but the logarithmic function of the probability density function. So, when we take any particular probability density function, the probability density function is specified by a number of parameters.

So, in particular what we have taken is the normal or Gaussian distribution and in case of normal or Gaussian distribution we have said that, there are two parameters which uniquely identifies this PDF. One of them is the mean vector and the other one is the covariant matrix. So, these are the parameters that is mean vector and covariance matrix say this is given, you know what is the nature or structure of the probability density function, which is involved in that case.

And so far what we have discussed is, we have taken the logarithmic of the probability density function or a combination of it, combined with the a priori probability of different classes to give us the discriminant function for different classes. And one you have taken the decision boundary between two different classes, and then we have seen that it is, nothing but the difference of the discriminant function.

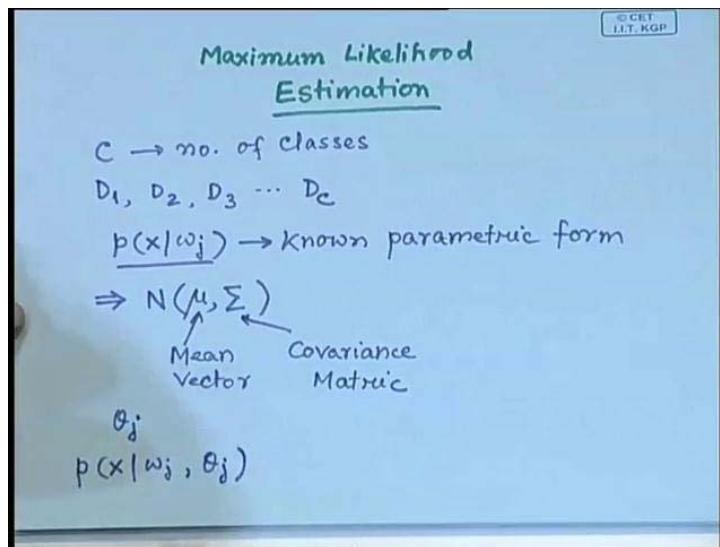
And we have also seen that, depending upon the nature of this parameter vectors or the nature of the mean vector and the covariance matrix, we can have different types of boundary, the decision boundary between two different classes. You can have linear boundary; we can have a linear boundary, which is orthogonal to the line joining the mean vectors. We have got linear boundary, but which is not orthogonal to a line between two mean vectors and we have also seen that, we can also have in general

the decision surfaces which are, nothing but quadrics in d dimension. Where d is the number of components in the feature of vector which, we use for classification.

So, effectively when we design such a kind of classifier or we want to find out the decision surface between two different classes, the nature of the probability density function is very, very important. And as such, as the nature of the probability density function is defined by the parameters mu and sigma that is, mean vector and covariance matrix. We have to accurately estimate, the mean vector and covariance matrix given, a Gaussian distribution, we know the form of mean and covariance matrix. But if the probability density functions is other than that, other than the Gaussian or normal distribution, in that case we have to find out what is the parameter vector that identifies the probability density function.

So, today what we will be discussing about is a particular technique for parameter vector estimation, for a known parametric form of the probability density function. So, the type of estimation techniques that we will discuss about is the maximum likelihood estimation. So, the problem is something like this.

(Refer Slide Time: 04:11)



Suppose we have got c number of classes so you have c number of classes and we are assuming that, the kind of classification, or the kind of learning that we are going to employ is supervised learning. That means for each of the classes, we have set of samples for which I know that, to which of the classes this set of sample belongs.

So, as we are considering that we have c number of classes so naturally we assume that we have c sets of feature vectors and let us represent that, c set of feature vectors as feature vectors D_1, D_2, D_3 up to D_c , as we have c number of classes. So, you can assume that the set of samples in each of the classes is something like this, that these set of samples are drawn independently according to the probability law $P\left(\frac{x}{\omega_j}\right)$. So, the samples belonging to class

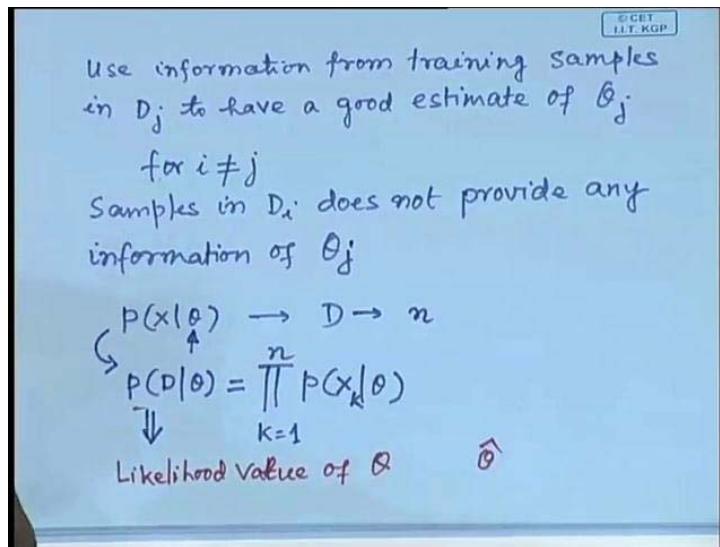
D_j or belonging to set D_j within these set of samples is drawn independently according to the probability law $P\left(\frac{x}{\omega_j}\right)$. And we also assume that, this $P\left(\frac{x}{\omega_j}\right)$ has a known parametric form.

So, in particular this known parametric form if it is a normal distribution or a Gaussian distribution then we have said that, this known parametric form is uniquely identified, is uniquely defined by the parameters, which are μ and Σ . Where, μ is the mean vector and Σ is the covariance matrix. So, these are the parameters which are, known for normal distribution or Gaussian distribution.

Similarly, I can have other types of distributions which are also parametric, but the parameters may be different from this. So, what I assumed is that the samples in set D_j is drawn, are drawn independently according to the probability law $P\left(\frac{x}{\omega_j}\right)$, where this $P\left(\frac{x}{\omega_j}\right)$ has a known parametric form. So, it has a known parametric form so in general I can assume that for class ω_j , the parameter vector is a vector given by θ_j .

So, this is my parametric probability distribution function, probability density function which has a known parametric form where the parameter is given by, this parameter vector θ_j . So, to write explicitly the dependence of this PDF on this parameter vector θ_j , I can write this as $P\left(\frac{x}{\omega_j}, \theta_j\right)$. So, this specifies that the probability density function has the parameter vector θ_j . So, our problem in this case is that we have to use the information available from the samples in the set D_j , to estimate this parameter vector θ_j . And the kind of technique that we are going to use, we are going to discuss for this estimation of parameter vector θ_j is, what is known as maximum likelihood estimate.

(Refer Slide Time: 08:49)



So, what you have to do is, we have to use information from training samples in set D_j , to have a good estimate of the parameter vector θ_j . So while doing so we also assume that the samples in set D_i does not provide any information about the parameter vector θ_j , for $i \neq j$. So, for $i \neq j$ samples in D_i does not provide any information of the parameter vector θ_j whenever, $i \neq j$ that means, the classes are or the samples belonging to different classes are unique independent. So, samples from one class do not provide any information of the parameter vector, of the probability density function of another class.

And as we have also assume that, the samples are drawn independently so I can make use of these individual samples to, estimate our probability density function, and because the samples in D_i , does not provide any information about the parameter vector θ_j . So, effectively what we have is, we have c number of independent problems so given a set of samples I have to estimate the parameter vector of the probability density function, which best estimates the data distribution within that particular set. So, I can simply get away with this subscripts i and j and all that, I can simply write $P(x/\theta)$.

Because, I have independent c number of problems, I have a set of samples for that given set of samples, I have to estimate the parameter vector θ of the probability density function, which best describes the data distribution of that particular set. So, instead of using this subscripts I can simply write that, I have to estimate $P(x/\theta)$ where I have

been given a set of samples. So, set of samples D and using the information available in these samples, I have to estimate this particular parameter vector θ . And as the samples are drawn independently and if I assume that D contains n number of samples.

Student: Sir, what is $P(x/\theta)$?

See, what I have said is $P(x/\omega_j, \theta_j)$, what is the interpretation of this, this is the probability density function of the set of samples in set D_j . I know that, the samples in set D_j actually belong to class ω_j and θ_j is the parameter vector which specifies this probability density function. And because of our assumption that, the samples in D_i , does not provide any information of θ_j . I have c number of such sets of samples, D_1 to D_c as I am considering that have c number of classes and each of these set of samples, independently provide me the information of the parameter vector, of the corresponding class. D_1 does not give me any information about θ_2 similarly, D_2 does not give me any information about θ_1 .

So, effectively what I have is, I have c number of independent problems given a set of samples D, I want to estimate the parameter vector θ , of the probability density function which best describes the data distribution in set D. So, because the samples in one set is not giving me information about the parameter vector, of the other set corresponding to the other set. So, I can simply remove this subscript j or even I can remove this, class conditional probability ω_j because what I have is, given a set of samples, I want to estimate the corresponding θ .

Student: Sir basically that are some descriptor of the.

It is the parameter vector that describes the probability density function.

Student: Which can be like weight Sir? weight

No, θ_j simply tells the what is the shape of the probability density function like in case of Gaussian distribution or normal distribution, θ represents the components of μ and the covariance matrix Σ . Is that okay? Similarly, for other probability density function the components of θ will be something else. Is that clear?

So, what I have is, I have a problem that, I have set of samples D, I am not bothered about which set is or from which class the samples have been drawn. I simply have a set of samples

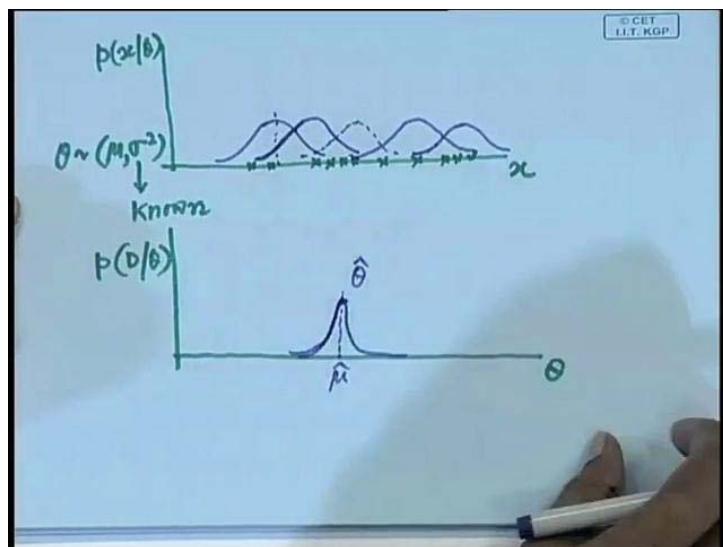
D , whose distribution is actually specified by the parameter vector θ . So, instead of this form the $P(x_j | \omega_i, \theta_j)$ because now ω_j is no more important to me because I have separate, separate problems. So, I am simply rewriting this in the form $P(x | \theta)$, this θ specifies, what is the nature or shape of the probability density function.

So, given this if I assume that set D , the set contains n number of samples which are drawn independently. So, because they are drawn independently so I can also write from here what I have to do is, I have to make use of the information available in the set of samples to, estimate the value of θ . So, I can write this because there are n number of samples I can write that, $P(D | \theta)$ in another form, θ specifies the parameter vector. So, I can write it in this form

$P(x | \theta)$, product of this, let me put a subscript $P(x_k | \theta)$, k varying from 1 to n . Because, I have n number of samples in this set D and the samples are drawn independently.

So, this can be written in the product form and this term $P(D | \theta)$, this is what is known as likelihood value of θ . And the maximum likelihood estimate of θ is, the value set θ , which maximizes this function. So when this function will be maximized and actual the set of samples which are drawn and the parameter vector θ , the estimate of the parameter vector which, best describes the probability density function of the distribution of the set of the samples x_k . So, that time this value will be maximum and the value $\hat{\theta}$, which maximizes this likelihood value of θ , that is called maximum likelihood estimate of this parameter vector θ .

(Refer Slide Time: 18:58)



Now to be more specific, let us have a diagram, suppose I consider univariate case. This is x and on this side I plot $P(x/\theta)$ and assume that because this is univariate case. I assume that, this $\theta \sim (\mu, \sigma^2)$, where μ is the mean and σ^2 is the variance. And suppose I have point distributions something like this, some arbitrary distribution something like this and this θ , the parameter vector is nothing but $\theta \sim (\mu, \sigma^2)$. Where μ is the mean and σ^2 is a variance.

Now out of this, if I assume that this sigma square is known, I am just simplifying the case. If I assume, that sigma square is known then what is unknown is μ so from these point distribution, I have to estimate the value of μ . So, you find that for different values of μ , I can have different types of probability density functions like this, $P(x/\theta)$. When I assume that, the value of μ is this for some other value of μ , the nature of probability density function will remain the same, but it will shift. Now, why this will remain the same because I have assumed that sigma square is known.

So, I can have this different types of distribution functions, where the variance value is same, but the mean value is different. Is that okay? Now, you find that all these different distributions they do not really, all of them does not really represent the distribution of this data set. Only one of them will faithfully represent the distribution of this dataset. So, for that this likelihood estimate, the likelihood value that $P(D/\theta)$ is very important.

So, if I plot this $P(D/\theta)$ you find that for different values of x this $P(x_k/\theta)$, where θ is nothing but this μ we have different value. And here for the likelihood value of θ , what I am taking is, I am taking product of these values. So, if I plot similarly, now I plot with respect to θ , $P(D/\theta)$. You find, you will find that this plot will be something like this and this will have a peak at $\theta = \hat{\mu}$.

And in this case, this will have a peak because θ is nothing but equal to μ assuming that, sigma square is known. So, it will have a peak at $\theta = \hat{\mu}$ so this is what is the maximum likelihood estimate of this $P(x/\theta)$, which best represents the mean value of this data distribution. Is that okay? so I have to find out this position of the mean, $\hat{\mu}$ or in this case θ .

(Refer Slide Time: 23:30)

$$P(D|\theta) = \prod_{k=1}^n P(x_k|\theta)$$
$$l(\theta) = \ln P(D|\theta)$$
$$= \sum_{k=1}^n \ln P(x_k|\theta)$$

So, this maximum, the likelihood value of θ as we said is represented by $P(D/\theta)$, which is

equal to $\prod_{k=1}^n P(x_k/\theta)$, take the product $k = 1$ to n , where n is the number of samples present

in the set D . And you have also said earlier that, instead of taking this likelihood value, as we have said in case of our, that discriminating function, that instead of simply taking the probability density function as the discriminating function if you take the logarithmic of it, that helps us in analysis.

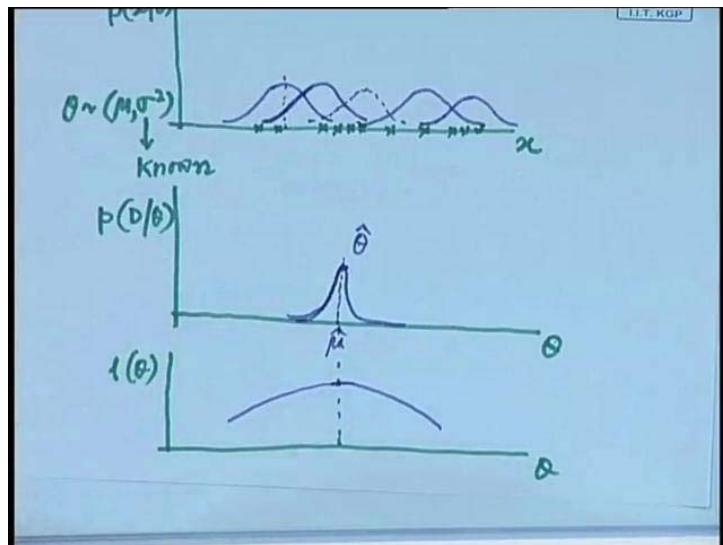
So, similarly here you find that this likelihood value is a product term so instead of taking this likelihood value directly, if we take the logarithm of this likelihood value, that is what is the log likelihood. So, I can define the log likelihood as $l(\theta)$ which is nothing but $\ln P(D/\theta)$. And which will simply be $P(D/\theta)$ is of this form $\prod_{k=1}^n P(x_k/\theta)$, which is a product term, when I take the logarithm it will become a summation term.

So, this will become further to $\ln \prod_{k=1}^n P(x_k/\theta)$, where k value is from 1 to n . So, this is the

log likelihood, this is the likelihood value of θ , this is the log likelihood value of θ . And because we are trying to estimate the value of θ I simply represent it, represented these as $l(\theta)$. Assuming this log likelihood is a function of θ , which we are trying to maximize. And

coming to this particular curve, instead of likelihood if I put log likelihood the position of the maximum will remain the same, but the curve will be smooth.

(Refer Slide Time: 26:01)



So, if I plot $I(\theta)$ versus θ , the position of the maximum will remain the same, but the nature of the curve will be something like this. So, it will be a smooth curve while this is a sharp curve and here you find that, as we have more and more number of points this curve will be narrower. As we have more number of points, this curve will be broader so that clearly says that if I have a narrower curve, I can estimate the maximum more accurately. If it becomes a broader curve then, the maximum estimation may not be that accurate because around maximum I have a flat curve.

Over here or over here if it becomes a broad one, around maximum I will have a flat curve. So, the maximum estimation is not that accurate and if I have more number of points more and more number of samples, this curve will be more and more narrow where the confidence in estimating the maximum is much more. If I have very less number of points then, the confidence in estimating the maximum is less so that is what it means.

Student: Sir will this data had been changes or will remain same.

Pardon.

Student: This data had changes or it remains the.

That is what I said, that depends upon the number of values that I have. Our confidence in estimating θ , depends upon the number of samples I have. If I have more number of samples, my estimation of θ is more accurate, if I have less number of samples the accuracy is not that high.

So, this is what is log likelihood whether I use the likelihood value or the log likelihood value, the estimate of θ it will also, it will always give me the maximum likelihood estimate. Because, the logarithm is a monotonic function now to maximize this, we know the formula of our school level mathematics, that we have to use the concept of differential calculus. So, what I have to do is I have to differentiate this $l(\theta)$ and equate that differential to 0. Because, here θ is a vector so instead of simple differentiation I have to take the gradient.

(Refer Slide Time: 28:37)

The image shows handwritten mathematical notes on a whiteboard:

$$P(D|\theta) = \prod_{k=1}^n P(x_k|\theta)$$

$$l(\theta) = \ln P(D|\theta)$$

$$= \sum_{k=1}^n \ln P(x_k|\theta)$$

$$\theta = (\theta_1, \theta_2, \dots, \theta_p)^t$$

$$\nabla_{\theta} = \begin{bmatrix} \frac{\partial}{\partial \theta_1} \\ \frac{\partial}{\partial \theta_2} \\ \vdots \\ \frac{\partial}{\partial \theta_p} \end{bmatrix}$$

So, I have to take ∇_{θ} with respect to θ and if this θ has got say, p number of components. So, if θ has got p number of components which are defined as θ_1, θ_2 up to say θ_p . I put it as transpose because θ is a vector so if it has got this p number of components then, ∇_{θ} operator

will be simply $\nabla_{\theta} = \begin{bmatrix} \frac{\partial}{\partial \theta_1} \\ \frac{\partial}{\partial \theta_2} \\ \vdots \\ \frac{\partial}{\partial \theta_p} \end{bmatrix}$, this is the gradient operator. So, what I have to do is, I have to find out the gradient of the likelihood value of θ 's.

(Refer Slide Time: 29:43)

$$\nabla_{\theta} l(\theta) = 0$$

Gaussian $\rightarrow (\mu, \sigma^2)$
 μ known.

$$\ln p(x_k|\theta) = -\frac{1}{2} [\ln(2\pi) + \frac{|Σ|}{2}] - \frac{1}{2} (x_k - \theta)^T \Sigma^{-1} (x_k - \theta)$$

$$\nabla_{\theta} \ln p(x_k|\theta) = \Sigma^{-1} (x_k - \theta)$$

That is, I have to find out $\nabla_{\theta} l(\theta)$ where the gradient has to be with respect to parameter vector θ and I have to equate this to 0. So, when $\nabla_{\theta} l(\theta) = 0$, I get a number of equations, simultaneous equations. I have to solve that set of simultaneous equations because this gradient has got p number of components because the gradient will have p number of components. And I equate this gradient to 0 that means, I will get p number of simultaneous equations and when I solve this p number of simultaneous equations, I get the components of the parameter vector θ . That is simply what you have done in our school level mathematics.

Now here the problem is, if I equate the gradient to 0 to get a number of simultaneous equations, I may lead to the actual mean or the global maximum or I may lead to local

maximum not only that, simply equating gradient to 0 also may give me the minimum. So, again you know that if I take the second derivative, that is gradient of the gradient, the second derivative will be negative if it is the maximum, the second derivative will be positive if it is a minimum.

So that, where I can filter out which ones represent the maximum values and which ones represent the minimum values. I am interested only in the maximum values so I get a set of values of θ which represent, the maximum. Then for each of them I have to actually find out what is the functional value, to identify what is the global maximum. So, that is what I have to follow to find out the maximum likelihood estimate of a parameter vector.

Now let us see that, this one really gives us what we want that is, the validation of our procedure. To validate this, let us take some known cases and let us see that, for those known cases whether I get, really what I want. So, for validation I will take the same Gaussian case or the normal distribution, for which I know that the parameter vector which is given by μ . And for simplification, instead of talking multivariate Gaussian, let me take the single variate case. So, the parameter vector is actually given by (μ, σ^2) and initially for simplicity, let me assume that σ^2 is known. So, what I have to find out is the mean μ .

So, let us see whether by using this maximum likelihood estimate procedure, I really get the mean value or not, given a normal distribution with mean μ and a known variance σ^2 . So, for this Gaussian distribution we know that, the log likelihood will be given by $\ln P\left(\frac{x_k}{\theta}\right)$ is nothing but $-\frac{1}{2}[\ln(2\pi).|\Sigma|] - \frac{1}{2}(x_k - \theta)^t \Sigma^{-1} (x_k - \theta)$ so this is my logarithm of the probability density function. So, what I want to do is, I want to differentiate this with respect to θ and in all case $\theta = \mu$.

So, if I take $\nabla_{\theta} \ln P\left(\frac{x_k}{\theta}\right)$, this will be simply, you find that here this term is constant and our variance σ^2 is known. So, this term is also known so if I differentiate this, this term will be equal to 0. So, what I have to do is, I have to simply take the differentiation of this with respect to θ , which is a function of θ . So, if I differentiate this, it will simply become $\nabla_{\theta} \ln P\left(\frac{x_k}{\theta}\right) = \Sigma^{-1} (x_k - \theta)$. And you find that, in the log likelihood I have to take the

summation of this, for all values, for all the samples that is $k = 1$ to n and I have to equate that to 0.

(Refer Slide Time: 35:40)

$$\begin{aligned}
 & \sum_{k=1}^n \Sigma^{-1} (x_k - \theta) = 0 \\
 \Rightarrow & \sum_{k=1}^n (x_k - \theta) = 0 \\
 \Rightarrow & \sum_{k=1}^n \theta = \sum_{k=1}^n x_k \\
 \Rightarrow & n \cdot \theta = \sum_{k=1}^n x_k \\
 \Rightarrow & \hat{\theta} = \frac{1}{n} \sum_{k=1}^n x_k = \mu
 \end{aligned}$$

So, effectively what I have to have is, I have to equate sum of so let me put this as $\sum_{k=1}^n \Sigma^{-1} (x_k - \theta) = 0$, where k will vary from 1 to n , this should be equal to 0. So, from here

you have find that, if I pre multiply both sides by Σ that is covariance matrix then, what I simply get is $\sum_{k=1}^n (x_k - \theta) = 0$, where k varies from 1 to n , this will be equal to 0. So, this

simply gives us that $\sum_{k=1}^n \theta = \sum_{k=1}^n x_k$, where here also k varies from 1 to n , here also k varies from 1 to n . What this means is, I have to sum θ n number of times so this is simply n multiplied by θ because it is the same θ which has been added n number of times.

That is equal to sum of x_k , k varying from 1 to n , so this simply gives us the parameter θ , which is nothing but the maximum likelihood estimate $\hat{\theta} = \frac{1}{n} \sum_{k=1}^n x_k$, k varying from 1 to n

which is nothing but the mean μ . So, we started with this assuming the mean is unknown, we have followed the procedure of maximum likelihood estimate of the parameters. And we have really landed to, the unknown parameter which is the arithmetic mean of the set of samples that I have, which is nothing but μ . Now, let us take the more general case assuming

that, neither we know mu nor we know sigma square, that is both mean and variance both of them are unknown.

(Refer Slide Time: 38:10)

$$\theta \approx (\mu, \sigma^2)$$

$$\theta_1 = \mu$$

$$\theta_2 = \sigma^2$$

$$\ln P(x_k | \theta) = -\frac{1}{2} \ln 2\pi \theta_2 - \frac{1}{2\theta_2} (x_k - \theta_1)^2$$

$$\nabla_{\theta} \ln P(x_k | \theta) = \left[\begin{array}{l} \frac{1}{\theta_2} (x_k - \theta_1) \\ -\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2} \end{array} \right]$$

So, over here our parameter vector θ is nothing but μ and σ^2 , it has two components, for simplicity I am concentrating the univariate case, for multivariate case the number of operations will be more, that is all, but the procedure will remain the same. So, here this parameter vector θ has got two components, one is μ and other one is σ^2 so assume that the first component θ_1 is μ and the second component θ_2 is σ^2 . Is that okay?

Now over here, the same $\ln P\left(\frac{x_k}{\theta}\right)$ is nothing but $-\frac{1}{2} \ln(2\pi)\theta_2 - \frac{1}{2\theta_2} (x_k - \theta)^2$. Because, it is

$\exp\left[-\frac{1}{2}\left(\frac{X-\mu_i}{\sigma}\right)^2\right]$ so it simply becomes $-\frac{1}{2\theta_2} (x_k - \theta)^2$, in the logarithmic form. So, again if I

take the derivative of this with respect to θ so $\nabla_{\theta} \ln P\left(\frac{x_k}{\theta}\right)$. So, it is nothing but ∇_{θ} , sorry I am not yet come to \mathbf{l} , because \mathbf{l} contains the summation term.

So, let me put it as $\nabla_{\theta} \ln P\left(\frac{x_k}{\theta}\right)$ so this is nothing but I have two components, one is derivative of this with respect to θ_1 , other one is derivative of this with respect to θ_2 . So, when I take the derivative of this with respect to θ_1 , you find that, this, the first component is independent of θ_1 , this will be equal to 0. So, what I have is the derivative of this, with

respect to θ_1 and when I take the derivative of this with respect to θ_1 , it simply becomes

$$\frac{1}{\theta_2} (x_k - \theta_1) .$$

And when I take the derivative of this, with respect to θ_2 , you find that the derivatives corresponding to both the terms will exist because in both the places I have θ_1 . And that will

simply become, $-\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2}$. So, I took the summation of this and summation of

this over, all the samples that is where k varying from 0 to n and equate that to 0, to give me the maximum likelihood estimate.

(Refer Slide Time: 41:58)

The image shows a handwritten derivation on a light blue background. At the top, there is a small logo in the top right corner that says "© GET IIT RGP". The derivation starts with two equations:

$$\sum_{k=1}^n \frac{1}{\theta_2} (x_k - \hat{\theta}_1) = 0$$

$$-\sum \frac{1}{2\theta_2} + \sum \frac{(x_k - \hat{\theta}_1)^2}{2\theta_2^2} = 0$$

Below these, an arrow points to the first equation, which is then solved for $\hat{\theta}_1$:

$$\Rightarrow \hat{\theta}_1 = \frac{1}{n} \sum_{k=1}^n x_k = \mu$$

Finally, the second equation is solved for $\hat{\theta}_2$:

$$\hat{\theta}_2 = \frac{1}{n} \sum_{k=1}^n (x_k - \hat{\theta}_1)^2 = \sigma^2$$

So, what I have is $\sum_{k=1}^n \frac{1}{\theta_2} (x_k - \bar{\theta}_1) = 0$, k varying from 1 to n that should be equal to 0, one

equation. And the second equation will be $-\sum_{k=1}^n \frac{1}{2\theta_2} + \sum_{k=1}^n \frac{(x_k - \bar{\theta}_1)^2}{2\theta_2^2} = 0$. So, I have to solve

these two simultaneous equations and when you solve these two simultaneous equations, you

will find that you will get answer of the form $\bar{\theta}_1$ is nothing but $\frac{1}{n} \sum_{k=1}^n x_k$, k varying from 1 to

n. And $\bar{\theta}_2$ will be nothing but $\frac{1}{n} \sum_{k=1}^n (x_k - \bar{\theta}_1)^2$, k varies from 1 to n. So, simply by solving these two equations I will get these two solutions.

Where you find that, this is nothing but μ and this is nothing but σ^2 . So, that validates that when I follow this maximum likelihood estimate then, I really get the parameter vectors, which truly represent the set of samples, that I have then of course, as I said that regarding the accuracy of this parameter vectors that I have, I have to depend upon the number of samples. The more the number of samples I have, more accurate our estimation will be, if I have less number of samples our estimation will be less accurate.

So, as far accuracy I have to depend upon the number of samples, I have to have more number of samples, but in most practical cases I do not get sufficient number of samples. So, as we cannot get sufficient number of samples our parameter estimation may not be sufficient, may not be accurate not only that, what kind of distribution it really follows, that is also difficult to predict.

I mean whether, it follows Gaussian distribution or it follows exponential distribution, what kind of distribution it follows that is also sometimes difficult to predict because we have insufficient number of samples. So, in such cases instead of trying for parametric probability distribution, we have a method called a non-parametric estimation. We will discuss that later on, but before that let me just take another example to illustrate how this parameter estimation can be done for arbitrary probability estimation, probability density function.

(Refer Slide Time: 45:41)

Example

$$p(x|\theta) = \begin{cases} \theta e^{-\theta x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$p(\theta|x) = \prod_{k=1}^n \theta e^{-\theta x_k}$$

$$\ell(\theta) = \sum_{k=1}^n \ln(\theta e^{-\theta x_k})$$

$$= \sum_{k=1}^n \ln \theta - \sum_{k=1}^n \theta x_k$$

Suppose, we have a probability density which is given by so I will take an example. So, I have $P(x/\theta)$, which is given by $\theta e^{-\theta x}$, for $x \geq 0$ and this is equal to 0 otherwise. And suppose again we have k number of samples so our problem is to have the best estimate, or the maximum likelihood estimate of this parameter θ .

So, I follow the same procedure that is, I found out the likelihood value $P(D/\theta)$, which is

nothing but $\prod_{k=1}^n \theta e^{-\theta x_k}$. Where k varies from 1 to n , once I have this likelihood, I found out

what is the log likelihood $l(\theta)$, which is nothing but logarithm of this. So, logarithm of this will simply be, logarithm of this term so the product term will be converted into summation term.

So, what I will have $\sum_{k=1}^n \ln(\theta e^{-\theta x_k})$, where k varies from 1 to n , this will simply be

$\sum_{k=1}^n \ln \theta - \sum_{k=1}^n \theta x_k$ because here again a product term that will be converted into two summation

terms so $\sum_{k=1}^n \ln \theta - \sum_{k=1}^n \theta x_k$ where k varies from 1 to n . And as before, it is $\ln(\theta)$ that will be

added n number of times because it is a constant term that I am adding n number of times.

So, this $n \ln \theta - \sum_{k=1}^n \theta x_k$, where k varies from 1 to n so this I can rewrite in the form. So, here

you find that it is θx_k , sum is taken over 1 to n , so I can take out θ outside summation. So, it

simply becomes $n \ln \theta - \theta \sum_{k=1}^n x_k$, k varying from 1 to n .

(Refer Slide Time: 48:39)

The image shows a handwritten derivation of the maximum likelihood estimate for the parameter θ . The steps are as follows:

$$l(\theta) = n \ln \theta - \theta \sum_{k=1}^n x_k$$

$$\nabla_{\theta} l = \frac{n}{\theta} - \sum_{k=1}^n x_k = 0$$

$$\Rightarrow \frac{n}{\theta} = \sum_{k=1}^n x_k$$

$$\Rightarrow \hat{\theta} = \frac{n}{\sum_{k=1}^n x_k} = \frac{1}{\frac{1}{n} \sum_{k=1}^n x_k} = \frac{1}{\bar{x}}$$

So, it will simply be $n \ln \theta - \theta \sum_{k=1}^n x_k$, where k varies from 1 to n so that is my log likelihood

$l(\theta)$. Next what I have to do is, I have to take the gradient of this. So, I have to find out the gradient of l and equate that to 0. So, if I take the gradient of l what do I get, gradient of this term with respect to θ , is nothing but $\frac{n}{\theta}$ because it is $\ln(\theta)$ if I take it, if I differentiate this

with respect to θ , I get $1/\theta$. So, this is simply $\frac{n}{\theta}$, if I differentiate this term with

respect to θ , this θ goes away so I simply have $\sum_{k=1}^n x_k$. So, it becomes $\sum_{k=1}^n x_k$, where k varies

from 1 to n and I have to equate $\frac{n}{\theta} - \sum_{k=1}^n x_k$ to 0.

So, when equate this to 0, you simply get $\frac{n}{\theta} = \sum_{k=1}^n x_k$, k varying from 1 to n . So, this simply

gives you the maximum likelihood estimate of θ , which is $\bar{\theta}$ is, nothing but $\bar{\theta} = \frac{n}{\sum_{k=1}^n x_k}$, k

varying from 1 to n , which is nothing but $\bar{\theta} = \frac{1}{n} \sum_{k=1}^n x_k$, k varying from 1 to n , which is

nothing but $\frac{1}{\mu}$, that is the arithmetic mean. So, this is what my maximum likelihood estimate,

if the probability density function is of this form.

So, what you have seen today is that, if the probability density function has a known parametric form so far our assumption is that, the pdf has a parametric form or a known parametric form. That means, I know that what are the parameters, that is what are the parameters which represent that particular probability density function. Now given that, and set of samples that is in case of supervised learning, I know the set of samples which are drawn according to that probability law, described by that set of parameter vectors.

So, using the information available in this set of samples, how I can best estimate the parameter vector and for doing that, what we have done is, we have used simply our differential calculus, that we have, that you have learned during your school level. Take the gradient equate, the gradient equal to 0 and the value that you get is, what is called the

maximum likelihood estimate, that is, this is the set of parameters which maximally supports the set of samples, which are drawn independently according to the given probability law.

And of course, as I said that the accuracy of this estimation will depend upon the number of samples that you have. Naturally, if I just have 5 samples I cannot say that these 5 samples will follow Gaussian distribution or normal distribution of a given mean vector and given variance. But if I have say, 5000 samples I can say that these 5000 samples, using these 5000 samples, I can accurately estimate. Of course, within some error, I can accurately estimate the value of mean and the value of variance. So, the accuracy always depends upon the number of samples that I have, more the number of samples more accurate my estimation will be and the less the number of samples, less accurate my estimation will be.

In some cases, the sample size is so small that I cannot go for parameter estimation at all or even I do not know what is the parametric form, I cannot even predict what is the parametric form. So, in such cases what we have to go for is, the non-parametric method, I do not use any parametric probability density function, but given the set of samples you estimate what is the probability, that a sample has some values x . Given, a distribution of a set of points. So, I will stop here today, more in the next class.

Thank you.

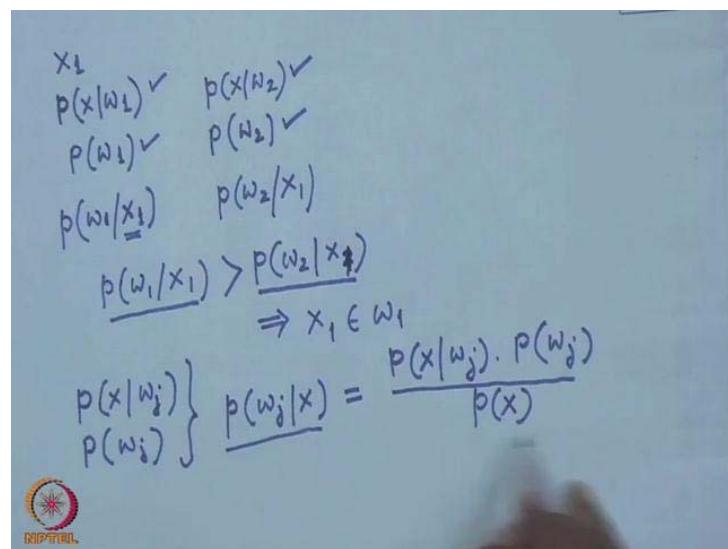
Pattern Recognition and applications
Prof. P.K. Biswas
Department Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 11
Probability Density Estimation

Hello, so welcome back to this video lecture series on pattern recognitions and applications. So, today what we are going to discuss is given a set of examples, how we can estimate the probability density function? The problem is very important, because you might have noticed that the last few classes, we have discussed about the Bayes decision theory, we have also discussed about the maximum likelihood estimate of the parameters.

So, when we talked about the Bayes decision theory we have said that if you are given say two classes, class ω_1 and class ω_2 , and given an unknown sample say x , a feature vector x , we have to classify this feature vector x into one of these two class. That is, I have to decide feature vector x which has to be classified to class ω_1 or it has to be classified to class ω_2 . And for that what Bayes decision theory says that I have to estimate what is the probability, that this unknown feature vector x belongs to class ω_1 and what is the probability that this unknown x belongs to class ω_2 ? So, for whichever class the probability is more, I have to classify this unknown feature vector x to a corresponding class.

(Refer Slide Time: 02:02)



The image shows handwritten mathematical notes on a light blue background. At the top left, there are two sets of terms: x_L and $p(x|\omega_L) \checkmark$ on the left, and $p(x|\omega_2) \checkmark$ on the right. Below these, $p(\omega_L) \checkmark$ is on the left and $p(\omega_2) \checkmark$ is on the right. In the center, there are two fractions: $p(\omega_1/x_1)$ on the left and $p(\omega_2/x_1)$ on the right. Below these, the fraction $p(\omega_1/x_1) > p(\omega_2/x_1)$ is written, with an arrow pointing to the right under the fraction. This leads to the conclusion $\Rightarrow x_1 \in \omega_1$. At the bottom, there is a bracketed term $\left. \begin{matrix} p(x|\omega_j) \\ p(\omega_j) \end{matrix} \right\}$ followed by the equation $\underline{p(\omega_j|x)} = \frac{p(x|\omega_j) \cdot p(\omega_j)}{p(x)}$. In the bottom left corner, there is a small logo with the text "NITTEL".

In other words, what we have done is, so we have this unknown feature vector x and I know what is the probability $P\left(\frac{x}{\omega_1}\right)$ or let me put it as unknown feature vector x_1 . And I have this class conditional probability density function which is given by $P\left(\frac{x}{\omega_1}\right)$, that is the probability density function of samples which come to the ω_1 . And I also know what is the class conditional probability, that $P\left(\frac{x}{\omega_2}\right)$ and it is also assumed that I know what is the a priori probability of class ω_1 and class ω_2 . That is, I know what is the a priori probability of class ω_1 ?

$$P\left(\frac{\omega_2}{x_1}\right) > P\left(\frac{\omega_1}{x_1}\right) P\left(\frac{\omega_j}{x}\right) P\left(\frac{x}{\omega_j}\right) \frac{P(\omega_j)}{P(x)} g_j(x) \ln P\left(\frac{x}{\omega_j}\right) P(\omega_j) P\left(\frac{x}{\omega_i}\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu_i)^2}{2\sigma^2}\right] \neq \\ P\left(\frac{x}{\omega_1}\right) P\left(\frac{X}{\omega_j}\right) = \frac{1}{(2\pi)^{\frac{1}{2}} |\Sigma_j|^{\frac{1}{2}}} \exp\left[-\frac{1}{2} \left\{ (X-\mu_j)^T \Sigma_j^{-1} (X-\mu_j) \right\}\right] P(X) = x \geq 0 P(k;\lambda) = P_r(x=k) \frac{\lambda^k e^{-\lambda}}{k!} P(\omega_i) \\ P\left(\frac{\omega_i}{x_1}\right) P\left(\frac{x_1}{\omega_j}\right) P\left(\frac{x_1}{\omega_i}\right) \geq \frac{n_j}{N} x^T P(x) h_j = \frac{n_j}{N \cdot W_j} \sum h_j = \bar{x}$$

And I know what is $P(\omega_2)$ which is nothing but the a priori probability of class ω_2 and even this I have to find out the probability of what is $P\left(\frac{\omega_1}{x_1}\right)$, where x_1 is the unknown feature vector. So, once I have this observation, this unknown feature vector x_1 I have to compute what is the probability that belongs to class ω_1 , which is nothing but $P\left(\frac{\omega_1}{x_1}\right)$.

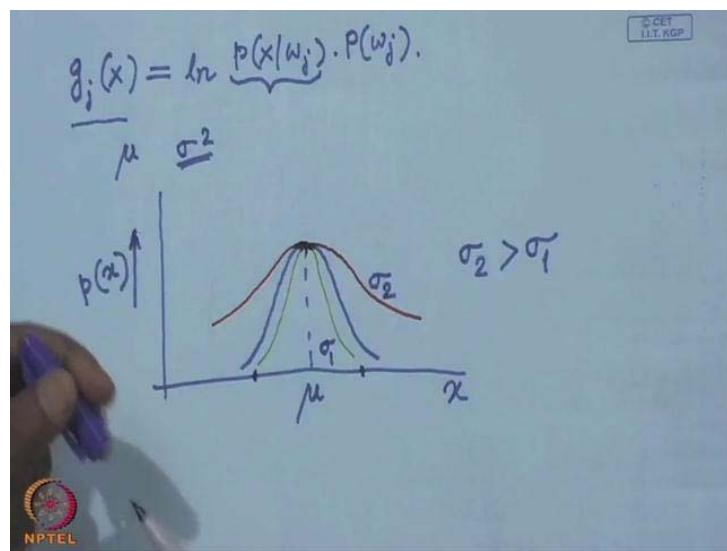
I also have to compute what is $P\left(\frac{\omega_2}{x_1}\right)$, that is given this unknown feature vector x_1 what is the probability that this x_1 belongs to class ω_2 . And then the Bayes decision rule says that if $P\left(\frac{\omega_1}{x_1}\right) > P\left(\frac{\omega_2}{x_1}\right)$, then my decision will be that x_1 belongs to class ω_1 . Otherwise, if $P\left(\frac{\omega_2}{x_1}\right) > P\left(\frac{\omega_1}{x_1}\right)$, then decision classification decision will be on x_1 belongs to class ω_2 .

So, that is how the Bayes rule works and when I compute this $P(\omega_1/x_1)$ or $P(\omega_2/x_1)$, we use the Bayes rule. That is as I know that what is $P(x/\omega_1)$ and what is the a priori probability, that is what is $P(\omega_1)$. Similarly, I know what is the class conditional probability density function, that is $P(x/\omega_2)$. I also know what is $P(\omega_2)$, what is which a priori probability or in general I know what is the class conditional probability function, that is $P(x/\omega_j)$, which is the class probability density function for the samples belonging to the class ω_j , I also know that what is the a priori probability that is $P(\omega_j)$ and from these two I can compute the a posteriori probability, what is $P(\omega_j/x)$ which is nothing but

$$\frac{P(x/\omega_j) \cdot P(\omega_j)}{P(x)}.$$

So, this is the one which has to be computed or a posteriori probability which has to be computed for all the classes. And the class for which this posteriori probability that $P(\omega_j/x)$ will be maximum, this x will be classified to the corresponding class. So, this was the basic Bayes classification rule and following this in last few classes we have also derived a number of functions called decision functions or discriminating functions. So, the discriminating functions for a class ω_j .

(Refer Slide Time: 06:47)



We have defined that as $g_j(x)$, which is the discriminating function of class ω_j and this we

have defined as $g_j(x) = \ln P\left(\frac{x}{\omega_j}\right) \cdot P(\omega_j)$

And while defining this you see that we have ignored one term which was in the denominator, that is $P(x)$. This we have ignored simply, because this denominator term is equal in all the a posteriori probabilities expression. So, whatever the ω_j this $P(x)$ will appear in all these a posteriori probabilities expressions. As a result, this $P(x)$ does not influence the decisions that to which class the x will belong.

So, when we have found out the expressions for $g(x)$, that is the discriminating function, we

have assumed that $P\left(\frac{x}{\omega_j}\right)$. That is the class conditional probability density function that

follows a predefined form or the expressions that we have derived for this $g(x)$. For that we have assumed that the form is a Gaussian distribution or a normal distribution. So, when we have a Gaussian distribution, you know that the probability density class that is internally defined by two parameters. One is the mean μ and the other one is the variance, which is σ^2 .

So, if I go for, if I look at the probability density function which is defined over a single variables say x or a scalar variable say x , then if I know μ that is the mean of the samples. And if I know the variance which I know that is σ^2 of the samples, I have defined what is the class, which represents this probability density function, okay. So, if I plot, it will be something like this. This is my variable x and along the vertical direction I plot $P(x)$ and the curve. If the μ is the mean of sample x then the curve will be something like this.

So, you all know that the position of the peak is defined by is decided by what is the mean of the samples, which is mean μ and the shape of this probability density called that depends upon the variants which is σ^2 . So, if σ^2 of the variance is very small in that case the curve will be very sharp. It will be something like this when the various is small and if the variance is large then the curve will be a flat one, which is like this. So, if this curve is for σ_1 , where σ_1 is the standard deviation and the σ is the variance and this is the curve, which is for σ_2 , mean value μ remaining the same. this indicates that $\sigma_2 > \sigma_1$ and if the mean is changed then the position of the peak of this probability density function that will shift along x axis.

So, for μ_1 , if $\mu_1 > \mu$, then this will be shifted to this location for another μ value. If that is less than μ , this curve will be shifted to the left. So, this is

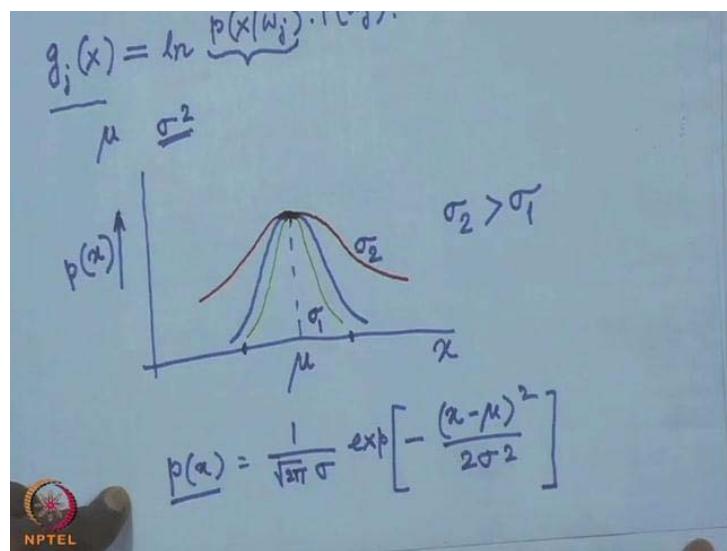
what we get for a single variable. If I have multi variable in which case the feature is represented in the form of a feature vector.

(Refer Slide Time: 11:41)

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_d \end{bmatrix}$$

So, such a feature vector is represented by capital X. And if this feature vector has say d number of components, then feature of vector capital X will be given by $[x_1, x_2, x_3, \dots, x_d]^t$ which becomes a vector having the number of components. And each of them whether its x_1 or x_2 or x_3 or x_d in general say ninth element, say x_i . This represents a particular element of the feature vector element x or it gives a particular characteristic or a particular feature of the pattern, which you want to classify and then generalizing this Gaussian expression so in this case.

(Refer Slide Time: 12:35)



You all know that the Gaussian expression, in this case will be $P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$.

So, this is well known form of a Gaussian expression and over here this represents the probability density functions of x. And if I want to make it a class conditional probability density function, that means instead of taking all the x, if I take the samples which belongs to class say ω_i .

(Refer Slide Time: 13:30)

$$P(x|\omega_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right] \checkmark$$

So, I want to make it $P(x/\omega_i)$. So, the expression of $P(x/\omega_i)$ will be similar. It will be

$P(x/\omega_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right]$. So, the only difference between this expression and this

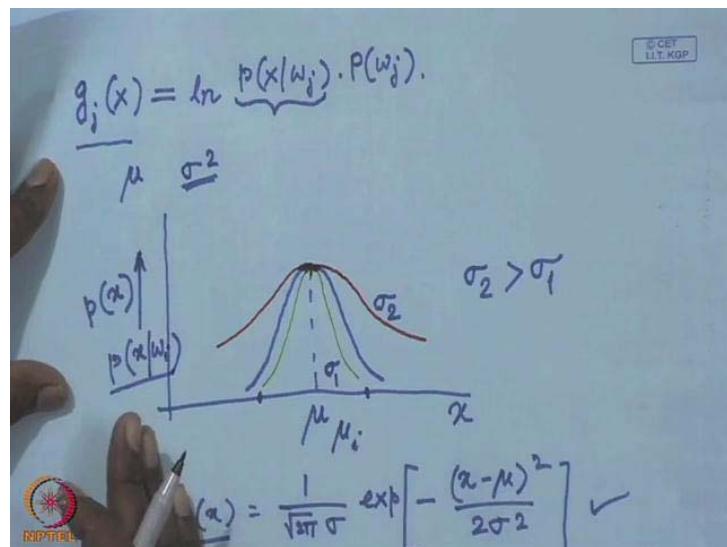
expression is that here, where we have taken different sample values of x. We have now considered whether these samples are taken from class ω_1 or the samples are taken from class ω_2 or the samples are taken from the class ω_3 and so on. We put all the samples together irrespective of what is the class belongings of the sample, but when I look at this expression, that is $P(x/\omega_i)$, I take only the samples from class ω_i . I do not take samples from class ω_j , where $j \neq i$.

So, naturally if I want to find out what is $P(x/\omega_1)$, I will take samples only from class ω_1 . I will not take any sample from class ω_2 , I will not take any sample from class ω_3 and so on.

Similarly, if I want to compute what is $P(x/\omega_3)$, I will take only samples from class ω_3 . I will not take any samples from ω_1 or ω_2 or ω_4 and so on.

So, once I take only samples from class ω_1 , this μ_i simply becomes μ_1 and the variance that I compute that is σ_i^2 that becomes σ_1^2 . So, it is computed only from the samples which are taken from the corresponding class, that is what is class conditional probability and in such case this density function, the probability density function that I plot

(Refer Slide Time: 15:56)



it simply becomes $P(x/\omega_i)$ and this μ will be replaced by μ_i , that is μ of the samples taken from class ω_i and this is the probability density function of the samples taken from class ω_i . So this is what we have in case of a single variable. Now, how this expression changes if I go for multiple variables, that is instead of x being a scalar. Now, x is a vector, so where the vector I represent in this form, it is represented by capital X and suppose, this vector capital X has d number of components from x_1 to x_d .

So, this expression that we had written in case of a variable. Now, it has to be converted, it has to be written for a vector, okay. So, in that case obviously this σ_i which is the variance of single variable of the samples taken from class ω_i . Now, this σ_i or the variance σ_i^2 has to be replaced by another term, which is called co variance matrix, right?

Because now, I am dealing with multiple number of variables where every variable represents one component of my feature vector. So, instead of simple variance I have to consider a co variance, that is how, what is the variance of x_2 to respect x_1 , earlier it was the variance of x to respect x . Now I have multiple components. So I have consider, what is the variance of x_2 and x_1 taken together, what is the variance of x_3 and x_4 taken together.

So, instead of a simple variance it will be covariance matrix. Similarly, this μ_i instead of being a scalar value, now it will be a vector value because this has to capture what is the mean of the components x_1 , what is the mean of the components x_2 , what is the mean of the components x_3 and so on. So, this μ_i will again be a vector which is the mean vector of the samples taken from class ω_i .

(Refer Slide Time: 18:27)

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_d \end{pmatrix}$$

$$p(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[-\frac{1}{2} (x - \mu)^t \Sigma^{-1} (x - \mu) \right]$$

$$p(x|\omega_j) = \frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp \left[-\frac{1}{2} (x - \mu_j)^t \Sigma_j^{-1} (x - \mu_j) \right]$$


So, as a result the probability density function for a vector will be simply

$$P(X) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[-\frac{1}{2} (X - \mu)^t \Sigma^{-1} (X - \mu) \right].$$

So, if you look at this particular expression,

in the vector form the probability density function becomes

$$P(X) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[-\frac{1}{2} (X - \mu)^t \Sigma^{-1} (X - \mu) \right].$$

Where this d is the dimensional of this, that is

the number of components that I have within this feature vectors. And Σ is nothing but the covariance matrix.

Where this μ is nothing but the mean vector of all the samples which are the sample values of the variable X . So, know the X is a vector variable, it is not a scalar variable and in the same manner if I want to put, what is the class conditional probability that is $P\left(\frac{X}{\omega_j}\right)$, which will be given by $P\left(\frac{X}{\omega_j}\right) = \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma_j|^{\frac{1}{2}}} \exp\left[-\frac{1}{2} \left\{ (X - \mu_j)^T \Sigma_j^{-1} (X - \mu_j) \right\}\right]$. So, this will be the expression of the class conditional priority density function for all the vectors X or sampled vector X taken from class ω_j .

Now, with that I have this expression for a single variable or I have for a vector variable the expression which is something like this. As we said that you find that the shape of the probability density class is entirely defined, decided a number of parameters. In case of a single variable it is decided by μ , which is the mean of X and the variance σ^2 . In case of vector variables, the shape and position of the probability density curve is decided by the mean vector which is μ_j .

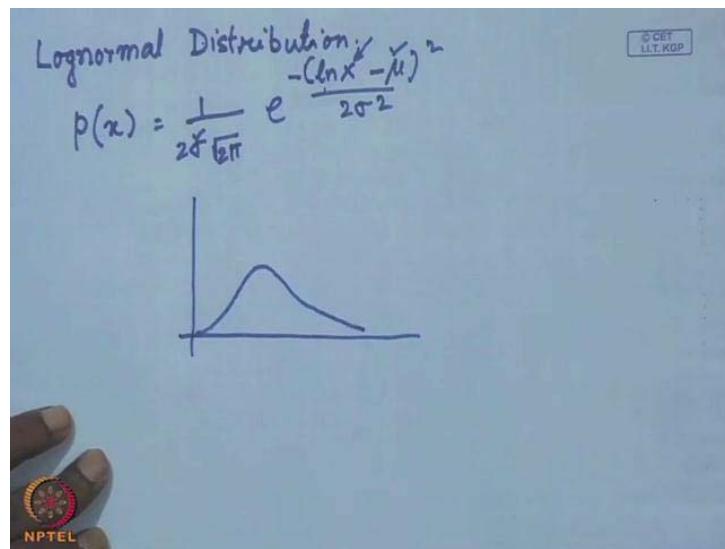
So, this mean vector decides the location of the probability density curve and in case of a vector instead of a curve it will be a surface. So, this μ_j decides that what is the position of that probability density surface, which represents the probability density function and the shape of the surface will be decided by covariance matrix, which is μ_j, Σ_j . So, these are the parameters, the mean vector and the covariance matrix which determines, what is the position and the shape of the probability density function. So, these are the expressions whether it is this or it is this. These are called parametric expressions.

So, the probability density function has a parametric form and we find that previous lecture when we talked about the maximum likelihood estimate of parameter values. We have assumed that the probability density function follows a parametric form. That means I have or more parameters which decides what is the position and shape of the probability density surface or probability density function and knowing those parameters, what are the parameters we have tried to estimate tried to find out the best estimate of those parameters from a set of samples which are given.

So, this is one such form of parametric representation of the probability density function and obviously this Gaussian is representation or the representation of the probability density

function by mean and the variances is not the only way in which a parametric probability density function can be represented. There are many other parametric probability density functions.

(Refer Slide Time: 24:47)



One of them is, what is called log normal distribution. The log normal distribution is given by

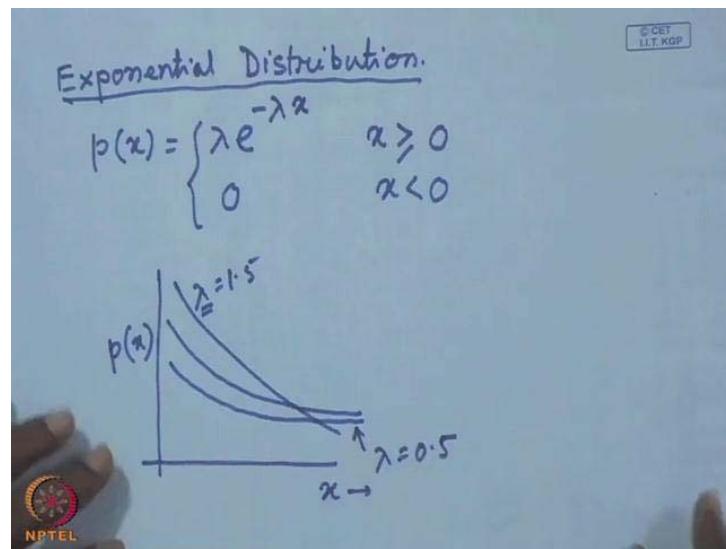
$$P(x) = \frac{1}{2\sqrt{2\pi}\sigma} \exp\left[-\frac{(\ln x - \mu)^2}{2\sigma^2}\right].$$

So, this is an expression for what is called log normal

distribution, and here again you find that in case of lognormal distribution, we have two different parameters. One of the parameters is σ as before which is the standard deviation. And the other parameter is μ , as before this is the mean of the sample values, okay.

So, a notable difference between a Gaussian distribution or a normal distribution and a log normal distribution is that in case of normal distribution, we have found that what we have put is the exponentiation contains $(x-\mu)^2$. In case of log normal distribution, the exponentiation or the exponent contains $(\ln x - \mu)^2$. If you do this distribution, the distribution will look something like this, which is similar to our Gaussian distribution, but there is slight difference. The slopes on this side and the slope on this side, they may be different or in general they are different. Similarly, there is another form of parametric distribution which is called exponential distribution.

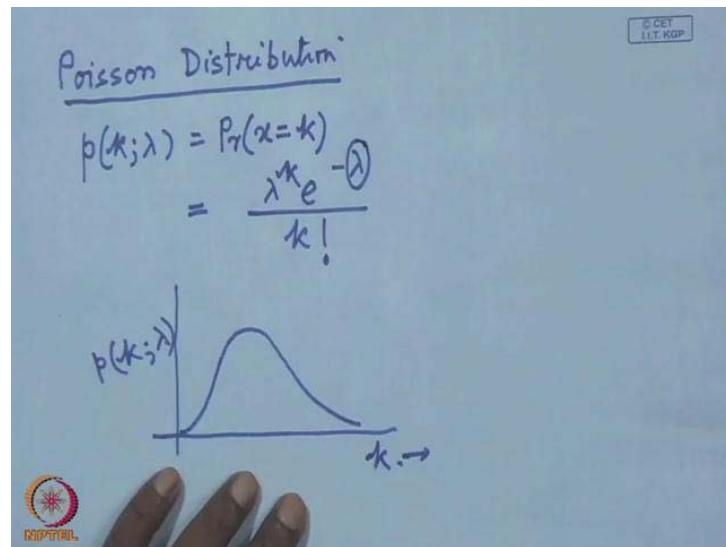
(Refer Slide Time: 26:57)



And the exponential distribution is given by $P(x) = \lambda e^{-\lambda x}$, for $x \geq 0$. And this is equal to 0 for $x < 0$. And if I plot this exponential distribution, the plot will be something like this. For different values of λ , I will get plots like this, so as I increase the value of λ as may be λ is equal to 1.5. This may be for say $\lambda = 0.5$.

So, this is my x and this is $p(x)$, so again in case of exponential distribution you find that your probability distribution function is entirely defined or decided by a single parameter, which is this λ . So, if I can estimate what is the value of λ , then I know that what is the probability density function or the probability distribution function, similarly another parametric distribution, probability distribution.

(Refer Slide Time: 28:53)



which is widely used in some of the applications is what is called Poisson distribution and the analytic form of the Poisson distribution is $P(k;\lambda)$ is simply this actually represents what is the probability that the variable x takes a value is equal to k . And the analytic expression for

this is simply the $\frac{\lambda^k e^{-\lambda}}{k!}$ and if I plot this Poisson distribution, the Poisson distribution again

appears something like this. So, again will find, so this is what is k and this is what is $P(k;\lambda)$? So, for different values of λ I get such different plots and here again you find that the shape of this probability distribution function is entirely defined by a single parameter, which is λ .

So, my whole purpose of telling you this is that when I go for say maximum likelihood estimate of the probability density functions, I have to know that what is the parametric form of probability density function or probability distribution function, unless I know that what are the parameters which defines that probability distribution function. I cannot use the maximum likelihood estimate, because maximum likelihood estimate after all gives you the best estimate of the parameters. Whichever can be computed the best estimates, that can be computed from the sample values. Now our problem is when I go for classification problem. Now, particularly in case of Gaussian distribution or all these parametric distributions, I can compute these parameter values provided that I have large number of samples representing this distribution.

Now, if the samples, the number of samples is not very large, then whatever estimates of the parameters I do that may not be faithful representation of the actual scenario. And the major problem is I do not know beforehand that what is the parametric form that this distribution is going to follow, the parameters are not known, so unless I know the parametric form I cannot estimate the parameters and in most of the situations I do not know what is the parametric form. So, given such situations how we can still classify the unknown samples following Bayes classification rule? So, if you look at what Bayes classification rule says in case of Bayes classification rule, what we have said is simply $P(\omega_i)$ given an unknown sample say x_1 .

(Refer Slide Time: 32:32)

$$p(\omega_i | x_1) > p(\omega_j | x_1) \Rightarrow x_1 \in \omega_i$$

$$p(x_1 | \omega_i) = \frac{p(x_1 | \omega_i) \cdot P(\omega_i)}{P(x)}$$

If it is greater than $P(\omega_j)$ for the same unknown sample x_1 , then we decide that x_1 belongs to class ω_i . So, what I need to know is what is this a posteriori probability $P\left(\frac{\omega_i}{x_1}\right)$. So, I need to compute this only for this unknown sample which is x_1 , and to compute this the probability density function or the probability density value that I have to use is what $P\left(\frac{x_1}{\omega_i}\right)$ because as we said earlier that $P\left(\frac{\omega_i}{x_1}\right)$ is nothing but $P\left(\frac{x_1}{\omega_i}\right)$ into a prior probability $\frac{P(\omega_i)}{P(x)}$. As we said that $P(x)$ appears in the denominator for such expressions, for different value of i or for different class ω_i .

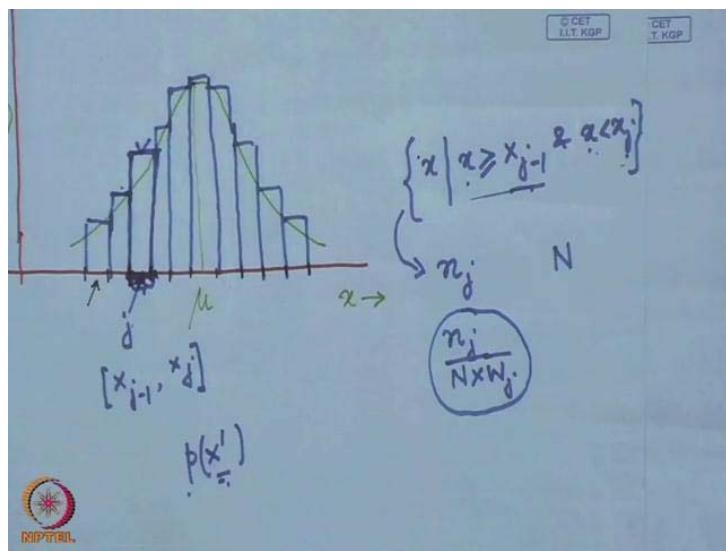
So, this $P(x)$ does not contribute anything to our decision because it is same for all the classes. So, what I really have to compute is what is $P\left(\frac{x_1}{\omega_i}\right) \cdot P(\omega_i)$, and $P(\omega_i)$ being a prior probability, it is always assume that is known priori. So, what I have to compute effectively is this value, what is $P\left(\frac{x_1}{\omega_i}\right)$. Now, the advantage that I have if I have a parametric representation something like this.

So, this being an analytical expression for given any x , I can immediately compute this value. So, that is the advantage that I have, but for that I have to know that these are the parameters which defines the parametric, that particular probability density function. And if I know that

these are the parameters, then only I can go for the best possible estimate of these parameters from the known samples, but our problem is that we don't know what is the parametric form of the probability distribution. And if I do not know what is the parametric form, then I cannot form any analytical expression like this.

So, my job will be that I have to compute $P(x_1 \in \omega_i)$, where x_1 is an unknown sample making use of the known samples which are provided for supervised learning. So, my problem is that, so I have a set of samples and using those set of samples I have to estimate what is $P(x_1 \in \omega_i)$. and Obviously, I do not assume that this probability distribution function follows any parametric, any known parametric form. So, how we can do that that is the problem that we are going to discuss today and may be tomorrow and day after as well.

(Refer Slide Time: 36:48)



So, let us start with this, let us start with the known form or Gaussian distribution function which was something like this. I had this mean location μ and for certain variance σ^2 , the density curve is something like this. So, this is my variable x , so this axis represents variable x and this axis represents $p(x)$ for which I know, what is μ . I know what is σ ? So this is the form I have. And you find that this is a continuous curve and for this I have an well-known analytical expression which is nothing but Gaussian expression.

Now, what I can do is this analytical curve I can represent by piecewise constant approximation.

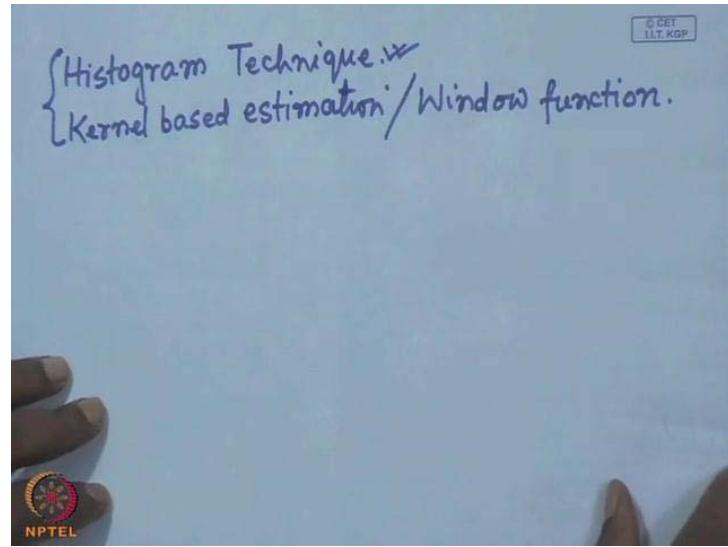
So, what I mean by piecewise constant approximation? What I do is this axis representing my variable x , I divide this axis into a set of intervals, something like this. I have intervals given like this. So, these are the different intervals in which the x axis or the axis representing the variable is divided and I assume that in each of this interval, the probability density is constant. So, the way I can represent this continuous curve by a piecewise constant representation is something like this.

So, this is a piecewise constant representation of this continuous probability density curve. Now, was I do that if I take any of this intervals say j^{th} interval, you find that in the j^{th} interval the height of this bar that have that indicates what is the fraction of the samples that falls under this particular interval, right? So, for all the sample x , suppose the boundary of this interval is given by x_{j-1} to say x_j . So, these are the limits of this particular interval, so all x where $x \geq x_{j-1}$ and $x < x_j$. If I take the set of all such samples, whether the sample value is greater than x_{j-1} , where the x_{j-1} is the left boundary of this j^{th} interval. And it is less than x_j where the x_j is the right boundary of the interval and the number of such samples if represent by n_j .

So, n_j is in the number of such samples which follows this condition and suppose capital N is the total number of samples that I have. Then this height actually represents $\frac{n_j}{N}$ into width of this interval, if i say w_j , that is the width of the interval w_j . So, this is nothing but what is the probability that a sample falls within this j^{th} interval. So now, given any other value of x' which is unknown and I have to compute what is $P(x')$. I simply have to look for that in which of this is intervals x' falls.

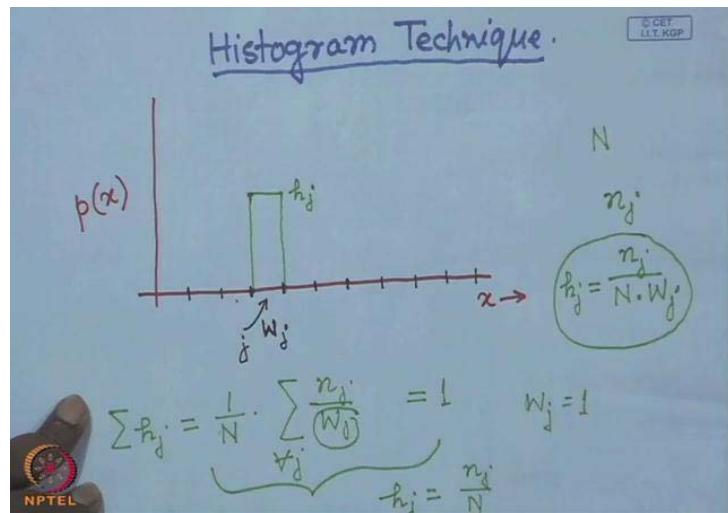
So, if x' falls within this j^{th} interval, then immediately I can say this $P(x')$ is given by the height of this bar. So, given any analytical expression I can always make a piecewise constant expression of that analytic expression of that probability distribution which is given like this. So, this gives as hint that even if I do not know what is the parametric form of the probability density function, but still I can estimate what is the probability of a particular unknown sample, so there are two ways.

(Refer Slide Time: 43:32)



So, there are many ways discuss mainly two of them, one of them is called a histogram technique and the other one is called kernel based estimation technique or this is also called making use of window function. So, we will mainly discuss about these two different techniques for probability density function estimation. So, first will talk about this histogram based technique.

(Refer Slide Time: 44:31)



So, let us see what is the histogram based technique for probability density function? So, we will talk about, so will first explain the concept using a scalar variable x or I have a scalar

variable x and then we can extend the concept to multiple dimensions, when we have features which are not scalar, but we have feature vectors having multiple number of components. So, let us see that how we can do it for a single variate case or scalar variables. So, first what we do is because it is a scalar variable. So, I will have horizontal axis which have variable x or different samples of the variable x , and the vertical axis will represent the probability density estimate which is $P(x)$.

So, as we have seen earlier that when we have gone for piecewise constant approximation of a continuous probability density curve where we have said that we divide the variable axis or x axis into a number of intervals. Now, width of different intervals may be same or width of different intervals may also be different. In general, different intervals have different width, but for simplification we can make the width equal, all the widths to be same. So, this is the kind of situation that we have, following the same concept what we do over here is we divide this x axis into a number of intervals.

So, the intervals are like this, out of this I take a particular interval say j^{th} interval for which the width will be given by W_j . Then what I have is the situation is something like this, I have large number of samples which represent different sampled values of the variable x . When I tried to estimate the probability density function along the horizontal axis, I put all those different sampled values of x and along the vertical dimension I will put the estimated value of the probability density function. And for doing this, what I have done is out of all these different intervals, I have taken by j^{th} interval whose width I have assumed to be W_j .

So, what I need to do is, I have to check all the samples and try to find out that out of all those samples, suppose I have total of capital N number of samples. So, out of this capital N numbers of samples I have to find out that how many samples actually fall within this j^{th} interval. Now, while doing so it may so happen that some of the samples will just fall on the boundaries, that is either on the boundary between the j minus first interval and j^{th} interval or on the boundary between j^{th} interval and the j plus 1 interval. So, in such situations obviously there is an ambiguity whether that sample should be considered to belong to j th interval or the sample should be considered to belong to j minus first interval.

Similarly, if the sample falls on the boundary between the j^{th} interval and the j^{th} plus first interval, there is an ambiguity that whether the sample should be considered to belong to j^{th} interval or it should be considered to $j^{\text{th}}+1$ interval. So, to avoid such ambiguity conventionally what is done is, whenever I have sample falling on the boundary

conventionally the sample is considered to belong to the to the interval which is to the right of it.

So, if I have a sample falling over here. I will consider that the sample belongs to the class of j plus first interval. If I have a sample following over here, I will consider that sample to belong to j^{th} interval. So, by considering this if I find that I have n_j number of samples which falls within this j^{th} interval, then I will have a bar graph representing the probability function. Where the height of the bar will be given by $h_j = \frac{n_j}{N \cdot W_j}$, where this W_j is width of

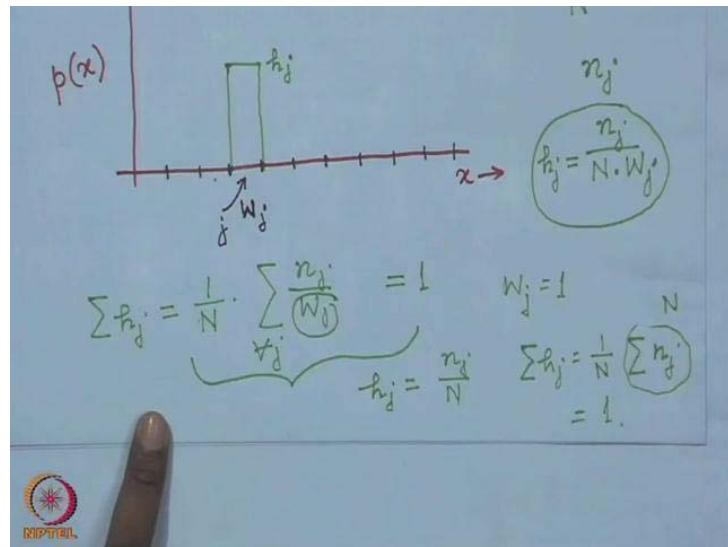
the j^{th} interval, capital N number is the total number of samples that I have and n_j is the number of samples. Out of these capital N number of samples the number of samples which falls in the j^{th} interval and this is what will give me the h_j or the height of the bar representing this probability density function.

So, I will have a situation something like this, I have a bar, rectangular bar placed over this where height of this rectangular bar in this h_j . So, which actually represents that what is the probability that x will have a sample value within this j^{th} interval value. Now, this particular computation has significance. The significance is that if we assume that this histogram what we get, so collection of bars is nothing but histogram.

So, if this histogram has to represent or approximate a continuous probability density function or continuous probability distribution function, then area under this probability distribution function has to be equal to 1 or in other words, I must have sum of h_j which is nothing but $h_j = \frac{n_j}{N \cdot W_j}$, for all j that must be equal to 1 and you find if I compute this h_j in this form this condition $\sum h_j = 1$ will always be satisfied.

Now, in a particular case that if assume that this $W_j = 1$, that is every interval has an unit width, in that case h_j , that is the probability density estimate in this interval in this j^{th} interval will be simply be $h_j = \frac{n_j}{N}$. So, it will be something like this in $h_j = \frac{n_j}{N}$, and in such case this is quite obvious because $\sum h_j$ will be simply $\frac{1}{N} \sum n_j$.

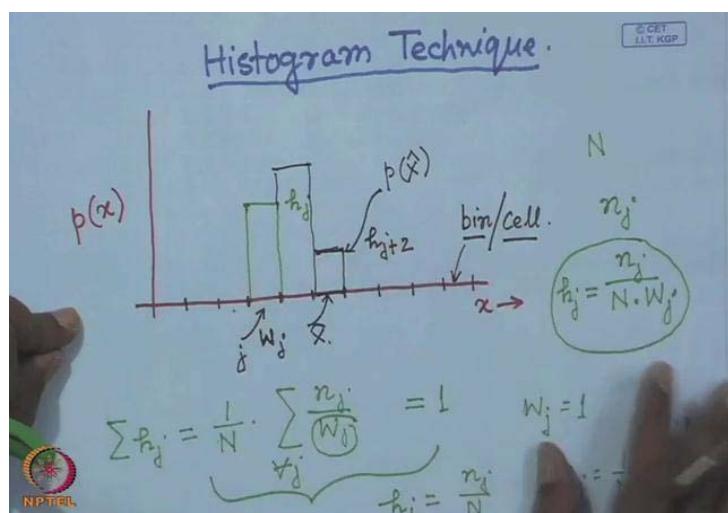
(Refer Slide Time: 53:19)



And the $\sum n_j$ nothing but capital N. So, this will be obviously $\sum h_j = 1$ which is also true in this particular case and that can be easily proved. So, given this what I have to do is in summary I have a large number of samples of the variable x, the x axis of the probability density function. When I plot the probability density function the x axis which represents my variable x, it is divided, broken into a number of intervals and for breaking this number of intervals, I have to have an idea of what is the minimum value of x and what is the maximum value of x or in other words what is the range of x.

So, once I have this knowledge of the range of x, that range I can always divide into a number of intervals. Then for this given number of samples, out of this total number of samples I have to find out, what is the fraction of samples which falls within the particular interval. The fraction of samples divided by the width of the interval tells me that what is the probability density function, estimate of the probability density over that particular interval.

(Refer Slide Time: 55:08)



So, once I have this kind of estimate, so this h_j is the probability density for this j^{th} interval. This the probability density of first interval, this may be probability density function of second interval and so on. So, if I have any unknown value of x , say \bar{x} over here, then immediately I will say that value this is say \bar{x} a sample value \bar{x} , immediately I will say that $P(\bar{x})$ is nothing but this, which is nothing but h_{j+2} that is the height of the bar in $j+2$ interval.

So, obviously it is here, assumed that within an interval the probability density is constant and the probability density will be different is likely to be different in different intervals. So, when I break this x axis variable into a number of interval something like this, each of these interval is usually called a bin or a cell or I can either call it a bin or a cell. So, this is how the histogram technique works, histogram based probability density function works. So, in the next class we will explain for the help of some examples, and then will extend this to multi dimension when x instead of being a scalar variable it is vector.

Thank you.

Pattern Recognition and Application
Prof. B. K. Biswas
Department of Electrical and Electronic Communication Engineering
Indian Institute of Technology, Kharagpur

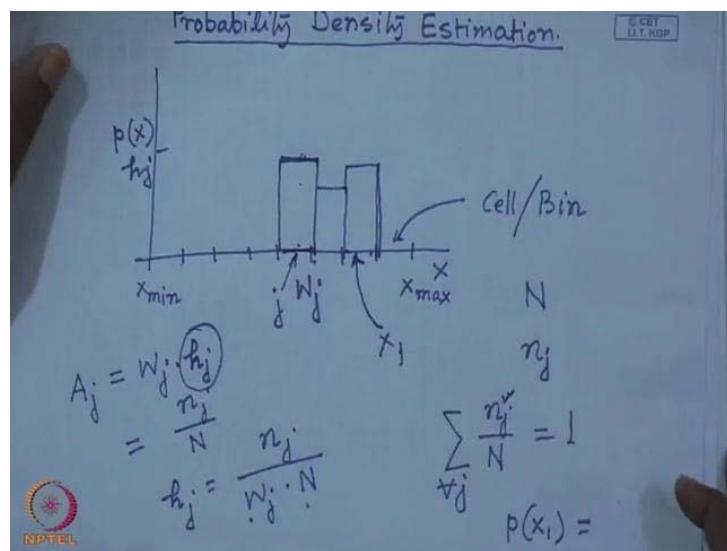
Lecture - 12
Probability Density Estimation (Contd.)

Hello. So, welcome to this lectures series on pattern recognition and applications. In our previous class, we have started discussion on probability density estimation. The reason why we have to go for this topic of probability density estimation is that, though we can have analytical expressions of probability density function in certain cases, Where the analytical expression will use some number of parameters and depending upon the kind of density that we have we have different types of different number of parameters. So, if I want to have analytical expression of the probability density functions I have to know beforehand, that what is the parametric form the probability density function will take.

If I know what is the parametric form or what are the parameters, then I can go for maximum likelihood estimate for the values of those parameters. I can make use of some other technique, to obtain the based possible values of those parameters from the trainings set of samples, which are given, but if the number of samples are very limited. In that case the estimated value of the parameters are not very reliable or not very suitable. Or in some other case we may have the problem that the parametric form is not really known.

We do not know that how to describe the probability density function as represented by the set of samples which are provided. So, from those set of samples we cannot really estimate what are the parameters or which parametric formed the probability density functions has to take. So, unless we know the parameters, we cannot obviously estimate the values of those parameters. So, we have gone for the probability density estimation from the given set of training samples without assuming any parametric form of the probability density function. So, obviously in this case I cannot have any analytical expression, but the probability density estimate that we can that we will get that we will solve our purpose so far as pattern recognition and applications is concerned, so in order to do that. What we have said in the last lecture is something like this.

(Refer Slide Time: 03:08)



If I take a single variable say x , for which I want to estimate the $P(x)$ or probability density function of x . Then first what I have to do is I have to know that what is the range of all those of x , that is what is the minimum value of x and what is maximum value of x . So, if I know the range of values of x , then that range is divided into a number of intervals. We have said that every interval is called either a cell or a bin.

So, what we have done is, suppose this is the minimum value of x or let me put it as x_{\min} . And this is the maximum value of x that I can have say x_{\max} . So, this range of x is divided into a number of intervals. And each of this interval is called a cell or a bin. Now, suppose to estimate this probability density function you have been given certain number of samples of x . So, suppose that total number of samples I have is capital N and I take any bin say j^{th} bin and I try to find out of this total number of samples, sampled values of x , how many such samples like in this j^{th} bin.

So, if the number of samples that lie in the j^{th} bin is given by n_j then I assume that this ratio of $\frac{n_j}{N}$, x may give an estimate of the density or the probability density. So, for as this particular j^{th} bin is concerned. So, this has to represent. So, this represents an estimate of the probability density function in this j^{th} bin. I also assume that in a particular bin, the probability density uniform. So, what I will have is a probability estimate something like this.

So, with in this j^{th} bin, the probability density function is uniform. I can have different values of this probability density function in different things. I also have another constraint that if this probability density function that aim estimating. It has to approximate a continuous probability density function, then the area under the probability density function has to be equal to 1. That means this sum of the areas, of all these bins has to be equal to 1. So, that clearly says that area of this particular area, under this probability density function has to be equal to n_j by N because some of $\frac{n_j}{N}$, if I take the summation over all values of j that is equal to 1.

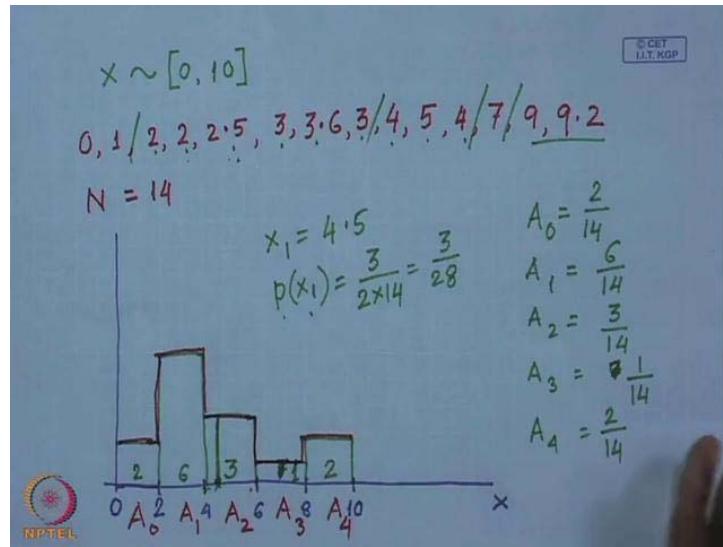
So, given this if the width of these j^{th} bin is equal to say w_j , I have to find out what is the height or the value of the probability density function for this j^{th} bin. And as I have said that the area has to be equal to $\frac{n_j}{N}$. Here n_j is the number of samples which are falling in this j^{th} bin and capital N is the total number of samples that I have. So, the area of these j^{th} bin, if I call it as a_j that has to be equal to w_j , which is the width of the j^{th} bin and h_j which is the estimated probability density function so far as this j^{th} bin is concerned.

So, this area a_j has to be equal to $w_j h_j$ and which is nothing but $\frac{n_j}{W_j \cdot N}$. So, from this I can have what is the value of h_j which is nothing but estimate of the probability density so far as this j^{th} bin is concerned. So, I get the value of h_j which is nothing but $\frac{n_j}{W_j \cdot N}$. Where this w_j is the width of the j^{th} bin, capital N is the total number of samples that I have and n_j is the number of samples, out of this total number of samples, the number of samples which falls under the j^{th} bin. So, this is how we can estimate the probability density function. Once I estimate the value of h_j for all values of j , that means for all the different bins I can have estimated values something like this.

Now, for an unknown x , say x_1 , if I find that this x_1 falls in this bin the probability $P(x_1)$ will simply be given by this $P(x_1)$ is nothing but the height of this particular bin, if x_1 falls within this bin because that is our exemption that within a bin or with in a cell the probability density is constant. So, this is one of the ways in which we can estimate the probability density function from a set of given examples. Now, to explain this further we will take an

example again for simplicity, initially we will take an example for single variable, then will extend this concept to vector, vectors spaces, where I have multi variate probability density function. So, let us take an example.

(Refer Slide Time: 10:05)



Suppose, I have a variable x where this x or the range of x is between say 0 and 10. So, the maximum value that x can take is 10 and the minimum value which x can assume is 0. So, I am assuming that I have the variable x , can assume any value within the range 0 to 10 where both 0 and 10 are in inclusive. And now, if I take a number of samples, say let us say a number of samples like this. So, suppose this is the set of samples which are given to estimate the value of $p(x)$, the probability density function for the variable x .

So, you find that here I have a number of samples, the sampled values of x as 0, 1, 2, 2. So, 2 appears twice. So, because I can have 2 samples having the same values then 2.5, then 3, 3.6, 3 again 4, 5, 4, 7, 9, 9.2. So, the number of samples that I have is 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14. So, I have capital N , that is number of samples is equal to 14. Now, I want to estimate the probability density function. So, for that as we have said earlier, that I will plot this x and let us assume that as this x or the range of x between 0 to 10 will be divided into a number of cells. Let us assume that every cell will be of equal width and we start with a cell width will be equal to 2.

So, this x it will have a minimum value of 0 and maximum value of 10. And I am having cells of width 2. So, the cell boundaries will be 0, 2, 4, 6, 8 and 10. So, these are the cell boundaries that I have. And now, I have to find out that out of these samples which sample falls in which of these bins. So, you find that 0, which is on the cell boundary and we have assumed that if a sample falls on the cell boundary, then by convention we assume that cell, that particular sample will belong to the cell which is to the right hand side.

So, this sample which is 0 will actually fall in this 0th bin whose boundaries are 0 and 2. Similarly, this will also fall in 0th bin, the sample value 1 that will also fall in this 0th bin. If you look, go through this layers do find that there is no other sample which is falling within this 0th bin. So, the number of samples which falls in the 0th bin is equal to 2, come to the next one 2, 2, 2.5, 3, 3.6, 3 again all these samples 2, 2, 2.5, 3, 3.6 and 3 all these samples they fall under this bin having the bin boundaries 2 and 4. So, the number of samples which falls within these bin is 1 2 3 4 5 and 6.

So, there are 6 samples which are falling within this bin. The next 1 within the next bin having the bin boundaries at 4, at 4 and 6 have samples 4 5 and 4 again. So, there are 3 samples which are falling within this bin. Next, 7 this particular sample, the sample 7 falls within the bin having the bin boundaries 6 and 8. And the next 2 samples 9 and 9.2 these 2 samples fall in the bin having the bin boundaries 8 and 10.

So, I have 2 samples for falling in this bin. So now, if I compute the areas you find that I call this to be a 0, this to be area 1, this to be area 2, this to be area 3 and this one having area 4.

So, value of a 0 will be equal to the number of samples in this 0 bin is equal to 2. And I have total number of samples which is equal to 14. So, this area of, this under the A_0 th bin will be equal to $2/14$. Similarly, A_1 that will be equal to $6/14$, A_2 will be $3/14$, A_3 will be $7/14$, sorry here I had only 1 sample not 7 sample.

So, it will be $1/14$ and A_4 will be $2/14$. And from this I can compute what is the probability density in different bins because that is nothing but the area divided by the width. And I have to assume that every bin has the same width is equal to 2. So, if I divide $A_0/2$, which is equal to width, I get what is the probability density estimate in the 0th bin. This divided by 2 gives me what is the probability density estimate in the first bin and so on.

So, if I compute this what I will have is say a 0 will have this is my h_0 which is nothing but $2/14$ into 2 that is $1/14$. Similarly, this will be $6/14$, 14 into 2 that will be the probability

estimate of this. So, I will get it a value something of this form over here it is 3. So, I will get the value of this form, here it is 1. So, I get a value of this form and here it is 2 again. So, I get a value of this form.

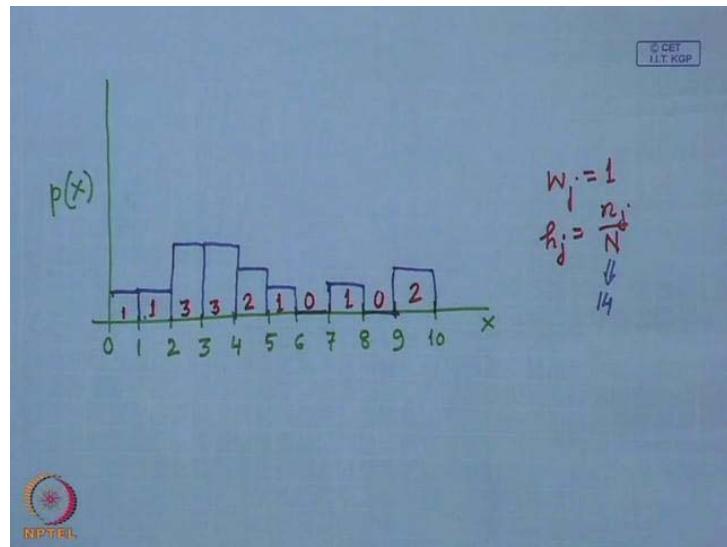
So, you find that given this set of samples and assuming that I have bin size equal to 2 where every bin of same size, the probability density estimate is given by a piecewise constant approximation which is this one. So, this my probability density curve. Now, if you are given a value of x an unknown x . So, I take any value say x_1 is equal to say 4.5 and I have to find out what is $P(x_1)$?

So, here you find that x_1 was value of 4.5 in which bin in this x_1 falls. And if you look at this diagram, this bar graph you find that 4.5 actually falls on this bin. So, 4.5 will be some over here. So, the probability of this x_1 is given by the height of this. So, which is nothing but $3/2*14$. This area is $3/14$, I have to divide this by 2 because width is 2. So, that gives me this height which is nothing but the probability density.

So far as these particular bin is constant and as x_1 falls within this bin, so the probability estimate $P(x_1)$ has to be equal $3/2*14$ which is nothing but $3/28$. So, this is how I can estimate the probability density function, for a variable from a given set of samples for that particular variable. Now, having done this, the other problem that one has to face is that what should be the width of the bins. Depending upon the width of the bins the number of bins will be different because I have a fixed range of the variable x and in this example what you said is the minimum value of x is 0 and the maximum value of x is 10.

So, if I reduce the bin width, instead of taking the bin width to be equal to 2, if I reduce the bin width to 1, I will have more number of bins. On the other hand, from 2, if increase the bin width to say 5, I will have only 2 bins or say bin width say 4, I will have lesser number of bins. So, let us now try to see that, if I vary the bin width, if I vary the number of bins in which I want to divide in the range of x , what effect it is going to have on the probability density estimate. So, I will take the same example and try it with different bin widths.

(Refer Slide Time: 21:39)



So, first let see that what will be the situation if I assume the bin width is equal to 1. So, this is x this is where on the vertical x is I want to put the estimated probability density which is p of x . And this is divided into bins of width 1. So, 1 2 3 4 5 6 7 8 9 and 10. So, here it is 0, here it is 10, 1 2 3 4 5 6 7 8 9 10. Now, given this bins let us see and I take the same set of samples which I have taken for the earlier example, that is 0, 1, 2, 2, 2.5, 3, 3.6, 3 again 4, 5, 4, 7, 9 and 9.2. So, give him a set of samples, you find that this sample 0 falls within the first bin and this is the only sample which is, which falls in this bin.

So, the number of samples which are falling in this bin is equal to 1. Similarly, the next sample 1, that also falls in this bin. So, the number of samples falling in this bin is equal to 1, then 2 which is on the boundary. So, I will assume that falls in the next bin. So, the next bin contains 1 2 2 and 2.5. So, these are the samples which are contained in the next bin. So, next bin contains 3 samples. Then comes 3 3.6 and 3 again. So, these are the 3 samples which are falling in the bin having the bin boundaries 3 and 4. So, I will have 3 samples over here again.

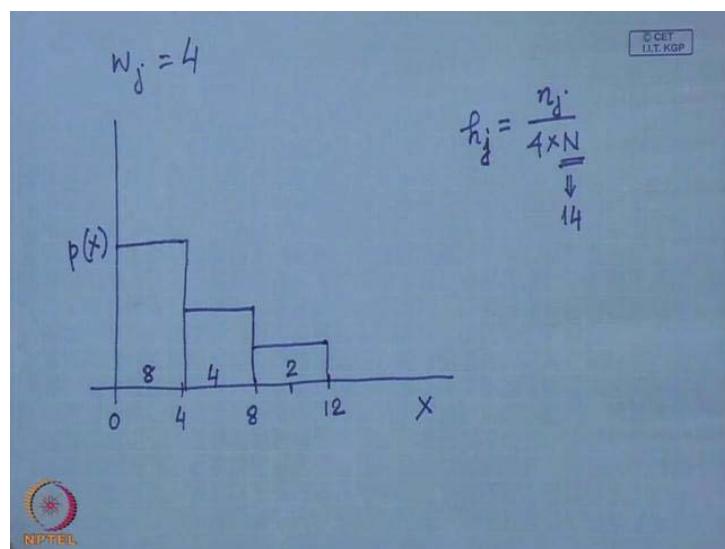
Then these 4 of the 2 samples which are falling in this bin, 5 goes to the next bin. So, a number of samples that I have in this bin, have been bin boundaries 4 and 5 is equal to 2. Whereas, the bin having bin boundaries 5 and 6 contains this sample value of 5. So, this contains only 1 sample, this bin having bin boundary of 6 and 7, you find that no sample is falling in that particular bin.

So, number of samples falling in these bin is equal to 0. This sample 7 it falls in the next bin having bin boundary is 7 and 8. So, the number of samples that I have over here is equal to 1. Again 8 and 9 this does not contain any sample. The next one 9 and 10 this contains 2 samples one is 9 other one is 9.2. So, the number of samples that I have over here is equal to 2. So, naturally given this sort of situation. You find that w_j that is the width of the bin is equal to 1.

So, what I will have is h_j , that is height of the probability density function or the probability density estimate will be simply is equal to $\frac{n_j}{N}$ Where N is the total number of samples as we said before n_j is the number of samples falling in that bin. So, if you follow this you find that this one within this 0 at bin I have a probability estimate which is nothing but 1/14 because one sample is falling in this bin. The total number of samples are capital N, that I have this is equal to 14. So, this will be 1 by 14, this will be 1/14 again, this will be 3/14. Here it will be 3/14 again, here it will be 2/14, here it will be 1/14, here no sample is falling with in this bin. So, value of n_j is going to 0.

So, this is 0 here again I have 1 bin only 1 sample here again it is 0 here I have 2 samples. So, this is the kind of probability estimate that I will get against this. So, if I reduce the bin width or if I increase the number bins, then the kind of situation that I have is I have is as shown here. Now, let us see the reverse case, if I increase the bin width or reduce the number of bins then what I will have.

(Refer Slide Time: 27:12)



So now, let me assume that I have bin width or w_j which is equal to say 4. That is same for all j . So, the situation that I have is something like this 0 4 8. And the next bin boundary is 12, so I have only 3 bins, but my actual range of x is up to 10. So on this axis I plot x and this axis I plot $p(x)$ or the probability of x . Again I use the same set of samples for estimating the probability density function. So, these are the samples I have. So, first I have to see, that out of this samples how many samples are falling within the bin 0 to 10. That means the values of x which are greater than or equal to 0, but less than 4.

And you find that all these samples they have values greater than or equal to 0, but less than 4, so I have 1 2 3 4 5 6 7 8, 8 samples which are falling within this bin. Next from 4 to 8.

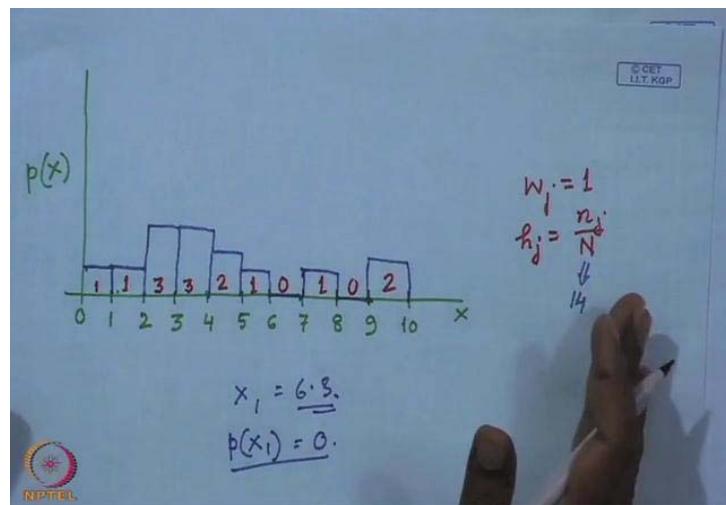
So, all the samples having values greater than or equal to 4, but less than 8 they will fall within this bin. So, here you find that I have these samples 4 5 4 and 7. So, 4 samples are falling within this bin. And the remaining 2 that is 9 and 9.2, these 2 samples they are falling within this bin.

So, if I use this, now you will find that here $w_j = 4$. So, height of every bin h_j will simply be $\frac{n_j}{N}$, where this $N = 14$. I have to total 14 samples. So, for the first one it will be $8/4*14$. So,

I will have something like this for the next one it will be $4/4*14$. So, the probability density estimate will be something like this and the next 1 will be $2/4*14$. So, the probability density estimate will be something like this. So if I use bin size of 2, this is the probability density estimate.

If I use bin size of 1, then this is the probability density estimate. If I use bin size of 4, then this what is the probability density estimate. So, you find all these different cases the probability density estimate that I get is different. Then the question comes what should be the proper value of bin width or what should be the proper numbers of bins, because apparently, it appears that this is, what I should have from these given set of samples.

(Refer Slide Time: 31:01)

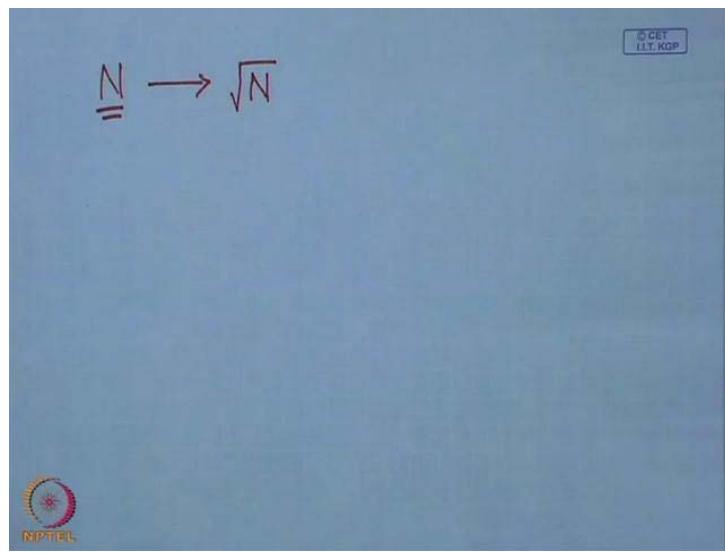


If I use this then this over fit because if I have a unknown sample say x_1 , which is equal to 6.3 and I compute this $P(x_1)$ as this 6.3 falls within this bin for which the probability density estimate was 0. So, p of x_1 will always be equal to 0 as per this. So, this is an over fit given the set of samples. Whereas, if I use a larger bin size, that is the situation of this form, then what I am going for is, I am going for more and more uniform density estimation. So, the bin size is even larger it will be more and more flat. So, as a result the certain details in the probability density estimate as seen over here will be lost.

So, the choice of the number of bins or the size of every bin should be such, that the bin width is not too large. If I use too large bin width will lose the detailed information. On the other hand, if the bin width is too small or I have large number of bins, then it is quite possible that many of the bins will be empty. That is none of the samples, the given samples will fall in many other bins. As a result, the probability density estimate for those bins will be equal to 0.

So, this bin width or the number of bins has to be properly chosen, but unfortunately there is no analytical method of choosing the number of bins or the bin width. So, there is a thumb rule which says that, if I have capital N number of samples.

(Refer Slide Time: 33:25)



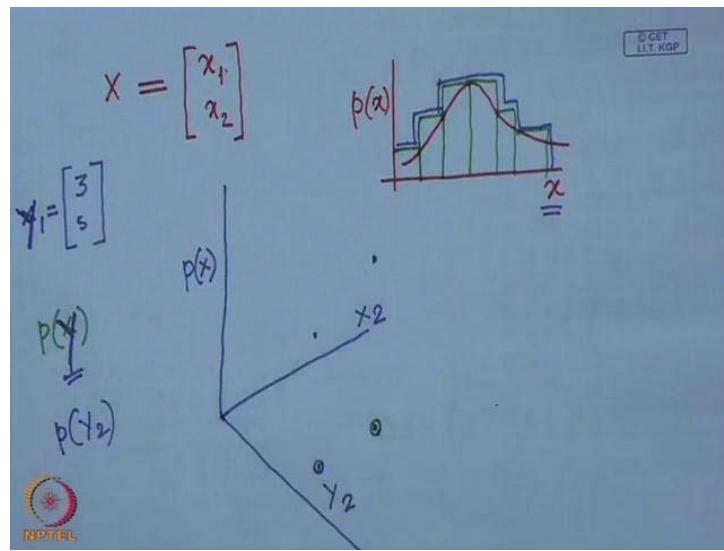
If I have capital N numbers of samples then the number of bins should be near about $\text{sq } \sqrt{N}$. This is just a thumb rule, but there are no analytical tools to decide what should be the number of bins for proper probability density estimation. So, this is the case that we have

done in case of one dimension. That is assuming that our, we have a single variable x and we want to estimate the probability density for that single variate case, but when we go for pattern recognition.

Earlier we have said that in case of pattern recognition, we normally do not deal with a single variable, but we deal with vectors which are called features vectors, where every element in the feature vectors gives some property of the pattern or some information of the pattern, and when I take all the information given by all the elements in the feature vectors together. So, all that information together gives me a representation of the pattern. So, in our case or similarly, many other applications the probability density that we need to consider is not a single variable probability density, but it is a multi variate probability density. So, what we have to use is a multi variate probability density estimation. Let us see that how this probability density estimation technique for a single variable can be extended to multi variate. If I have a vector space having multiple

number of, vectors having multiple number of elements, how this technique can be extended to multi variate case or if I have a vectors spaces?

(Refer Slide Time: 35:27)



So, I will start with a simpler case where I assume that I have two dimensional feature vectors. That means now this x instead of being a single variable, it is a vector which I represent by capital X and this vector has got two components, one I say x_1 , the other one is

x_2 . So, this is the vector and when I take different samples for estimating the probability density what I will have is different sampled values of x that is this vector. Where every sample will have a corresponding sampled values of x_1 and corresponding sample values of

x_2 . So, I will have different instances of these vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

Now, you find that unlike in case of one dimension, where this probability density function was a curve. So, this was my x this was my $p(x)$, if I go for continuous probability density I get a curve something like this, and if I go for this approximation I have piece wise constant approximation of this probability density estimate which is something like this and so on. Where effectively my probability density curve is this one.

So, find that in one dimensional case, the probability density function is represented by curve. So, in case of one dimension, this probability density function is represented by a curve. So, find that if I extend this to two dimension where I have to two different axes. One axis is representing this component variable x_1 and the other axis is representing component variable x_2 .

So, I have a two dimensional space or a plane. So, if I represent, that by coordinate system I have this axis is representing component x_1 and I have this axis is representing the component x_2 . And any vector having a component value x_1 and a component value x_2 . Say for example,

if I take vectors a $\begin{bmatrix} 3 \\ 5 \end{bmatrix}$, this is a vector say $x_1 = 3$. So, this $\begin{bmatrix} 3 \\ 5 \end{bmatrix}$, this particular vector is

nothing but a point in my two dimensional space or two dimensional plane defined by x_1 and

x_2 . So, this $\begin{bmatrix} 3 \\ 5 \end{bmatrix}$ is a vector or is a point some over here. So, this is my point $\begin{bmatrix} 3 \\ 5 \end{bmatrix}$. So, every

vector, every two dimensional vector is nothing but a point in this plane.

Which is defined by x_1, x_2 . So, if I want to now find out what is $p(x)$, that is the probability of vector x . So, the $p(x)$ will be represented on an axis, which is perpendicular to both x_1 and x_2 .

So, what I get is a three dimensional space $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ representing the plain in which all the

vectors will lie. And I have another dimension another direction which is orthogonal to both x_1 and x_2 and that represents my $p(x)$, where x is a vector.

So, $p(x)$ for this particular case will be a point in three dimension somewhere over here. Similarly, if I take a vector over here which is say X_2 or instead of calling them x , let me call

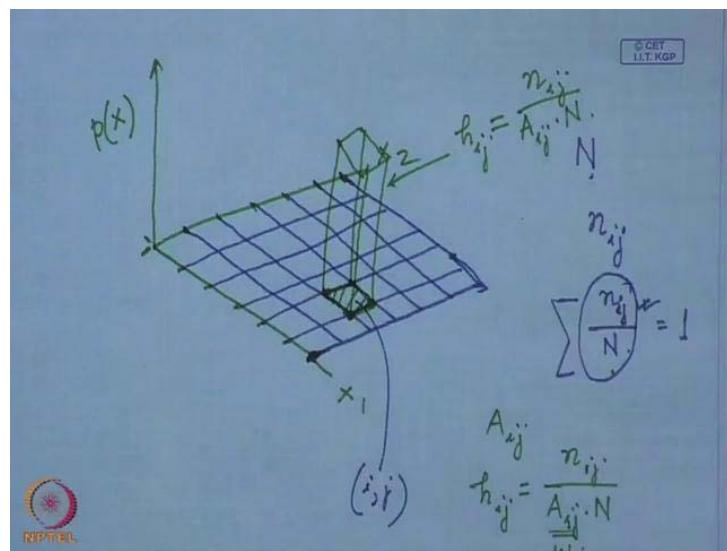
them as Y , to avoid ambiguity. So, my vector is Y , whose components are $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. So, this is

a vector Y_1 instead of calling it X_1 , let us call it as Y_1 . Similarly, if I have a vector Y_2 over here that will $p(Y_2)$ will be another point in this three dimensional space. So, if I join all these points in three dimensional space to represent my probability density function. You find that all these points define a surface in the three dimensional space.

So, when I say the probability density function of a vector, this probability density function is nothing but a surface in a three dimensional space, whereas in case of one dimension for a single variable this probability density function is a curve in a two dimensional space. So, that is the difference from curve we are coming to the surface. So, for probability density estimate, as in one dimensional case we have divided the range of x into a number of bins of certain width. In the two dimensional case I have 2 different variables x_1 other one is x_2 these are the two components of my vector X .

So, both x_1 and x_2 , they will also have their ranges. So, x_1 will have a minimum value of x_1 and it will have a maximum value of x_1 . Similarly, x_2 will also have a minimum value of x_2 and it will have a maximum value of x_2 , which defines what is the range of x_1 and what is range of x_2 ?

(Refer Slide Time: 42:16)



So, once I have that, then this x_1 , x_2 plane, let us put it like this x_1 , this x_2 . So, this is, let us assume that this is the maximum value of x_1 and this is the minimum value of x_1 . So, it is 0 to maximum. Similarly, this is the minimum value of x_2 this somewhere over here we have the maximum value of x_2 . So, which defines what is the range of x_2 ? So, if I divide this x_1 into a number of bins as we have done in case of one dimension. Similarly, x_2 is also divided into a number of bins as we have done in one dimension.

So, find that effectively what we are doing is, this limited space, bounded space, which defines the range of x_1 x_2 because I cannot have any value of x_1 which is greater than this, I cannot of any value of x_2 is greater than this, I cannot any value of x_1 which is less than this I cannot have value of x_2 which is less than this.

So, this defines a bounded space or bounded area and when I break this x_1 and x_2 into a number of bins, effectively what I am doing is this bounded area is divided into a number of rectangular size. So, the cells that I have will be rectangular in nature something of this form. So, these are the cells that I have. And you find that if I go to this one dimension, what the cells were linear in nature. So, these were the cells in one dimension which are linear or line segments and the cells in two dimension there will be rectangular cells or rectangular bins something like this.

So, as you have done in one dimension here what I will do is I will take a cell say $(i, j)^{\text{th}}$ cell because it is in two dimension. So, I have to have 2 indices unlike a single index in one dimensional case. So, I pick up an $(i, j)^{\text{th}}$ cell. As before the total number of samples which have given for probability density estimation, if I assume is capital N and out of this total number of samples suppose n_{ij} is the number of samples which have falling within this $(i, j)^{\text{th}}$

bin. Then the probability density estimate of these will be given by $\frac{n_{ij}}{N}$. and you notice another point that as I said in case of this two dimensional vector the probability density function is given by a surface.

So, if this surface representing the probability density function is an approximation of a continuous probability density. In that case, the volume under this surface here. So, you note the difference in case of one dimension I had a curve. So, I had considered the area under the curve. In this case, in two dimensional case it is a surface. So, I have to consider the volume under these surfaces, suppose this is the surfaces and this is x y plane.

So, I have to find out what is the volume within the surface and this x y plane. So, if this probability density estimate is an approximation of a continuous probability density function. Then the volume under this surface, the surface which represents the probability density function has to be equal to 1. Or in other words what I must have is this some of this has to be equal to 1 and which is obviously true because n_{ij} is the number of samples out of this now which falls in the $(i, j)^{th}$ bin. And this N is the total number of samples. So, if I simply sum

them of overall i and j that has to be equal to 1 i.e. $\sum \frac{n_{ij}}{N} = 1$.

This has to represent volume of the probability density function over this $(i, j)^{th}$ bin. So, you find that in case of two dimension the kind of estimate that we had over here. So, these represents an area, here this represents a volume. So, the probability density estimate, if this is my p of x , where over this $(i, j)^{th}$ bin, the $p(x)$ will be given by a bar something like this.

Where the volume of this bar is given by this $\frac{n_{ij}}{N}$ and if the area of the base of this bar is A_{ij} ,

then the height of this bar h_{ij} will be given by $h_{ij} = \frac{n_{ij}}{A_{ij}N}$

So, if u compare this with our one dimensional case instead of this area A_{ij} what I had was the width of the bin w_j . So, over here this is not the width, but it is the area of the bin (i, j) , this is the area of the $(i, j)^{th}$ bin which is A_{ij} . Now, it is a very simple extension of the probability density estimation technique from one dimension to two dimension. So, you have been given a large number of samples or capital N number of samples. I have to check that out of these capital N number of samples, how many samples are falling in the $(i, j)^{th}$ bin.

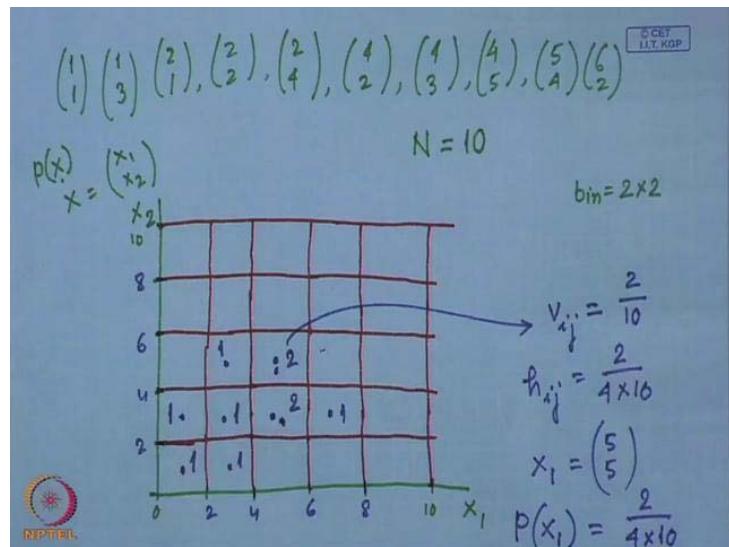
And if I know or obviously I have to know that what is the area of $(i, j)^{th}$ bin, then the probability estimate or h_{ij} will be given by n_{ij} that is the number of samples falling in the $(i, j)^{th}$ bin divided by capital N , $\frac{n_{ij}}{N}$, that is total number of sample divided the area of the $(i, j)^{th}$

bin which is A_{ij} . So, over here the height of this which represents the probability density estimate within this bin, that is h_{ij} is simply equal to $h_{ij} = \frac{n_{ij}}{A_{ij}N}$.

So, this is a simple extension from our one dimensional case to two dimensional case. So, again to explain this further to clarify this concept again I will take an example to explain,

how this probability density estimation in two dimension will work. So, again let us take an example with a number of two dimensional feature vectors.

(Refer Slide Time: 51:14)



So, I have a set of features vectors, let us assume that we have set of features vectors that is

$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ all I in two dimension $\begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \begin{bmatrix} 4 \\ 3 \end{bmatrix}, \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \begin{bmatrix} 5 \\ 4 \end{bmatrix}, \begin{bmatrix} 6 \\ 2 \end{bmatrix}$ and say $\begin{bmatrix} 6 \\ 2 \end{bmatrix}$. So, suppose

these are the two dimensional features vectors that we have. And using these two dimensional feature vectors I have to estimate the probability density function or $p(x)$ where

this X is a vector of the form $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

So, total number of sample vectors that I have is 10. So, my capital N or the total number of samples is equal to 10. So, what I will do is, I will consider a two dimensional features space,

having $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, has two different axes. Here if you analyze if you find out the values of x_1 and

if you find out the values of x_2 , here you find that I can easily say that the range of x_1 is between 0 and 10 and the range of x_2 is also between 0 and 10 because I do not have any value of x_1 the first component which is $10 > x_1 > 0$.

Similarly, for x_2 I do not have any component x_2 whose value is $10 > x_2 > 0$. So, I can very easily assume that the range is between 0 and 10, here also the range is between 0 and 10. And now, let us say that we have the bins, every bin is say of size 2×2 . So, x_1 will be divided into 5 bins, each of length to x_2 is also be divided into 5 bins each of length 2. So, suppose these are the bins. So, once I do this effectively what I am doing is, I am dividing this space into 5×5 , that is 25 rectangular bins.

So, these are the bins in the two dimensional space I have. Now, let us try to see that out of these given samples which samples is falling in which bin. So, if you look at the first x_1 this

is $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ nothing this, but because this of size two. So, this is 2, 4, 6 and 8 and 10, and 2, 4, 6, 8

and 10. So, $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ is the point somewhere over here. So, it is falling with in this bin $\begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ is a point somewhere over here.

So, which is falling with in this bin. Similarly, $\begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ is a point somewhere over here. So,

it is falling within the bin $\begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}$ is this point, but by convention as we said that we

assume that this falls within this $\begin{bmatrix} 2 \\ 4 \end{bmatrix}, \begin{bmatrix} 2 \\ 4 \end{bmatrix}$ again that is this point, and by convention we

assume that this falls on this bin, $\begin{bmatrix} 4 \\ 2 \end{bmatrix}$ which is this, by convention we take that that it falls

within this bin, $\begin{bmatrix} 4 \\ 3 \end{bmatrix}, \begin{bmatrix} 4 \\ 3 \end{bmatrix}$ is somewhere over here by convention I assume that it falls over

here. $\begin{bmatrix} 4 \\ 5 \end{bmatrix}, \begin{bmatrix} 4 \\ 5 \end{bmatrix}$ is somewhere over here by convention I assume that it falls within this bin,

$\begin{bmatrix} 5 \\ 4 \end{bmatrix}$ by convention I assume it falls within this bin.

Then $\begin{bmatrix} 6 \\ 2 \end{bmatrix}$ it is this point I assume it falls in this bin. So, you find that here I have only one

sample, one sample, one sample, one sample, one sample, here I have 2 samples, I have 2 samples, I have one sample over here. So, the volume of the bar over this, if I take this

particular one, volume of the bar, $v_{ij} = 2$ divided by, I have total number of samples which is equal to 10. Height of this bar h_{ij} , which will be simply this divided by area of the base and this bin of size 2×2 , area of the bases is equal to 4. So, $h_{ij} = \frac{2}{4.10}$.

And that is nothing, but probability density estimate with in this bin. Now, find that if I have X_1 which is equal to let us say $\begin{bmatrix} 5 \\ 5 \end{bmatrix}$. So, $\begin{bmatrix} 5 \\ 5 \end{bmatrix}$ falls within this particular bin. So, $p(X_1)$ is nothing, but this $h_{ij} = \frac{2}{4.10}$. So, you find that we have simply extended the concept of probability density estimate from one dimension to two dimension. I will continuous with this further in our next lecture.

Thank you.

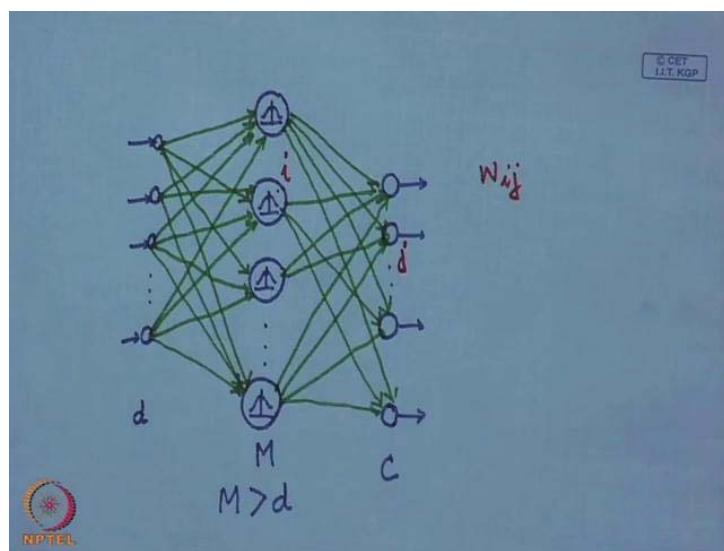
Pattern Recognition and Application
Prof. P.K Biswas
Department of Electrical and Electronics Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 13
Probability Density Estimation (Contd.)

Hello, so in the last class we have started discussion on radial basis function, neutral network and we have seen that a radial basis function neutral network consists of 3 layers. One is of course input layer and one of the three layer contain is the outer layer and in between the input layer and output layer, we have a hidden layer. So, unlike in case of multi-layer perception where can have one or more hidden layers in case of the radial basis function network we have only one hidden layer and every neuron, in the hidden layer computes a radial basis function.

So, when I have the neurons in the hidden layer, for every neuron which computes a radial basis function, radial basis functional value for an input feature vector, every radial basis function has got two parameters. One is called the receptor and other one which defines the spread of the radial basis function, so the architecture that we have is something like this.

(Refer Slide Time: 01:46)



We have one input layer, so the input layer contains a number of neurons and the number of neurons in the input layer is same as the dimensional, dimensionality of the feature vector.

So, if the features vectors are of dimension d , I will have d number of nodes in the input layer. So there will be d number of nodes when the dimensionality of the feature vector is d . In the hidden layer I will have a number of nodes and suppose the number of nodes in the hidden layer is say capital M . So, as we discussed in our previous class that the purpose of the hidden layer nodes is to project that the d dimensional nodes into a higher dimensional node.

So, as I have M number of nodes in the middle layer, so obviously this M the number of node in the hidden layer, in the hidden layer is greater than the dimensionality of the feature vector which is d . As we said that every node in the hidden layer computes of this radial basis function like this. And at the output layers which are basically the classifying rounds, I have the number of neurons or of the number of layer which is the same as the number of classes that we have. So, if we have c number of classes then the output layer I will have c number of neurons, so there we have seen C number of neuron, and C is the number of class in which the patterns are to be classified.

Then every node in the input layers connected, is feeding input to every node in the hidden layer and output of the hidden layer, every node output from the hidden layer is connected to every node in the output layer. So, I have the connections which is something like this, so these are the connection form the input layer nodes to the hidden layer nodes because the propose of this connection is simply to forward the input feature vector to the nodes in the hidden layer, we can assure that weight of this connection is equal to 1, and that is a difference with the connection from the hidden layer to the output layer nodes because in every output layer node computes a linear combination of the outputs of the hidden layer node.

So, the connection from the output layer nodes to the connection from hidden layer nodes to the output layer nodes is something like this. So, where we can see every i^{th} node in the hidden layer is connected to the j^{th} node in the output layer through a connection weight which is equal to W_{ij} . So, because of this, every node in the output layer computes, a linear combination of the outputs of the hidden layer and based on this, the value of the linear combination the output layer nodes decide to which class the input vector should be classified.

Now, what can be done is, these output layer nodes can also impose a nonlinear function, to ensure that if a particular input feature vector belongs to class ω_j . In that case only the output of the j^{th} node will be equal to 1 and output of all other output nodes will be equal to 0, similarly if a feature vector input feature vector belongs to say class 1. Then

only the output of the first node in the output layer will be equal to 1 and outputs of all other nodes in output layer will be equal to 0. So, as we discussed in the previous class that such are radial basis function network an RBF network incorporates two type of learning.

One is we have to learn that for every node in the hidden layer because every node in the hidden layer represents a radial basis function what should be the receptor of that radial basis function. What should be the spread of that radial basis function, so if the radial basis function is a Gaussian function that is if it is something like this.

(Refer Slide Time: 08:27)

$$\phi_i(x) = e^{-\frac{\|x - t_i\|^2}{2\sigma_i^2}}$$

Say $\phi_i(x) = e^{-\frac{\|x - t_i\|^2}{2\sigma_i^2}}$, where t is the receptor and σ which is the variance it decides that what the spread of the radial basis function. So, every for every i^{th} radial basis function $\phi_i(x)$, t_i is the receptor and σ_i is the spread, so I have to know that what is the receptor for every radial basis function, and what is the spread of every radial basis function. So, this is one level of learning and the second level of learning is once through these radial basis functions a d dimensional feature vector is projected onto an M dimensional feature vector.

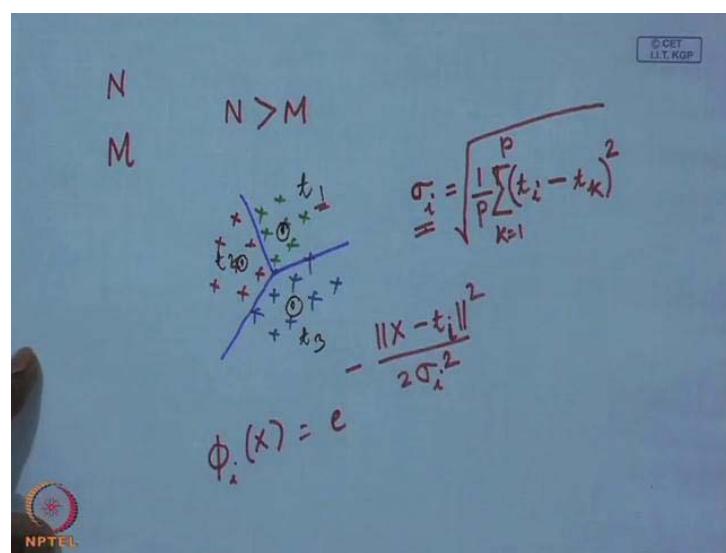
So, basically what we are doing is, we are increasing the dimensionality of the feature vector, and as we have indicated in our last class that the purpose of increasing dimensionality is that if the feature vectors are linearly non separable in the d dimensional space, when we cast them into a high dimensional space, then the possibility that they will be linearly separable in a high dimensional space increases, and this possibility increases with a value of M .

So, as we increase that dimensionality more and more, the possibility of linear separability of the feature vectors also increases. So the feature vectors in d dimensional space which are not linearly separable when I cast them into an M dimensional spaces and $M > d$ it is more likely that those feature vectors will be linearly separable in an M dimensional space. And once feature vectors are linearly separable in the M dimensional space then the linear combination of the outputs of the hidden layer is likely to give me the class belongingness.

And that linear combination is decided by the weight vectors by the connection weights from the hidden layer nodes to the output layer nodes, so we also have to learn that what should be the connection with W_{ij} from say i^{th} node in the hidden layer to j^{th} node in the output layer. So, this is the second level of learning, so in the first level of learning for every radial basis function we try to learn what is the receptor and what is the spread of the radius basis function.

And for, in the second level we try to learn what is the connection weight from the hidden layer node to the output layer nodes and as we have discussed in the previous class that the usual way, a common method of learning. The radial basis of function is if you are given a set of feature for the training purpose and suppose the value of $M = 3$. So, what we do is we partition or we cluster this set of feature vectors into 3 number of clusters, so if we have M number of nodes in the hidden layer.

(Refer Slide Time: 12:26)

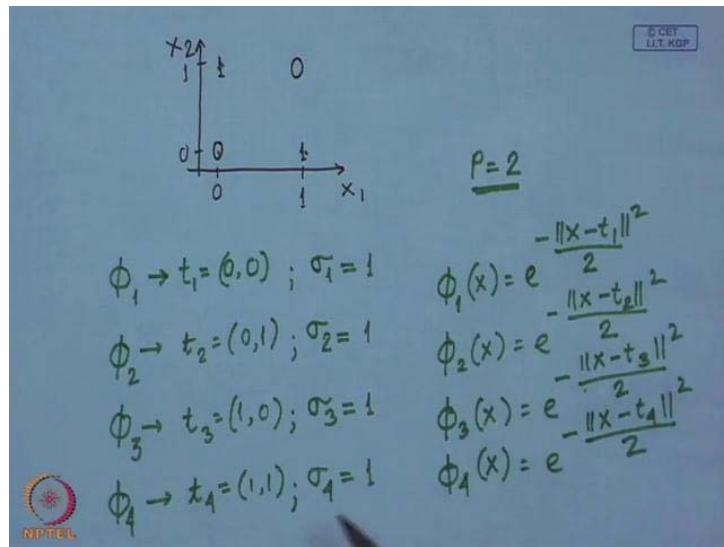


I have a number of N numbers of feature vectors, N number of feature vectors which are given for training purpose and I have M number of nodes in the hidden layer. Obviously, in this case N has to be greater than M, otherwise clustering N number of feature vectors into M number of clusters does not make sense, so I have to have more number of feature vectors than the numbers of clusters that I have. So, I cluster this N number of feature vectors into M number of clusters and I can assume that centroid or mean of every cluster represent the corresponding receptor, okay.

So, if I take i^{th} cluster, the i^{th} cluster represents the receptor, i^{th} radial basis system, so this situation is something like this if I have a set of feature vectors. Say these are the features factors belonging to different classes typically what I do is, I cluster this feature vectors into 3 different clusters every cluster center known represents receptor. So, this is one receptor and this is one receptor, this is one receptor, so this is a receptor t_1 , this receptor t_2 and this receptor t_3 . So, the first operation that we have to perform is the clustering of the feature vector and this clustering operation is we will discuss in detail in feature lectures. Now once I have these different receptors I have to find out what should be spread of a particular radial basis function what you do is for say i^{th} receptor I find out p number of nearest neighbors or p number of nearest receptor. For this p number of receptors, I compute what is the mean distance or root mean square distance, so there are different possibilities. I can take any value, I can choose any value out of this p number of distance values, so what I do is the way I compute σ_i for the i^{th} cluster, for i^{th} radial basis function is I have t_i which is the receptor for the i^{th} radial basis function.

Then I take p number of nearest receptor, which are nearest to t_i , so suppose one such receptor is t_k , so what I do is, I compute $\sigma_i = \sqrt{\frac{1}{p} \sum_{k=1}^p (t_i - t_k)^2}$. Take summation of this for $k = 1$ to p, as I have p receptors, so this defines the spread of the i^{th} radial basis function. So, for every i^{th} radial basis function, t_i and σ_i and once these two are known then radial basis function $\phi_i(x) = e^{-\frac{\|x-t_i\|}{2\sigma_i^2}}$, okay. Now let us see that by using this concept, whether I can make linear classifier using the radial basis function concept for the XOR problem and XOR is a very, very common problem, which is used for illustrating such operation, so as we have said earlier.

(Refer Slide Time: 17:18)



If I take an XOR function, I have a 2-dimensional feature vector, binary feature vector, having components X_1 and X_2 , suppose this represent 0 this is $X_1 = 1$, here I have $X_2 = 0$. Here, I have $X_2 = 1$, the value of the XOR function when it is $(0, 0)$ is 0, $(0, 1)$ the value is 1, $(1, 0)$, the value is 1 and $(1, 1)$ again the value is 0. So, you find that here I have two dimensional, binary feature factors and what I do is this 2 dimensional feature vector I want to cast into a 4 dimensional space by using four radial basis functions.

So, I have the radial basis functions ϕ_1 , ϕ_2 , ϕ_3 and ϕ_4 , for ϕ_1 , I choose t_1 is equal to say $(0, 0)$, that is the receptor of the radial basis function ϕ_1 . Similarly, for ϕ_2 , I can choose t_2 which is say $(0, 1)$ that is the receptor of the radial basis function ϕ_2 , similarly the receptors of other radial basis function I can choose as $t_3 = (1, 0)$. And for this I choose $t_4 = (1, 1)$, so these are the 4 receptors for 4 the radial basis functions, next I have to choose the spread, σ_1 for the first radial basis function.

I have to choose σ_2 for the second radial basis function, σ_3 for the third radial basis function and σ_4 for the fourth radial basis function. now for these for every receptor I have to find out p number of nearest receptors. And Suppose I choose the value of $p = 2$ now, here you find that for every receptor there are 3 neighbor 2 of the neighbors at a distance at 1 and one of the neighbors is at a distance of 1.4 that is $\sqrt{2}$. So, that is easily verifiable from here I have receptor over here which is t_1 , t_2 is at a distance of 1, t_3 is at a distance of 1, but t_4 is at distance of $\sqrt{2}$ which is 1.4 or 1.414.

So, when I take $p = 2$, I have to take two nearest neighbors both of them are at distance 1 and root means square distance of these 2 distances will also be equal to 1. So, I have $\sigma_1 = 1$, I spread also $\sigma_2 = 1$, I have spread $\sigma_3 = 1$ and I have spread $\sigma_4 = 1$. So, I get $\phi_1(x)$ which is of

the form, $\phi_1(x) = e^{\frac{-\|x-t_1\|}{2\sigma_1^2}}$ and that $\sigma_1 = 1$, this will be equal to 2.

Similarly, for $\phi_2(x)$, I will have $\phi_2(x) = e^{\frac{-\|x-t_2\|}{2}}$, $\phi_3(x) = e^{\frac{-\|x-t_3\|}{2}}$ and ϕ_4 . So, if I compute these values for each of the feature vectors taking $(0, 0)$ is one of the feature vector, $(0, 1)$ as other feature factor, $(1, 0)$ as another feature factor and $(1, 1)$ as another factor vector, the functional values will be something like this.

(Refer Slide Time: 22:43)

Input	ϕ_1	ϕ_2	ϕ_3	ϕ_4	$\sum w_i \phi_i$	Output
0 0	1.0	0.6	0.6	0.4	-0.2	0
0 1	0.6	1.0	0.4	0.6	0.2	1
1 0	0.6	0.4	1.0	0.6	0.2	1
1 1	0.4	0.6	0.6	1.0	-0.2	0
	-1	+1	+1	-1		

So, I put that in the form of a table, here I have input feature vectors inputs are $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$ and I have the RBF function ϕ_1 , ϕ_2 , ϕ_3 and ϕ_4 . So, when you input the feature vector $(0, 0)$ to ϕ_1 you find that your X is equal to t_1 , so these exponent is equal to 0 which means $\phi_1 = 1$. So, this ϕ_1 is over here this will be 1.0, similarly for ϕ_2 my X is $(0, 0)$, where t_2 is $(0, 1)$, okay, so if I compute these ϕ_2 you will find this $\phi_2 = 0.6$.

Similarly, I will put just values over here ϕ_3 will also be 0.6 and ϕ_4 is will be 0.4. when the input vector is $(0, 1)$, ϕ_1 will be 0.6, ϕ_2 will be 1.0, ϕ_3 will be 0.4, ϕ_4 will be 0.6. For $(1, 0)$ this is 0.6, this is 0.4, this is 1.0, this is 0.6 again and for the input vector $(1, 1)$, I have ϕ_1 is equal to 0.4, ϕ_2 will be 0.6, ϕ_3 will be 0.6 and ϕ_4 is that will be 1.0. So, you find that given our 2

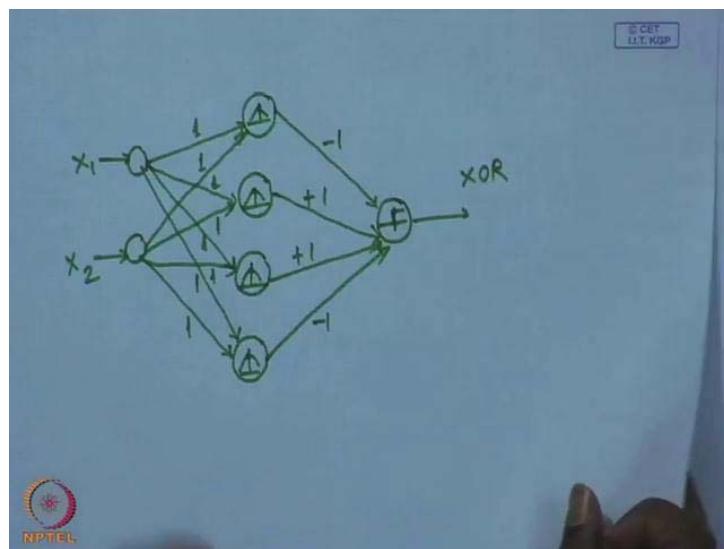
dimensional feature vector 0 0, this has been cast into a 4 dimensional factor vector where the components of this 4 dimensional feature vector are 1.0, 0.6, 0.6 and 0.4.

Similarly, (0 1) this is a 2 dimensional input feature vector which has been cast into a 4 dimensional vector the components being 0.6, 1.0, 0.4 and 0.6. So, every input feature vector where the 2 dimensional feature vector, every 2 dimensional input vector is converted to a 4 dimensional feature vector by using 4 radial basis functions. So, if I take the linear combination of this and for linear combination for ϕ_1 , if I give an weight of 1, for ϕ_2 , I give an weight of -1. For ϕ_3 , I give an weight of +1, for ϕ_4 , I give an weight of -1, okay.

So, the function that I finally output at the output are node in the output layer will be $\phi_2 + \phi_3 - \phi_1 - \phi_4$ and if I compute this let us see what are the values that I get. So, here I will write sum of $W_i \cdot \phi_i$, where i varies from 1 to 4, so here it will be $0.6 + 0.6$ is 1.2 - 1.4 this will be -0.2, similarly here it will be 1.4 - 1.2, so it will be + 0.2.

Here, again it will be 1.2 - 1.4, so this is -0.2 and if I take a discussion that if the value is more than 0 the output will 1, if it is less than 0 the output will be 0, then the final output that we have is. Here, I write output this will be 0, this will be 1, this will be 1 and this will be 0, so which is nothing but the XOR function output, so over here the architecture of the radial basis function that we have used is.

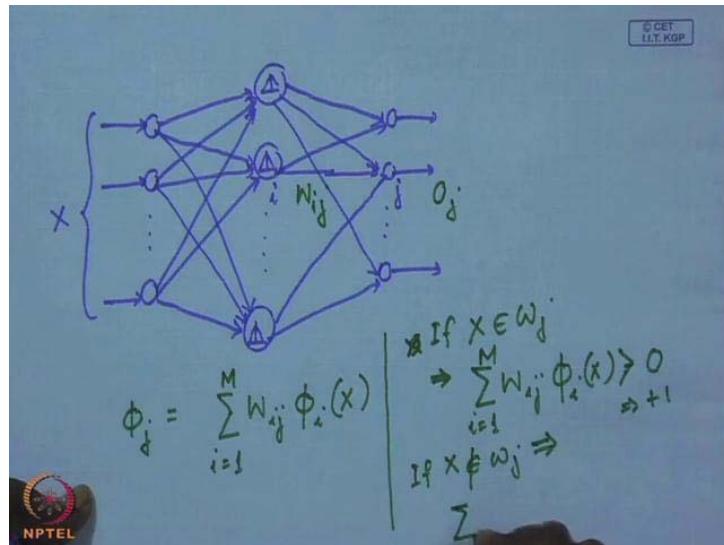
(Refer Slide Time: 28:08)



We had two input layer nodes where X_1 is feed to one node, X_2 is feed to another node, I had 4 nodes in the hidden layer which computes the radial basis function and I had 1 node in output layer which I can say that it is finding out. It is a nonlinear operator or a threshold operator the connections are like this, where each of this connection have a connection weight is equal to 1 and over here these connections. As you can see over here ϕ_1 to output layer of node 1 has a connection weight of -1, ϕ_2 to output layer node has connection of +1, ϕ_3 to output layer node has connection of +1, ϕ_4 to output layer node has connection weight of -1.

So, here the connection weights are -1, +1, +1, -1 and this output actually gives me the XOR function, so this example clearly shows that by casting the 2 dimensional feature vectors into 4 dimensional feature vector I can implement the XOR function using a linear network or a single layer preceptor because this part is nothing but a single layer preceptor, so now let us theoretically try to find out. Try to find an expression for the training of the output layer or how do I find out this connection weights, so in general I have a network something like this.

(Refer Slide Time: 31:02)

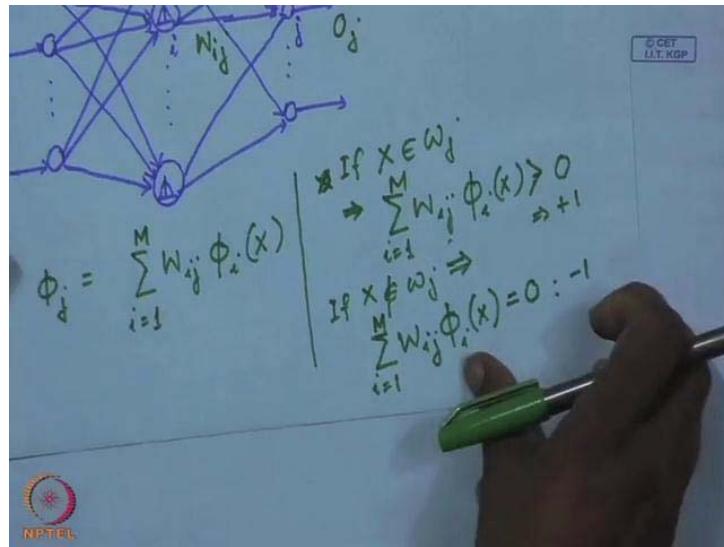


I have a set of input layers, set of input layer nodes, I have a set of hidden layer nodes and I have a set of output layer nodes, the feature vector is fed to the input layer nodes. So, here I fed X which are fed to the hidden layer nodes through connection weights, which are 1 and outputs of this hidden layer nodes, these are my radial basis functions outputs of the hidden layer nodes are connected to the output layer nodes, so like this and I take the output from every output layer node. So, if my input feature vector X belongs to say i^{th} class then output of

the i^{th} output node will have a high value likely to be 1 and outputs of all other output layer nodes will have a low value likely to be 0.

And I assume that the i^{th} node in the input layer is connected to the j^{th} node of the output layer through a connection weight say W_{ij} . So, given this if I say the output of j^{th} node is O_j , I will have $O_j = \sum_{i=1}^M W_{ij} \phi_i(x)$, for an input vector x and this summation, I have to compute over all nodes in the hidden layer. So, here I will have this summation has to be computed over $i = 1$ to M , as I have M number of nodes in the hidden layer and naturally over here if this feature vector X . So, I will write if X belongs to class ω_j then I have to have $O_j = \sum_{i=1}^M W_{ij} \phi_i(x)$, $i = 1$ to M this must be greater than 0, I will put this as +1 and if X does not belong to ω_j that indicates that sum of

(Refer Slide Time: 34:41)



$O_j = \sum_{i=1}^M W_{ij} \phi_i(x)$, i varying from 1 to M , that must be equal to 0 or I can also put it as -1, so

let us assume that if X belongs to class ω_j , $O_j = \sum_{i=1}^M W_{ij} \phi_i(x)$ that has to be equal to +1. If X does not belong to ω_j this has to be 0 and that is what has to be the output from the j^{th} node in the output layer, now taking this. Now, I can go for training of the output layer that means I have to find out what should be the values of this W_{ij} . Now, if I compute only the connections

weights, if I right now consider only the connection weights which are connected to the j^{th} node in the output layer then for every vector X_k , suppose I have capital N number of vectors.

(Refer Slide Time: 35:57)

$$x_k; k=1 \dots N$$

$$\phi_i(x_k) \rightarrow \phi_{ik}$$

$$\sum_{i=1}^M w_{ij} \phi_{ik} = +1 \text{ if } x_k \in \omega_j$$

$$= 0 \text{ if } x_k \notin \omega_j$$

Say I have vectors X_k for k varying from 1 to capital N , I have capital N number of input vectors which are given for training purpose or for learning as we are using supervised learning then $\phi(X_k)$, for simplicity i will write this as ϕ_{ik} . Now, by using this I can, as I said

that sum of $\sum_{i=1}^M \phi_i(X_k) \cdot W_{ij}$, so my condition is if you remember this one if you remember this one.

So, $\sum_{i=1}^M \phi_i(X_k) \cdot W_{ij}$, for i varying from 1 to M , this has to be equal to +1, if X_k belongs to class ω_j , and this has to be 0 if X_k does not belong to ω_j . So, this is the output that I expect, so for X_k , I have for every X_k , I have such a kind of linear equation that this summation will be either +1 or 0 and all those capital N number of equations, now I can write in the form of a matrix, so in the matrix form this can be written as.

(Refer Slide Time: 38:02)

$$\begin{bmatrix}
 \phi_{11} & \phi_{21} & \phi_{31} & \cdots & \phi_{M1} \\
 \phi_{12} & \phi_{22} & \phi_{32} & \cdots & \phi_{M2} \\
 \vdots & & & & \\
 \phi_{1N} & \phi_{2N} & \phi_{3N} & \cdots & \phi_{MN}
 \end{bmatrix}
 \begin{bmatrix}
 W_{1j} \\
 W_{2j} \\
 \vdots \\
 W_{Mj}
 \end{bmatrix}
 =
 \begin{bmatrix}
 b_{1j} \\
 b_{2j} \\
 \vdots \\
 b_{Nj}
 \end{bmatrix}$$

$b_{ij} = 1$ if $x_i \in \omega_j$
 $= 0$ if $x_i \notin \omega_j$

Let me write the matrix equation that ϕ_{11} which means $\phi_1(X_1)$, okay, ϕ_{21} that is $\phi_2(X_1)$, ϕ_{31} , ϕ_{M1} , this means $\phi_M(X_1)$. Similarly, ϕ_{12} which means $\phi_1(X_2)$, ϕ_{22} , ϕ_{32} up to ϕ_{M2} and as I have capital N number of samples for training, so I will have ϕ_{1N} , ϕ_{2N} , ϕ_{3N} up to ϕ_{MN} which indicates $\phi_M(X_N)$ into W_{1j} , W_{2j} up to W_{Mj} . So, you find the what it computes $W_{1j} \cdot \phi_{11} + W_{2j} \cdot \phi_{11} + \dots + W_{Mj} \cdot \phi_{M1}$, that is for the first input vector X_1 .

Whatever is the output of individual middle layer nodes or hidden layer nodes this equation simply makes a linear combination of outputs the hidden layer nodes for the input feature vector, which is 1 or X_1 , okay. So, these has to be equal to again I output in form of vector b_{1j} , b_{2j} up to b_{Nj} , where every $b_{ij} = 1$ if the corresponding X_i belongs to class ω_j and that will be equal to 0 if the corresponding X_i does not belong to 1 class ω_j .

So, every b_{ij} will assume a binary value either 0 or 1, so this $b_{ij} = 1$, if X_i the corresponding input vector X_i belongs to class ω_j , the j^{th} class or it will be equal to 0 if X_i does not belong to ω_j , so this the kind of situation I have and these whole expression, this matrix equation I can write in a short form.

(Refer Slide Time: 41:38)

$\phi W_j = b_j$
 $e = \phi W_j - b_j$
 $J(W_j) = \| \phi W_j - b_j \|^2$
 $\nabla J(W_j) = 2 \phi^t (\phi W_j - b_j)$
 $W_j = (\phi^t \phi)^{-1} \phi^t b_j$

That is $\phi W_j = b_j$ where this ϕ is this matrix and W_j is this weight vectors which are connected to the output layer node j and b_j is the output of the j node in the output layer which is represented in the vector like this for different input vectors, okay. So, if the network is properly trained that is all W_{ij} has got the trained value then this equation should be satisfied. But, what we are trying to do is, we are trying to train the networks we are to set the weights W_j , so it cannot expect that this equation will be satisfied initially. So, if the equation is not, if this equality is not satisfied then what I can do is, I can define a error e , which is nothing but $e = \phi W_j - b_j$ and, now training involve adaptation of this weight W_j .

So, that this error can be minimized see in order to do that as we have done earlier for mean square error optimization for mean square error technique for classifier learning or classifier training I can also define. Here, a criteria function $J(W_j)$ which is given by

$J(W_j) = \| \phi W_j - b_j \|^2$ and then I take gradient to with respected W_j , so $\nabla J(W_j)$ which will be simply $2\phi^t(\phi W_j - b_j)$. And by equating this to 0, what we get is $W_j = (\phi^t \phi)^{-1} \phi^t b_j$, and as we have seen earlier this, $(\phi^t \phi)^{-1} \phi^t$.

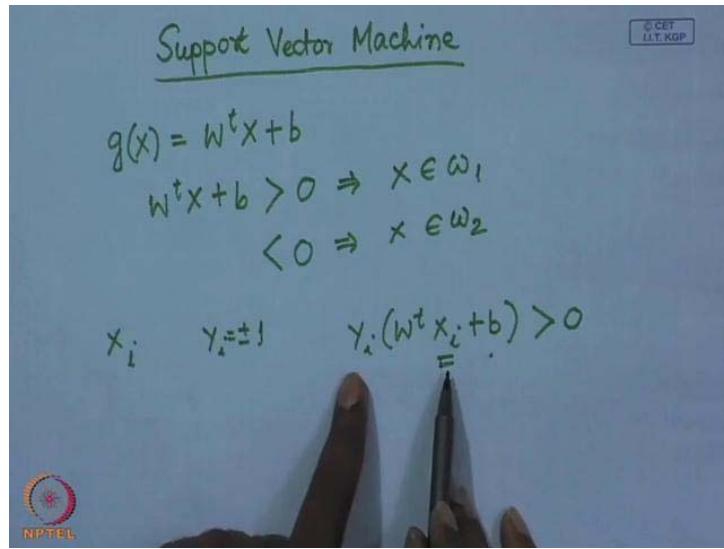
This is what is called pseudo inverse and that is represented as ϕ^+ so we have W_j by this pseudo inverse technique, we have this $W_j = \phi^+ b_j$, where b_j is defined beforehand every component of b_j will be either 1 or 0. It will be equal to 1 if the corresponding feature vector, input feature vector belongs

to class ω_j and component will be equal to 0 if the corresponding that does not belong to class ω_j . So, I have this vector b_j , actually indicates that what should be output of the hidden layered nodes for every feature vector from that I compute what is matrix ϕ . So, once I have matrix ϕ and I have this b_j , I can compute what will be the connection for different nodes in the hidden layer to the j^{th} output layer node.

This if I do for every output layer node, I can compute what is the connection weight from different outputs of the hidden layer nodes to different output layer nodes. And that is what completes my training of the RBF neural network and the RBF neural network will be ready for classification, know if you compare this RBF neural network with your against say multi-layer perceptron, you will find that the training of the R B F neural network is faster than the training in multi-layer perceptron because in case of multi-layer perceptron, the training is done by back propagation algorithm which takes large number of iterations, okay. So, the training of the R B F neuron will be faster than training of the multi-layer perceptron. The second advantage is that I can easily interpret what is the meaning or what is the function of every node in hidden layer, okay, which is difficult in case of multiply layer perception, I cannot easily interpret the role of different nodes in the hidden layer in case of multi-layer perception and not only that I also cannot easily decide that what should the number of hidden layers and what should be the number of hidden nodes in every hidden layer, so those are the difficulties in the multi-layer perceptron which is not there in case of RBF network, however the RBF network as a disadvantage that though the training is faster.

But, you find that the classification takes more time in case of RBF network then in case of MLP because in case RBF network every node in the he hidden layer has to compute the radial basis functional value for the input feature vector, which is time consuming. So, the classification in case of classification RBF network takes more time than the classification time in case of multi-layer perceptron, okay, so with is we come to a conclusion on the radial basis function neural network, know over here I will just briefly discuss about another kind of classifier which is called a support vector machine.

(Refer Slide Time: 48:37)



So, I will briefly discuss support vector machine, so support vector machine is another type of linear classifier, so if you remember what we discussed in case of a linear classifier that given a two class problem. We have said that I can define a discriminating function say $g(X)$ which is of the form say $g(X) = W^T X + b$, okay and we have said in case of linear discriminator. If this $g(X)$ or $W^T X + b$, this is greater than 0 that indicates say feature vector X belongs to class ω_1 , and if this is less than 0 then feature vector X belongs to class ω_2 .

So, you here find for classification purpose the actual value of $g(X)$ is not really very, very important, but what is important what is the sign of $g(X)$, if the sign is positive, I infer that X belongs to class ω_1 . If the sign is negative I infer that X belongs to class ω_2 , so over here if with every X_i , I indicate a number Y_i , Y_i can be either +1 or -1. In that case, this $Y_i(W^T X + b)$ it will be always greater 0 if the sample X_i is properly classified, which quite obvious because if I say that Y_i is equal to +1 for a sample X_i which belongs to class ω_1 .

And for a sample which is belongs to ω_1 this $W^T X + b > 0$, Y_i is also positive, so Y_i times this will obviously greater than 0 if X_i belongs to ω_2 then W transpose X_i , it will be less than 0 and for that I set $Y_i = -1$. So, $-1(W^T X + b)$ will obviously greater than 0 and this is a concept that actually we have used when we have discussed about on the perceptron criteria or design the linear classifier.

That is for every feature vector belonging to class ω_2 , we have negated the feature vector before we try to design the classifier.

So, that every feature vector irrespective of whether the feature vector belongs to plus ω_1 or the feature vector belong to class ω_2 my discriminate function value will always be positive, if the feature vector is correctly classified. So, that is true if the feature vector belongs to class ω_1 or even if the vector belongs to vector because the feature vector belonging to ω_2 before trying to design the classifier. We have negated the feature vector, so we will discuss about this support vector machine more in details in our next class.

Thank you.

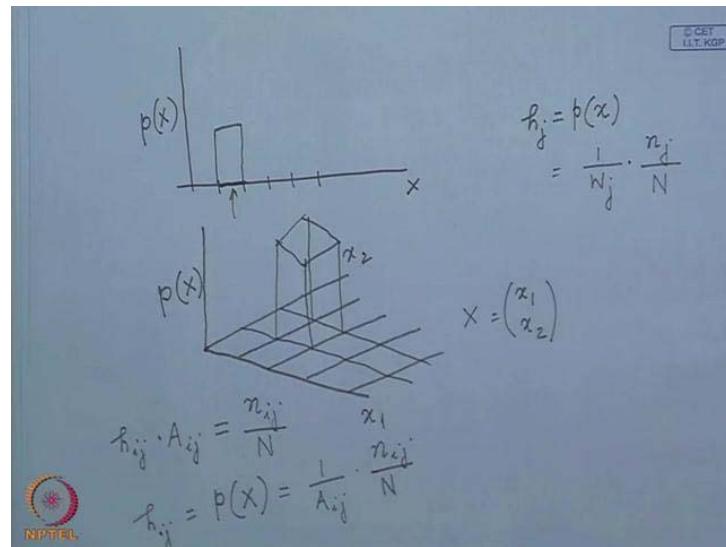
Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electrical and Electronics Communication Engineering
Indian Institute of Technology Kharagpur

Lecture - 14
Probability Density Estimation (Contd.)

Hello, so we were discussing about the histogram technique for probability density estimation. And we have considered different cases in one dimension, in two dimensions and we started our discussion in multi dimension and the dimension is more than two. So, we have seen that in one dimensional case, what we have is a probability density curve, in two dimensional case, what we have is a probability density surface.

And in both the cases, our constraint was that, in case of one dimension the total area under the probability density curve has to be equal to 1. In case of two dimension the total volume under the probability density surface that has to be equal to 1 and accordingly we have computed that what is the probability density value on every bin whether it is in one-dimension density or in two dimensions. So, in case of one dimension bins are actually the line segments like this.

(Refer Slide Time: 01:33)



So, if I have variable X , and on this axis I plot the probability density function $p(X)$, then what we have done is the X axis is divided into number of bins and number of cells. And within a cell we have assumed that the probability function is constant, so it is

something like this, where the height of this bar which is nothing but the value of the probability density within this cell, that is determined by the fraction of the training samples which falls under the bin and simultaneously what is the width of this particular bin.

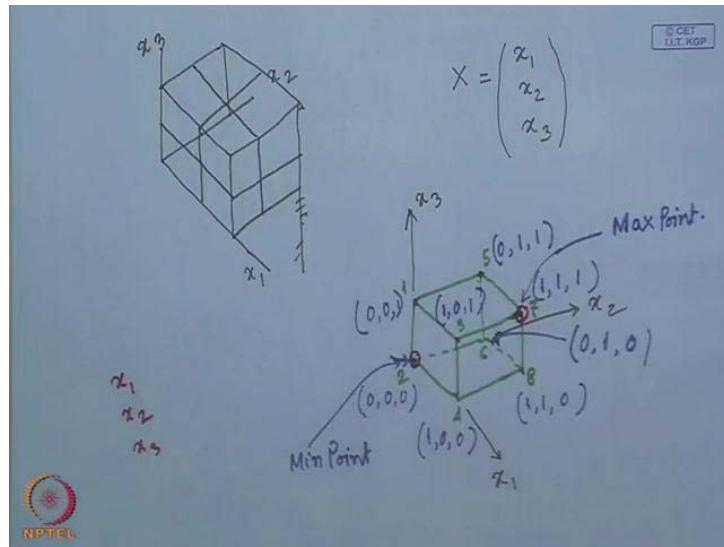
So, this h_j for j^{th} bin which is nothing but $p(X)$, where X lies within this j^{th} bin which is given by $\frac{1}{W_j} \cdot \frac{n_j}{N}$ where W_j is the width of the j^{th} bin, n_j is the number of training samples, which falls under the j^{th} bin and capital N is the total number of the training samples. So, this is what we had in one dimensional case or when a variable x is a scalar variable.

In two dimension, we have extended this concept, so we have vector variable X whose components are x_1 and x_2 , so I have this plane x_1, x_2 , where this is the x_1 axis, this is the x_2 axis and axis perpendicular to this, this indicates what is the probability x ? Where x is a vector. So, following the same concept we defend this plane x_1 and x_2 plane into a number of bins like this where each of this bins is now a rectangle. The probability density value over a bin will be represented by a bar something like this, where this bar is having some volume is given by if this bin $(i, j)^{\text{th}}$ bin.

So, h_{ij} times A_{ij} , where A_{ij} this is the area of this bin and we say that this has to be equal to $\frac{n_{ij}}{N}$, where n_{ij} is the number of samples which falls under this $(i, j)^{\text{th}}$ bin and capital N is the total number of samples. So, from this we can compute h_{ij} which is nothing but $p(X)$ where this X is a vector. So, what is the probability that this vector x which falls under the $(i, j)^{\text{th}}$ bin? So which is given by $\frac{1}{A_{ij}} \cdot \frac{n_{ij}}{N}$ and here again it is assume that within a bin the probability density is constant, but the probability value varies from bin to bin.

So, we have extended this concept to three dimensions where obviously in case of one dimension as the bins are line segments. In case of two dimensions the bins are area segments which are nothing but rectangles in our case. In case of three dimensions the bin will be volume segments, so it is a cuboids or say parallelograms thing like that.

(Refer Slide Time: 06:06)



So, in the last class we have taken a particular scenario, so my vectors x will be a three dimensional vector. So, vector X is equal to x_1, x_2, x_3 , so it has got three components, so accordingly I said this is my vector x_1 axis this is my x_2 axis and this is my x_3 axis. So, accordingly I define a three dimensional space and every vector is a point in that three dimensional space. So, to estimate probability density function what you have to is this three dimensional space has to be divide into a number of volume elements which are nothing but the bins.

So, in the last class we had taken a volume element something like this, so here you find out that there are eight volume elements, four on this side and four on this side. And once I divide this three dimensional space into such volume elements every volume element will have eight vertices. So, if I consider a single volume element out of this, so this volume element has got eight different vertices the vertices are one, two, three, four, five, six, seven and eight. So, these eight vertices uniquely defined this particular volume element.

Now, we take an advantage that because these volume element is a rectangular parallelogram, I need not specify all the eight different vertices rather if I specify only two vertices then also my volume element is completely defined. So, one of them if I say that this is my X axis or x_1 axis and this is the x_2 axis and this is x_3 axis considering this, you find that if I simply specify the coordinates of these volume element and the coordinates these volume elements that uniquely specify particular rectangular parallelogram, where the coordinates of this vertices, the x_1 component will be minimum of x_1 component of all the eight volume elements of all the eight vertices, x_2 component will be

the minimum of x_2 component of all these vertices, x_3 component will also be the minimum of x_3 components of all these vertices. Similarly, for this vertices you will find that x_1 component is the maximum of x_1 component of all the eight vertices. x_2 component will also be maximum of components of x_2 vertices of all the eight vertices, x_3 component will also be the maximum of x_3 component of all the eight vertices.

To make it more clear, let us assume that this is a unit cube and if you find that if it is unit cube and this vertex is at the origin, the coordinates of vertices will be $(0, 0, 0)$ coordinates of this vertices will be $(0, 0, 1)$. These vertices will be 1, x_2 components will be 0, x_3 component will be 0, for this vertices both x_1 and x_2 components will be 1 whereas x_3 component will be 0. So, it is $(1, 1, 0)$, for this vertices, all of them will be 1 for this component, I will have x_2 component 1, x_2 component 1, x_3 component will be 1 component is 0. So, this becomes $(0, 1, 1)$ for this one we have already done and for this which hidden it will be x_1 is 0, x_2 is 1 and x_3 is again 0.

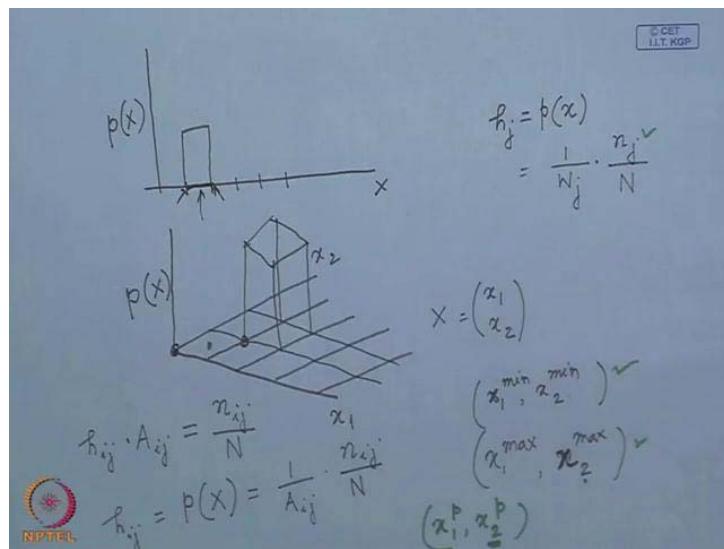
So, you find that I have got, this one is left, so this one will have x_1 component is 1 and x_3 component is 1 and x_2 component is 0, so it is $(1, 0, 1)$. So, I have one, two, three, four, five, six, seven, eight, so all the vertices. Now, if you look at the coordinates of all the eight vertices, you find that considering the x_1 component, the x_1 components of all the eight vertices are either 0 or 1, so the minimum is 0. Similarly, x_2 component minimum is 0, x_3 component minimum is 0. So $(0, 0, 0)$, is one of the vertices which is required to define this volume element and that is what is this one. Similarly, the maximum of all the x_1 component is equal to 1, maximum is also x_2 is also 1 and the maximum of all the x_3 component that is also 1.

So, it is $(1, 1, 1)$ which is another vertex which is necessary for defining this particular rectangular parallelogram and which is this one. So, if I simply know these vertices and these vertices I can form this parallelogram and because this is the minimum of all the coordinates. So, I call this as min point and this being maximum all the coordinates, so I call this as max point. So, one thing is clear that if I know the min point and max point of every volume element then my volume element is uniquely identified.

Now, what is the use of this min point and max point we said before whether it is in one dimension or two dimensional cases that to estimates the probability density function in a cell or bin. So, in three dimensional case each such volume elements of the cells or the bins. So, to estimate the probability density function in each cell or in each bin I need to find out how many vectors, how many points or how many training vectors are actually falling in the bin.

So, in case of one dimension you find that I know boundary, this minimum boundary of the line segment and I know maximum boundary of this line segment. So, value of x which is greater than or equal than to this and less than this falls under this bin. So that is my check how do I found out that, how many training samples are falling within this j^{th} bin you come to two dimensional case every point here also I have say for example this bin.

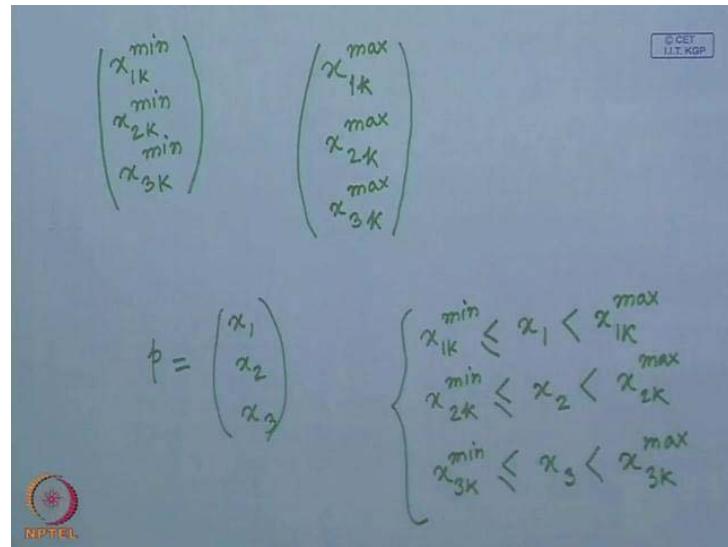
(Refer Slide Time: 15:04)



So, this will be the min point, this will be the max point, so this min point has coordinate x_1^{\min} and x_2^{\min} . So, this is the coordinate this point and coordinate of this point is x_1^{\max} and x_2^{\max} this is the coordinate of this point. Now for any point p lying within this cell. I must have say point p who is coordinate is x_1^p and x_2^p condition must be that x_1^p has to be greater than or equal to x_1^{\min} and it has to be less than x_1^{\max} , simultaneously x_2^p has to be greater than or equal to x_2^{\min} , it has to be less than x_2^{\max} , so if I have simply have the min point and max point, I can easily determine that what are the samples which falls under which bin and I can count the

number at such samples. So, the number of sample falling under the j^{th} bin, in case of one dimension will be n_j and the number of samples falling under $(i, j)^{\text{th}}$ bin in case two dimensional will be n_{ij} , so in three dimension as we have defined this min point and max point here also you find that for the min point

(Refer Slide Time: 16:52)



I have the coordinates, let us assume that $\begin{bmatrix} x_{1k}^{\min} \\ x_{2k}^{\min} \\ x_{3k}^{\min} \end{bmatrix}$, these are the min points of this, this is the

min point of say k^{th} bin. Similarly, max point I defined as $\begin{bmatrix} x_{1k}^{\max} \\ x_{2k}^{\max} \\ x_{3k}^{\max} \end{bmatrix}$ $\times 1$ max as it is for the k^{th}

bin, , so this is the max point of the k^{th} bin. Now, any point p at $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ if it has to lie within

this k^{th} bin in three dimension, I must have the condition that x_{1k}^{\min} must be less than or equal to x_1 which must be less than x_{1k}^{\max} .

Similarly, x_{2k}^{\min} must be less than or equal to x_2 which is less than x_{2k}^{\max} and x_{3k}^{\min} must be less than or equal to x_3 which must be less than x_{3k}^{\max} . If all these three conditions are satisfied simultaneously then only I can say that this point p falls within the k^{th} bin and p for whichever bin this condition all these conditions are simultaneously satisfied the point p fall within that particular bin. So, by using this given a set of three dimensional vectors, I can find out that how many of these vectors which are falling under which bin.

(Refer Slide Time: 19:43)

$$V_k \cdot p_k(x) = \frac{n_k}{N}$$

$$p_k(x) = \frac{1}{V_k} \cdot \frac{n_k}{N}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

So, if I find that k numbers of vectors are falling under k^{th} bin and capital N is the total number of training vector then the fraction of vectors which are falling under this k^{th} bin is

given by $\frac{n_k}{N}$. Now, the volume of the k^{th} bin if that is V_k and the probability density in the k^{th}

bin, if that $p_k(X) = \frac{1}{V_k} \cdot \frac{n_k}{N}$. Now, I find that I can find analogy from one dimensional case

and in two dimensional cases. In case of one dimension we said that the area under the probability density curve has to be equal to 1, in case of two dimensional case we have said that the volume under the probability density surface is equal has to be equal to 1.

Now, following the analogy over here you find out that this is my probability density value within the k^{th} bin, this is the volume of the k^{th} bin. So, this being the density and this being the volume I can say that $V_k p_k(X)$ represents mass, so the total mass under this probability density has to be equal to 1. So, this is a simple analogy which can be drawn from one dimension or two dimension to three dimension and later we see after some time that this can be extended to multi dimension.

So, once I have this, my simple calculation is the probability density value will be in the k^{th}

bin of x , where x is my three dimensional vector. So, this x is nothing but a vector $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

which falls under the k^{th} bin. So, the probability that a vector x will fall under the k^{th} bin is simply given by $p_k(X) = \frac{1}{V_k} \cdot \frac{n_k}{N}$

Where V_k is the volume of the k^{th} bin, n_k is the number of sample which is falling under the k^{th} bin and capital N is the total number of training samples. So, I can easily find out the probability density function in case of three dimension.

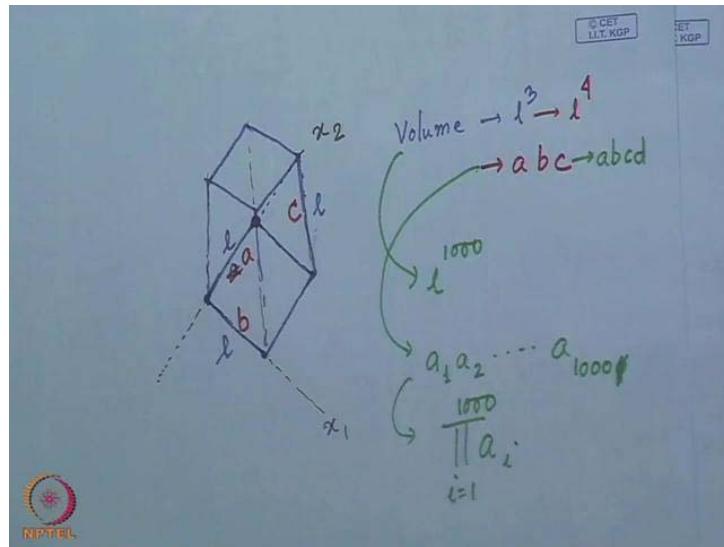
Now, let us see that how we can extend these two multi dimension because as you have seen in earlier in our case it is not necessary that I will have only the scalar variable or vectors variables, two dimensional variables or three dimensional variables and so on. The dimensionality of our vectors can be much more than that because when you are dealing with the pattern recognition each vector is formed by different features of the patterns.

I can compute one feature, I can compute two features, I can compute three features, I can compute five features. I can consider ten features, I can consider hundred features, I can consider thousand features. These depends upon what is the complexity of the that pattern we are trying to deal with, if the pattern is very simple only one feature may be sufficient in which case univariate probability density function is sufficient for us. If the pattern is slightly more complicated, I may be satisfied with the three numbers of features which gives me three dimensional feature vectors.

So, probability density function in three dimension like this what you have computed that may be sufficient if the pattern is very, very complicated or I may have to hundreds of feature to represent the pattern, may be a thousand. So, those thousand features forming a feature vector the dimensionality of my feature vector becomes thousand. So, the space that I have to consider in which I have to estimate the probability density is neither one nor two nor three it has to be thousand dimensional space.

And Visualization of a thousand dimensional space is not that simple may be somehow we can be visualized up to three dimension, but we cannot usually visualize a thousand dimensional space, but we have do is I have to work using the concepts. So, before I go into that let us assume, let us just look at that how I can evolve a multi-dimensional space, let us consider a very simple case initially.

(Refer Slide Time: 25:11)



Suppose, I have a point, so this is a point and we all know that a point is of zero dimension. It neither has length nor breadth and so on. So, it has zero dimension, now if I pull this point in a certain direction along the certain lines what I get is a line segment and a line is in one dimension. So, I find that from zero dimension which was a point if I pull the line in a particular direction what I trace is a line and this line is defined in a single dimension because I know what the direction of the line is.

So, what I have is a line in one dimension because I know what is the direction of the line so what I get is a one dimensional space and a straight line is defined in one dimensional space. Now, given this if I pull the line or drag the line in a direction perpendicular to the direction of the line or in other words, I draw the line in this direction, now I get the line over here.

So, you found that I have done is a rectangle, so from one dimensional space which contains the line if I draw the line in a direction which is perpendicular to the direction of the line what I define this rectangle and this rectangle is in two dimension. So, if this was one direction dimension this is the other dimension one of the dimension I call as x_1 and the other dimension I call as x_2 . So, I have a 2 dimensional space and in the two dimensional space I have a rectangle or even a square which is defined in a 2 dimensional space. Now, what I can do is I can drag this square of this rectangle in a direction which is perpendicular to the plane x_1 and x_2 , so if I drag it in a direction which is perpendicular to x_2 like this.

So, this was the plane I have dragged this plane in this direction perpendicular to x_1 , x_2 what I have got is this now I find that what I have actually got is a cube because if the displacement

or the amount of drag in every direction is same that I get a cube if they are not same I get parallelogram. This parallelogram is defined in three dimensions, now come to a case that will this point was dragged by an amount say l , the length of the line becomes l . If I drag this line perpendicular to the direction of the line by the same length l I get a square, whose area is l^2 , if I drag this square in the direction perpendicular to the plane $x_1 - x_2$ I get cube.

So, this is also one and I get a cube whose volume is simply l^3 , now if the amount of drags are different suppose this drag is a , this drag is b , the amount of this drag is c amount of this drag is d . What I have get this parallelogram whose volume is simply $a b c d$, now let us use some other concept that I have a three dimensional space and in the three dimensional space I have a cube or a parallelogram, now I can also assume that this cube or this parallelogram is something like a planar segment, a plane segment in a hyper plane as we said that up to three dimensions, we say planes beyond three dimensions, which we cannot visualize easily we add the term hyper.

So, I can consider that this cube or this parallelogram is a plane planar segment in a hyperplane, so if I drag this cube in the direction which is orthogonal to the hyperplane what I have is I have another cube or another parallelogram in four dimensions, so this is something conceptual. So, I have another cube or another parallelogram in four dimension and if that drag is same as that drags which you have done given before. That is how drag it the same displacement l the volume of that hyper cube in four dimension will be into the power four.

So, what we have done over here that in three dimensions the volume of the cube is l^3 , in four dimensions this will l^4 if it is hypercube in the four dimension whereas if I drag it by displacement d along the fourth dimension. I get a hyper parallelogram in four dimensions and the volume of the hyper parallelogram will be simply abc which was the volume of the parallelogram in three dimensions in four dimensions it will be $abcd$. So, you find out the concept is very simple and I always have analogy to lower dimensions, I may not stop there I can consider this is this hypercube is actually a plane segment in hyper plane in four dimension.

So, if I drag this four dimensional planner segments in a direction which is orthogonal to that hyper plane I get another parallelogram in fifth dimension. And if that amount of drag is same as l the volume of that hyper cube in five dimensions will be instead of l to the power of 4 it will be l to the power of 5. And the if the amount drag is e in the fifth dimension the

volume of the hyper parallelogram that I get in find the dimension will be a b c d times e where e is the amount of drag in the fifth dimension.

And this concept I can go on it is extending in multiple dimension, so if I have a feature space of dimension say 1000, so volume of a hyper cube in 1000 dimensional space will be given by 1 to the power 1,000 as 1000 is the dimensional of the space. This is a cube in 1000 dimensional spaces where every side of that hyper cube is equal to 1 considering from here. If a_1 is the length of the hyper parallelogram in first dimension a_2 is the length of that hyper parallelogram in the second dimension like this a 1000 is the length of that hyper parallelogram in 1000th dimension.

Then, the volume of the hyper parallelogram will be a_1 into a_2 into a three up to a 1000, so I can simply represent this as prod at a_i , $i = 1, 2$ and 100, so this will be the volume of the hyper parallelogram in thousand dimension. Now, whatever we do as we have seen in case of two dimensional spaces or in case of three dimensional space that every bin can be uniquely identified by the location of two vertices. One of the vertices is the min point the other vertices is max point and for any point to lie within this bin its corresponding coordinate must be greater than or equal to the corresponding coordinate of the min point and less than the corresponding coordinate of the of the max point.

So similarly in this multi-dimensional space, what we are doing is, this is basically either a hyper cube or a hyper parallelogram in a multi-dimensional space. So, to uniquely identify a hyper cube or a hyper parallelogram multi-dimensional space if I have the coordinates of the min point and the coordinate of the max point that is sufficient. I do not need to store any other vertex, the information of any other vertices of that hyper cube or hyper parallelogram, so if I have the min point and the max point that is sufficient for me, so given this now I can write the conditions.

(Refer Slide Time: 36:30)

The image shows handwritten notes on a blue background. At the top left, there are two green arrows pointing right: one labeled $V \rightarrow \text{min}$ and another labeled $W \rightarrow \text{max}$. In the center, the text "n-cube" is written above a red bracket under the word "cube". Below this, there are two sets of variables: $v_i ; 1 \leq i \leq n$ and $w_i ; 1 \leq i \leq n$. To the right of these, the text "p in ith dimension coordinate p_i" is written in red. Below the variables, the inequality $v_i \leq p_i < w_i \mid 1 \leq i \leq n$ is written. At the bottom, there is a diagram showing a 2D grid with points labeled p_k and r_k . A formula is written: $\rho_k = \frac{1}{\text{vol}_k} \cdot \frac{r_k}{R}$, where vol_k is the volume of the kth bin, R is the total number of samples, and r_k is the number of samples in the kth bin. The NPTEL logo is visible in the bottom left corner.

That say V is the min point and let us say W represents the max point of a volume element. Whether the volume element is in one dimension or in two dimension or in three dimensions, four dimensions, thousand dimension ten thousand dimension do not matter. I have defined what is the min point and what is max point and suppose this is defined will say n dimensional space it may be one it may be two it may be three it may be anything this is n dimensional space. Incidentally, let me mention that in n dimensional space a cube, a hyper cube in n dimensional space is actually called an n cube. Similarly, a hyper parallelogram in n dimensional space will be called n hyper parallelogram, these are the terms that we use. so what I have is n dimensional space.

So, accordingly this min point V will have n number of components, so it will have V_i , for i , $1 \leq i \leq n$. So, different coordinates of the min point similarly, for the max point I will have different coordinates which is given by W_i where $1 \leq i \leq n$. So, once I have defined my min point and max point, now suppose I have a point p in i th dimension the coordinate of this point p will be p_i , so in i^{th} dimension or in i^{th} direction, coordinate is p_i , so here again i varies from 1 to n .

As we have n dimensional space, so p is n dimensional vector as V or W each of them are dimensional vectors, so if this p has to lie within the volume element or within the bin defined by V and W , I must have the condition that V_i must be less than or equal to p_i . The corresponding coordinate which must be less than W_i and this has to be satisfied for all i within 1 to n , so $V_i \leq p_i \leq W_i$, V being the min point and W being the max point. So, if this is satisfied for all i between 1 to n that means in every dimension the corresponding coordinate of p must be within the minimum limit and the maximum limit of the corresponding dimension of the hyper cube.

So, if this is satisfied for all the i that is every coordinate of the point p , then we say that point p is contained within the hyper cube or the corresponding hyper parallelogram. So, given this, now you find that estimating probability density function or probability density value even in the histogram technique in n dimension is as simple as we have done in case of one dimension or in two dimension or in three dimension. So, what I have to do is this n dimensional space that has to divided into a number of bins or cells or every bin or cell we have dimensional. So, these will be divided into number of bins or number of cells in n dimension then when I have a set of training samples so the training vector.

So, every training vector is of dimension n , I have to compute that how many of this training vectors falls under say k^{th} bin and where this bin is n dimensional and I identify with a vector is falls in the k^{th} bin or not simply by using this condition that the min point of the k^{th} bin and

at the max point at the k^{th} bin, so every coordinate of the feature vector must be greater than or equal to min point of the k^{th} bin the corresponding of the min point of the k^{th} bin. It must be less than the corresponding coordinate or the max point of the k^{th} bin, so by this I can identify that how many of this samples falls under the k^{th} bin.

And then, the fraction of which falls under the k^{th} bin simply $\frac{n_k}{N}$ where n_k can assume that

the number of samples which fall under the k^{th} bin and capital N is the total number of samples. So, this fraction of samples falling under the k^{th} bin divided by the volume of the k^{th} bin where volume of the k^{th} bin is simply either l^n because we are using in the dimensional case or if it is a hyper cube and l being side of the hyper cube or a I take the product I went from one end if it is a parallelogram.

So, after all it is the fraction of the sample which falls under the k^{th} bin divided by the volume of the k^{th} bin that gives me the probability density estimate within the k^{th} bin. So, a sample will fall within that k^{th} bin with a probability density as estimated by this method, so what we

have is a p_k is the probability that a sample will falls under the k^{th} bin it is simply $\frac{1}{V_k}$ where

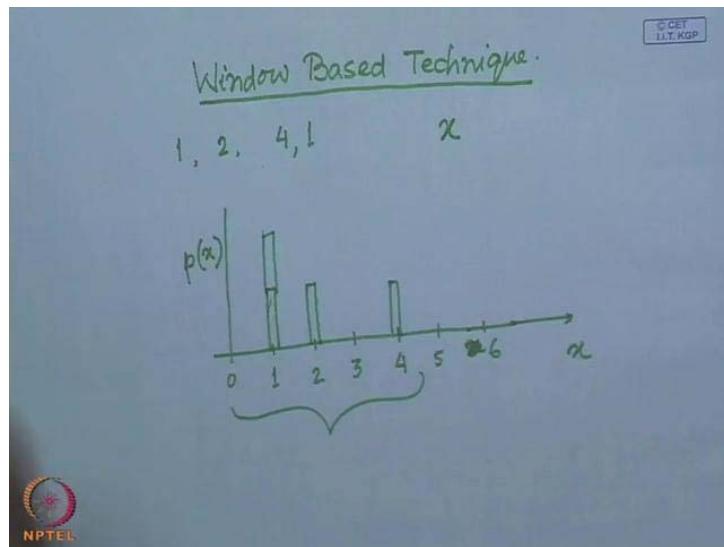
V_k is the volume of the k^{th} bin or W and V are getting. Let me put it other way, so q instead of p let me call it q because we are using p to represent the point here.

So, q_k which is the probability that a sample falls under the k^{th} bin in simply given $\frac{1}{\text{VOL}_k}$

which is the volume of the k^{th} bin into say let me put it as say $\frac{r_k}{R}$ where r_k is the number of samples in k^{th} bin and R this is the total samples, total number of samples, so whether we try to find out that trying to estimate the probability density function in 1 dimension or in two dimension, three-dimension n dimension case does not matter, the concept is this.

I can use it in five dimensions, I can use it in ten dimension I can also use in hundred dimension it does not matter. So, this is what we have discussed about the histogram based technique for probability density estimation. We said initially that there are many other techniques, but the other techniques will have that we are going to discuss in this course is what is called window based technique.

(Refer Slide Time: 45:43)



So, the next technique that I will discuss now is what is window based; now what is called window based technique? In case of window based technique it is assumed that every sample is a representative of a probability density function. So, this representative can be delta function or it may be any window function however our constraint is that that total area under this density probability curved we are going to estimate that must be equal to 1, so if I use a delta function are very simple case if I use that probability density function is a delta function. And every sample represents the probability density function which is an area function, so what is a delta function it is a spike with a various small width and as before the area under this delta function will be the width multiplied by the height of that spike, so if I situation something like this that I have say let us actually will go with three samples 1, 2 and say 4. So, these are the three sample that we have obviously in this case we have assuming that we have a scalar variable x and these are three samples scalar of this variable.

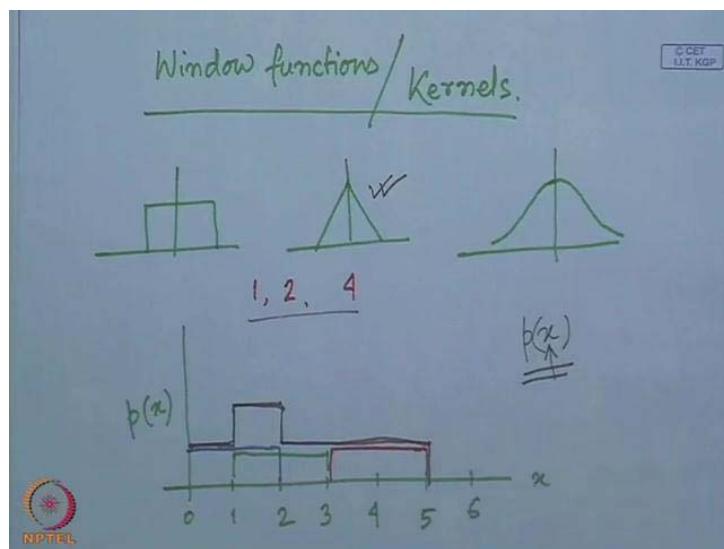
So, if I use, so this is my x and this direction I plot probability of x, so I have a sample this is 0. This is 1, this is 2, somewhere here it is 3 and this is 4, this is 5, this is 6 and so on and as I said that every sample is a representative of a probability density function. So, initially that assume the representative is probability density function is a delta function something like this, so I have one delta function at location $x = 1$, I have another delta function at location $x = 2$. And I have another delta function at location $x = 3$ and as we said that a constraint is total area under this probability density function has to be equal to 1. So, that clearly says that area of this delta function is to be $\frac{1}{3}$ area of this delta function has to be $\frac{1}{3}$ and area of this delta has to be $\frac{1}{3}$.

Now, if I have any situation something like this that my samples are 1, 2, 4 and I have one more sample at location 1, then obviously there are two delta functions which will be super imposed that location is equal to 1. So, this height of the delta function will be just a double and not only that area of individual delta functions has to be one-fourth in the earlier case I had only these samples. So, the area of individual delta function was equal to one-third, so that when I add all of them all the delta functions together the total area becomes 1, but now I am considering a case where I have four delta functions because I have two samples lying at location at 1.

So, I have to have four delta functions and the total area has to be equal to 1, so the area and under every delta function has to be $\frac{1}{4}$, so I have a situation something like this area of the delta function at $x = 1$ has to be doubled of the of the area the delta function at location $x = 2$ or area of the delta function at location $x = 4$ total area has to be equal to 1. Now, if I have this sort of situations, then you find that this is not really approximating a continues probability density function because then what I will is I have a set of spikes a set of disjoint spikes.

So, I am not really estimating a continues probability density function, but what I have to do is, I have to estimate continues probability function from the set of sample which are given. So, the delta function to be a representative probability density function is not really suitable instead of delta function, I can use some other functions called window functions or kernels, which can help us to estimate the probability density function in a smooth way.

(Refer Slide Time: 51:27)



So, what we will use is, window functions or which are also called kernels, of course they have to be properly normalized so that the total area remains to be 1. So, this window function or kernels again can be different types, I can use a very simple rectangular window, I can use triangular window. So, the window if I use a rectangular window, the window will be something like this which has made in box function if I use a triangular window, the window will be something like this. Or even I can also use a normalized normal distribution, so the window will be something like this, so different types window function or different types of kernels can be used to estimate the probability density function, so find that over here when I have a situation that I have three samples at location 1, 2 and 4.

The situation was something like this, so if I use say simpler case this rectangular window then at 1 I will have rectangular window. Suppose, the width of the window is let us assume two so I place a window two over here, at location 2 again I have to put a rectangle window let us say whose width again equal to 2, so this window will be something like this and location four, again I will put a rectangular window whose width is again equal to 2. So, I will have a window something like this, so the over all probability estimate is basically sum of all these window functions and I have to normalize these windows in such a way that, because I have three different windows.

So, the total area under this three different windows has to be equal to 1 and by using this you find that over all probability density function as it is the sum over all the density estimates will be something like this. So, this curve gives me what is the probability density estimates of $p(x)$, so now what we have done is something visually. In the next class, we will talk about if I use instead of rectangular windows say a triangular window or some other window, how can we estimate the probability density functions. We will also find analytically that how we can estimate given these different samples say 1 to 4 or something like this.

These different samples and given the kind of window function that you are going to use then how we can estimate the probability density at an unknown location x . So, I want to estimate this $p(x)$ where x is not in any of this samples which are given for training. So, these samples are given, your given a window function which has to be use to estimate the probability density. Now at an unknown x , I want to estimate what will be $p(x)$. So how we can do it by using the window function, so that part we will consider in our next class.

Thank you.

Pattern Recognition and Applications
Prof. B. K. Biswas
Department of Electrical and Electronics Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 15
Probability Density Estimation (Contd.)

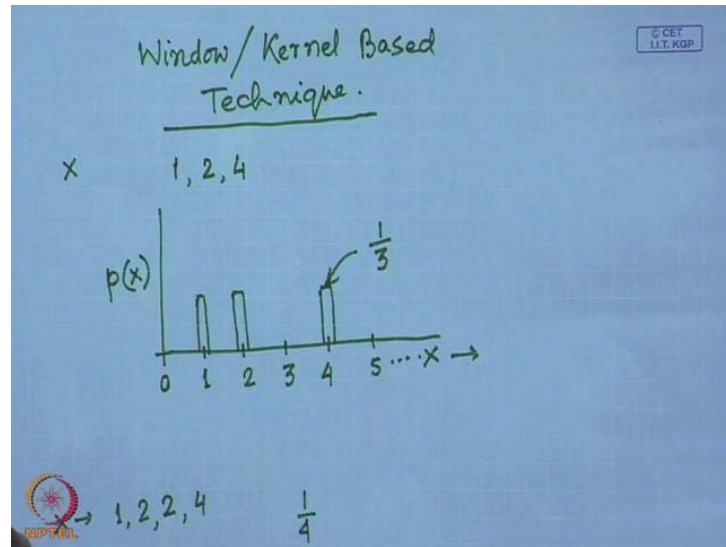
Hello, so till the last class we have discussed about the histogram based technique for probability density estimation. And we have started our discussion on kernel based technique for probability density estimation and after discussing about the histogram technique when we started the kernel based technique for probability density estimation. You have noticed that in histogram based technique what we do is, we divide the feature space or the vector space into a number of cells or bins where every cell or bin in one dimension or two dimensions.

In case of one dimension, it has a certain length or certain width; in case of two dimensions it has got certain area, in case of three dimensions it has got certain volume and when we move above three dimensions say dimension four and above, we call them as hyper volumes. Accordingly, we can define the volume of every cell or every bin, and then the density multiplied by length or the area or the volume, we have said that this should be equal to the fraction of the samples, the fraction of the feature vectors which are given for turning purpose.

So, this probability density multiplied by width of the bin in case of one dimension or the probability density multiplied by area of the bin in case of two dimensions or it is the probability density multiplied by the volume of the bin, in case of three dimensions and more dimensions that should be equal to the fraction of the samples falling within the bin. And accordingly, we can find out that what is the probability density function in that particular bin? And our basic assumption was that within the bin, we assume that the probability density function is constant.

So, once we do that then we can find out, we can estimate what is the probability of a sample falling within a bin, because we know what is the probability density function, value of the probability density function within the bins. And then we have said that we have got other estimation technique, that is the window or kernel based technique.

(Refer Slide Time: 03:11)



And so, what we are going to discuss now is the window or kernel based technique. So, in this window or kernel based technique what we have said is each sample, each feature vector is a representative of a probability density function. And this probability density function has to be so normalized that the total area or the total volume under the probability density function remains equals to 1. So, in the simplest case if we assume that every sample represents a probability density function where the probability density function is a delta function. So, we started with an example we said that suppose we have got, let us take a very small example.

Say we have got samples, I assume that I have a single variable or a scalar variable x and the samples which are given for estimation of the probability density function those sample are say $x = 1$, $x = 2$ and $x = 4$. So, using just three samples, we want to estimate the probability density function and I assume that every sample represents a probability density function which is a delta function. So, a delta function has a very small width and certain height and because we have got three samples at 1, 2 and 4, for estimation of probability density function.

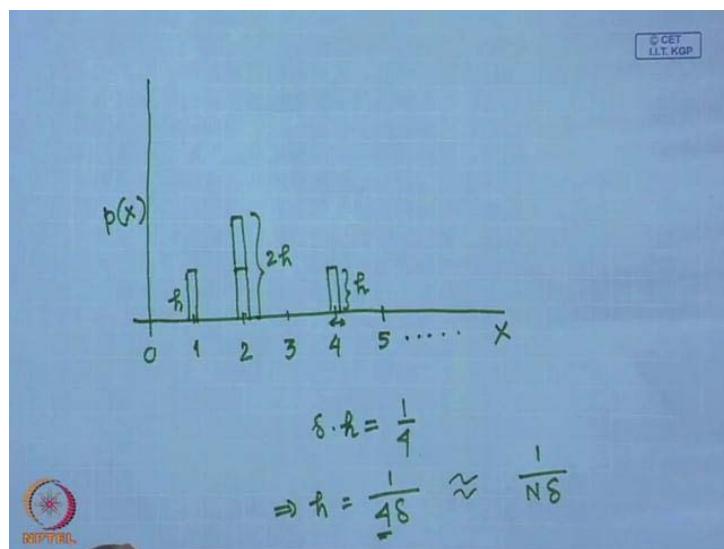
So, we have three such delta functions and the total area of the delta functions have to be equal to 1, which means that the area of a single delta functions, area under single delta function has to be equal to $\frac{1}{3}$, so I have a situation something like this. I have this variable x and on this axis I put probability of x and I have $x = 0$ here,

$x = 1$ here, $x = 2, 3, 4, 5$ and so on. So, here you find that if a sample at location 1 gives a probability density estimate which is a delta function, so this probability density estimate will be something like this. Similarly, at 2 it has a probability density estimate which is something like this; at 4 it also has a probability density estimate something like this.

And the total area of these three delta functions have to be equal to 1, so that clearly means that area of each of these probability density functions have to be $\frac{1}{3}$ because we have three such delta functions. Now, I can have situations something like this that I can have more than one sample or more than one instances of a single value of x . So, I can have a situation something like this that I have one sample of x at location 1, two samples or two values of x which are 2 and I have 4.

So now, I have four samples of x and using this four samples of x , I have to estimate the probability density function, so because I have four samples of x . So, I will have four such delta functions and the total area under these delta functions have to be equal to 1, so the delta function or the area of the delta function are, for a single sample obviously has to be $\frac{1}{4}$.

(Refer Slide Time: 07:48)



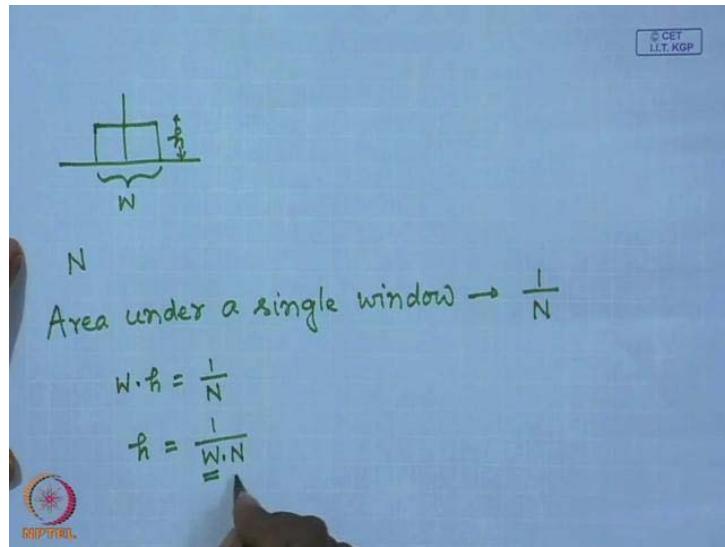
So, given this I will have a situation of this form, I will have so on this side I will put probability of x , this is my x , this is 0, 1, 2, 3, 4, 5 and so on. So, I have one delta function at location 1, I have two delta functions at location 2 because you see that I have two samples of x both of them having value 2, so I have two samples at location 2, so here I have two samples. So, the height of this delta function has to be doubled and I have one sample at $x = 4$, so this is the kind of situation I have and if the width of every delta function is say δ and area has to be $\frac{1}{4}$.

So height of every delta function, if it is h , I have $h = \frac{1}{4\delta}$, so here is 4 because I have four samples to estimate the probability density function, if I have N number of samples then this $h = \frac{1}{4N}$. So here the height of this delta function will be h , height over here will be $2h$ and height over here will be again h .

Now, when I have such a kind of delta functions to be used for the estimation of probability density functions. You find that I do not get a smooth probability density estimate or in other sense that this is not really suitable for estimation of continuous probability density function. So, if I want to estimate a continuous probability density function from the set of given samples, this choice of delta function is not a good choice.

So, instead of using this delta function what I can use is a window function or a Kernel function and the window function or the Kernel function and the windows can be of different types. I can have a rectangular window, I can have a triangular window or even the window function can have the shape of normal distribution, so depending upon the choice, I can use a rectangular window.

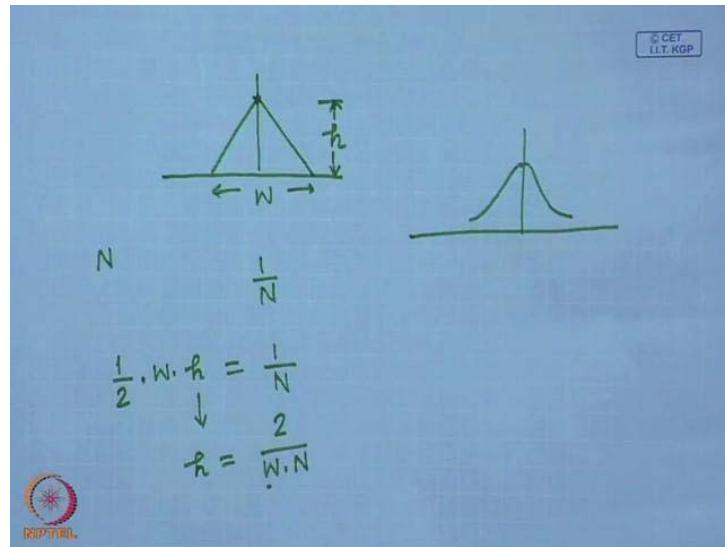
(Refer Slide Time: 10:53)



So, the rectangular window will be something like this of course, it will have certain width and certain height. So, if I decide if I fix the width of the window then the height of the window will actually be determined by the total number of samples which are available for the probability density estimation. So, if I have a total of capital N number of samples, I will have N number of such windows and the total area of all these windows taken together has to be equal to 1. So, the area of a single window area under a single window, so I will write area under a single window will be $\frac{1}{N}$, if N is the total number of samples which are, which is given for the probability density estimation and from here you find as I have fixed the width of the window function that is W .

So, the area of under this window is $W \times h$ which will be equal to $\frac{1}{N}$. So, naturally the height of the window function is $h = \frac{1}{WN}$, where W is the width of the window and N is the number of samples which are given for probability density estimation. Now, instead of this rectangular window if I use a triangular window, let us see what kind of window, if I use a triangular window.

(Refer Slide Time: 12:57)



Triangular window will be something like this; again this triangle will have certain width and certain height and if I fix the width to be W then height h of this triangular window will be determined by again the number of samples. So, again if I assume that I have total N number of samples given for the probability density estimation, then the area under a single window. I have total N number of samples and I have total N number of such windows and the area of all these windows are taken together will be equal to 1. So,

area under a single window will be $\frac{1}{N}$, so over here if I compute the area of this triangular

window, area of the triangular window is nothing but $\frac{1}{2} W \cdot h$ and which has to be equal to

$$\frac{1}{N}$$

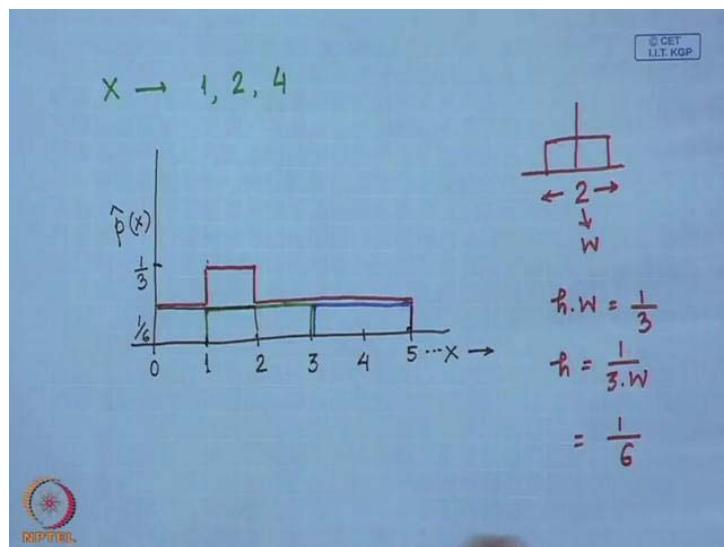
So, from here I can compute the height of the triangular window h which has to be equal to

$h = \frac{2}{WN}$, where this W is the width of the window or the length of the base of the window

and h is the height of the window. So, every sample represents a probability density function where this probability density function is given by such a triangular probability density having base width is equal to W and the height is equal to h . I can also have a normal density function to represent the density given by every individual sample.

So these, I can have other types of window functions as well this is not all that I can use however these are the simpler ones. And we will explain our concept of window technique for probability density estimation using the rectangular window or a triangular window. So, let us take the same example of three samples.

(Refer Slide Time: 15:40)



I have this scalar variable x having sample at location 1, a sample at location 2 and a sample at location 4. So, what I will do is, I will first use this rectangular window and see how this probability density can be estimated using the rectangular window. So, I will plot estimated probability density p of x and because it is estimation instead of writing $p(x)$, I will write $\hat{p}(x)$ because I exactly do not know what is $p(x)$ and along the horizontal axis or along the x axis, I will put this scalar variable x .

So, this is 0, this is 1, this is 2, this is 3, this is 4, this is 5 and so on and initially I will use a rectangular window. And let me assume that the width of the rectangular window is equal to say 2, so I use a rectangular window and width of the rectangular window I assumed to be 2. So, this is my width W and as I said before that because I have got three such samples, so area of individual window has to be $\frac{1}{3}$.

So, accordingly I have to compute what is the height of this rectangular window, so the height of the rectangular window $h \cdot W$ that has to be $\frac{1}{3} \cdot W$. So, height has to be $\frac{1}{3} \cdot W$ and

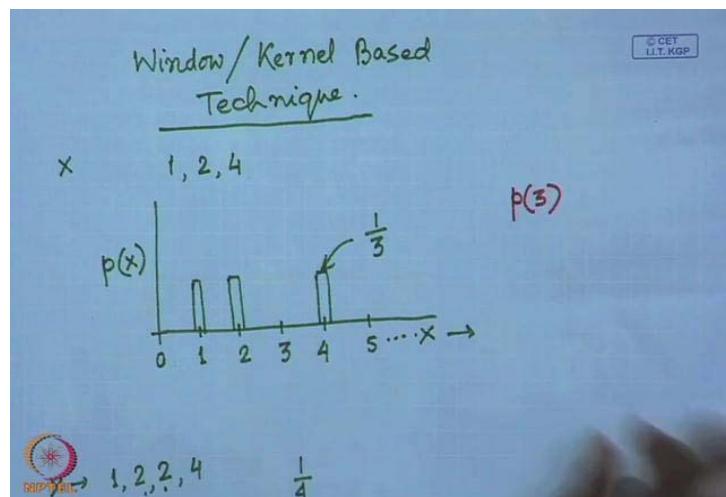
in this case $W = 2$, so this will be $\frac{1}{6}$. Now, let us place such a window of width 2 and height $\frac{1}{6}$, one at each of the sample locations. So, I have one window of width 2 and height $\frac{1}{6}$, this is $\frac{1}{6}$ at location 1, at 2 I have another sample. So, at location 2, I have to put one such window again so that window let me use different colors will be this height remains the same $\frac{1}{6}$.

And the window spans from 1 to 3 because the width of the window is 2 and the same window I also put, I also place at location 4, so I will have a window something like this it spans from 3 to 5. So, once I have this sum of all these areas gives me an estimate of the probability density function. So, if I add them up you find that from location 0 to 1 in this range, I have got only one window so the height of this window will remain $\frac{1}{6}$.

Then from location 1 to 2 in this region, I have two windows which are super imposing one window which was placed at location 1 and the other window which was placed at location 2, so these two windows overlap.

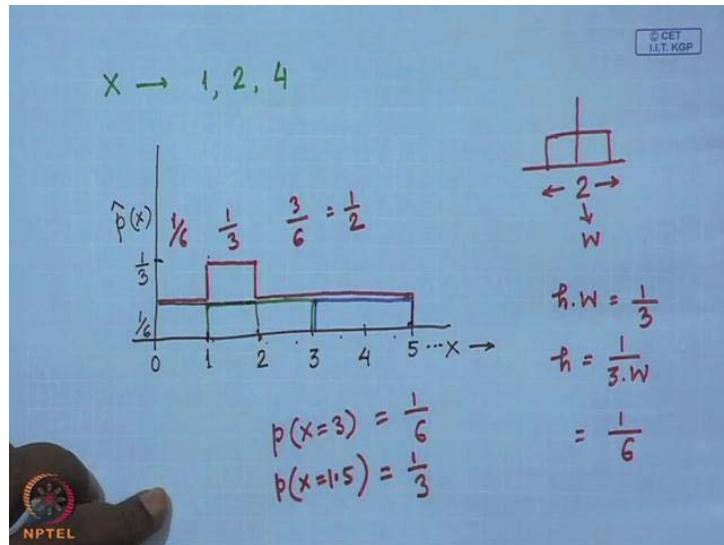
So, obviously here the probability density will be the sum of the heights of the individual windows, so here it was $\frac{1}{6}$. So, at this location this height will be $2 \cdot \frac{1}{6}$ that is $\frac{1}{3}$ and from 2 to 5 the height will be $\frac{1}{6}$. So, this red line or the red curve gives me an estimate of the continuous density of the probability function as represented by these three samples at location 1, 2 and 4. And now, if you compare this with this one that we did earlier with delta function you find this probability density estimate is not a continuous probability density estimate, but it is a discrete probability density estimate.

(Refer Slide Time: 21:02)



Say for example, if I want to estimate what is $p(3)$ here it will give $p(3) = 0$ because the delta functions do not span up to $x = 3$.

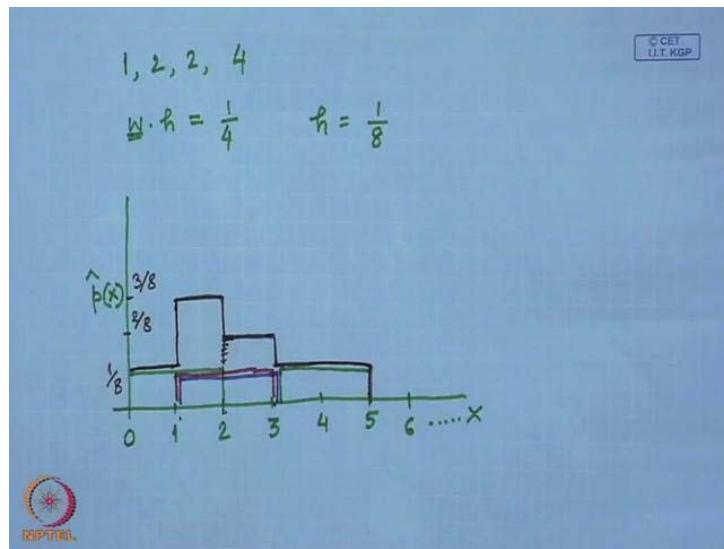
(Refer Slide Time: 21:18)



But in this case, when I use a window function or a kernel function I can say that $p(3)$ will be equal to which is height of this probability density curve, which is equal to $\frac{1}{6}$, whereas over here if I want to compute this $p(1.5)$ that probability that $x = 1.5$ which is given by $\frac{1}{3}$ that is the height of the probability density curve in the region 1 to 2. So, I can use this window technique for such probability density estimate. Now, if I want to compute what is the total area under this probability density curve, you find that from here to here, the area is $\frac{1}{6}$ because height $\frac{1}{6}$, width is 1.

So area is $\frac{1}{6}$, from here to here, the area is $\frac{1}{3}$ and from here to here, the area is I have 1, 2, 3, three different segments of length one each is having a width of $\frac{1}{6}$. So, it will be $\frac{3}{6}$ which is nothing but $\frac{1}{2}$, so if I add all these two, I will get the total area under this probability density curve. So, this was a very, very simple case when we have used a rectangular window for estimation of probability density. Now, let us see for the same example, one more thing what will happen if I get multiple numbers of samples at a particular location.

(Refer Slide Time: 23:35)



So, before I go for use of other window functions, let us see that what will happen, If I have the samples as 1, 2, 2, 4 if I use this then obviously I have to place two windows at location 2 and total number of samples I have is 4. So, area under every window will be $\frac{1}{4}$ and let us assume the width of the window I use the same which is equal to 2.

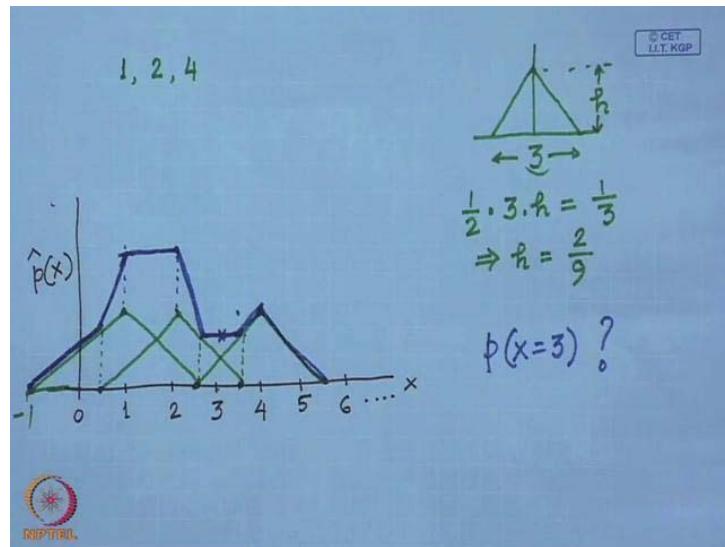
So, width into height has to be equal to $\frac{1}{4}$, now this is width is 2, so height of every window will be $\frac{1}{8}$ instead of $\frac{1}{6}$ which we had over here which we had when we had only three samples. Now, we have four number of samples though there are two samples at $x = 2$, so when I have this, now let us try to estimate the probability density I have x , I have the estimated probability density $\hat{p}(x)$.

So, again let us put 0 1, 2, 3, 4, 5, 6 and so on, so here what I will do is I will put a window of width 2 and height $\frac{1}{8}$ at location 1. So, this window will span from $x = 0$ to $x = 2$, I place two windows of same height and same width at location 2. So, I will have one window and this will span from $x = 1$ and $x = 3$ because width is 2, so this is one window and this is another window and then I have a single window at location $x = 4$ again of width 2 and height $\frac{1}{8}$. So, this window will span from $x = 3$ to $x = 5$ because width equal to 2 and now

if I combine all these windows together that is what is going to give me the probability density estimate.

So, here you find that I had only one window this is $\frac{1}{8}$, here from 1 to 2 in this region; there are three windows which are superimposed. So, in these locations the height of the probability density that will be $\frac{3}{8}$, from 2 to 3 I have two windows which are superimposed, so here, the height will be $\frac{2}{8}$ and then from 3 to 5 I have a single window, so height will be this. So, this is what is my final probability estimated probability density function using a rectangular window of width 2 and I have four such samples with two samples at location $x = 2$. So, this is how I can estimate the probability density function, now let us see that instead of using rectangular window if I use a triangular window, then what will be the situation.

(Refer Slide Time: 28:00)



I use the same set of samples one sample at location 1, one sample at location 2 and one sample at location 4, so I have three samples and I assume that my rectangular window that I will use that will have a width of say 3. So, I have to find out what will be the height of this window, this is the width, so as you know that area of such a triangle is nothing but $\frac{1}{2} W.h$ width which is 3. So, this is the area of the triangle and area of each of these triangles has to

be equal to $\frac{1}{3}$ because I have three samples and the total area of all the windows have to be equal to 1.

So, the area of individual triangular window has to be equal to $\frac{1}{3}$, so I put this is equal to $\frac{1}{3}$ and once I have this from here I can obviously compare what is the height h which is nothing $\frac{2}{9}$. So, this is quite fine, now let us see that how if I place this windows use this windows rectangular windows, then how the overall estimated probability function is going to look like.

So, again I put it the same way this is x , this is the estimated density which is $\hat{p}(x)$, this is x equal to 0, 1, 2, 3, 4, 5, 6 and so on. So, I have to put this triangular window at location $x = 1$, one of this windows I have to put one window at location $x = 2$, I have also to put one window at location $x = 4$. So, let us place this window at location $x = 1$ I put a triangular window and the width of the triangular window is 3. So, that means that I will have 1.5 to the left side and 1.5 to the right side. So, it will be something like this say here if I have – 1, I will have situation something like this, this is the triangular window placed at location 1.

Similarly, the triangular window placed at location 2, it will span from 0.5 to 3.5, so I will have this triangular window like this. I place one window at location 4, because I have a sample at location 4 and this window will span from 2.5 to 5.5, so it will be like this, so my final probability density estimate will be the sum of all these triangular windows. So, if I compute this I will have a situation of this form say up to this point, I have only one window from here to here I have superposition of two windows and again up to this point I have superimposition of windows.

So, the kind of probability density estimate that I am going to have will be something like this from here to here it is single window, then up to this point it will be like this from here to here, it will be constant, again from here to here, it will be like this. Then again it will be constant, then it will follow single window like this, so my final probability density estimate it is given by the superimposed curve which is shown in this blue line, this is the final probability density estimate. Now, given this obviously if I want to find out that what is the probability density at location $x = 3$. You find that probability density location $x = 3$ is

nothing but the height, the functional value of the final probability density which is given by this value.

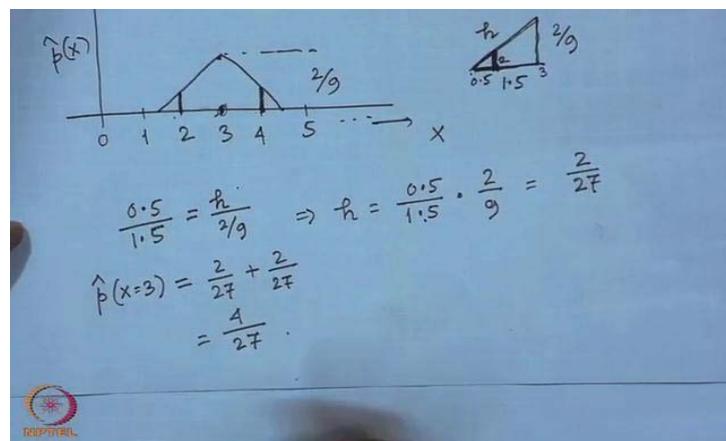
Now, how do you compute this value this is what conceptually, but finally I have to find out what is $p(x)$ at $x = 3$, this is what I have to compute. Now, if you notice carefully you find that this superimposition of different windows placed at different sample locations is nothing but a convolution operation. So, what I am doing is, if I assume that every sample location at every sample location I have unit pulse that is a pulse of height equal to 1.

And so, what I have is I have a set of pulse trains, one unit pulse at location $x = 1$, one unit pulse at location $x = 2$ and one unit pulse at location $x = 4$. So, if this pulse train is convolved with my triangular window the shape of the triangular window is defined over here. Then I will get this same probability, estimated probability density curve. So, this is nothing but convolution operation and because it is a convolution operation, so what will happen if I have multiple number of samples at a particular location.

So, as before if I have two samples at location $x = 2$ the height of that pulse I have to take is equal to 2, if I have a sample, single sample at a particular location, the height of that pulse will be equal to 1. It will be a unit pulse and if I have say k number of samples at a particular location the height of the pulse at that location will be equal to k . So, I can have pulse train of different heights and once I have this pulse train, what I will do is I will convolve this pulse train with the window function of the kernel function.

This convolution output will be my final estimated probability density function and that is what it is, so once I have this concept once I understand that this is nothing but a convolution operation. I have other way of estimating the probability density at location say $x = 3$, so what I will do because it is understood to be a convolution operation I can simply put it like this.

(Refer Slide Time: 36:55)



So, I will have this $p \hat{x}$ and this direction I will have x , I have $x = 0, 1, 2, 3, 4, 5$ and so on, what I will do is I will put a window at location 3 of height as we have said the height of this window is $\frac{2}{9}$. So, I will put one such triangular window at location $x = 3$, so this triangular window will be like this. Then what I will do is, I will find out that what the samples which are falling under this triangular window, and what is the value of this triangular window function at those sample locations wherever I have a sample.

And these values if at any location within this window I have multiple numbers of samples say k number of samples that value has to be multiplied by k that is what is convolution and add all these products together. So, that is what gives me the convolution, so here at location 2, I had one sample at location 4 I also had one sample, but I did not have any sample at location 3, height of this window as we computed it was $\frac{2}{9}$. I have one sample over here.

So, I have to compute what is this height I had one sample over here, I have to compute what is this height and sum of these two heights is the estimated probability at location $x = 3$. So, I can easily compute that following symmetrical triangles I have a triangle this is half of the base so this 1.5, the distance between 3 and 2 that is nothing but 1.

So, this location 3 and this is location 2 so this distance is 1, so this distance is 0.5 this height as we said it is $\frac{2}{9}$, so I have to compute what is this height. Now, if you follow the properties of symmetric rectangles, you find that I get two triangles over here, one triangle is this and one triangle is this and these triangles are symmetrical.

So, if this height is h following the properties of symmetrical triangles I can immediately say that $\frac{0.5}{1.5} = \frac{h}{\frac{2}{9}}$. So, I can immediately compute that this $h = \frac{0.5}{1.5} \cdot \frac{2}{9}$ and this will be $h = \frac{2}{27}$. So, this height is $\frac{2}{27}$ and by symmetry this height is also $\frac{2}{27}$ because this window is a symmetrical window.

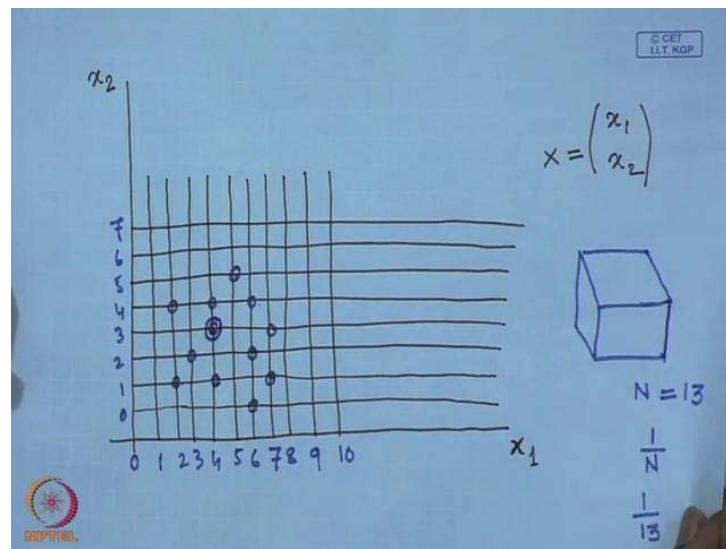
So, final probability density at location 3 will be twice of this $\frac{2}{27}$ because it is this height plus this height because these two are the samples which are in line within my rectangular window. So, the probability estimate $p \hat{x}$ of x equal to 3 this will be simply $\frac{2}{27} + \frac{2}{27}$

which is nothing but $\frac{4}{27}$, so this is what I have. So, the probability $x = 3$ estimated probability $x = 3$ is simply $\frac{4}{27}$ which I can compute like this if I use this one. So, you find that the probability density estimated probability density at location $x = 3$ is this one which is $\frac{2}{8}$ if I use a rectangular window assuming that I have two samples at location $x = 2$.

So, whether I use a rectangular window or I use a triangular window like this I can always estimate the probability density function at sample locations where the samples are not originally present and that is what is required for our application. So, all these sample locations where we have the samples they are used for training purpose or for learning purpose. And once your system is learnt, then the probability function density is the learnt using that estimated density probability function then I can go for classification of an unknown sample.

So, this is what we have done in one dimension, what will happen in multi dimension if the feature instead of being a scalar feature it is a feature vector. We have done using the histogram based technique, how I will do it in case with the kernel based or window based technique, so for simplicity I will assume that our window are rectangular windows.

(Refer Slide Time: 42:27)



So, let us define a two dimensional space, so I have feature vectored X which is of the form

$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ where x_1, x_2 are the components of the feature vector X. So, I am using two

dimensional feature vectors and I will put this, use this as x_1 , axis x_1 dimension and the vertical axis as x_2 dimension let us put some grid. So, these are the grids that I have and suppose I have been given samples, let us take some arbitrary samples say one at location, so I will put at this as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10, 0, 1, 2, 3, 4, 5, 6, 7 and so on. So, suppose I have been given these samples of this say one sample over here, one sample here one sample here one sample here.

Let us assume that we have got two samples here, then one sample here, one sample here one sample here, one sample here, these samples are arbitrarily distributed. So, using these samples I have to compute my probability density function using the kernel based technique, we have already seen how to do it using the histogram based technique. In this histogram based technique the entire x_1, x_2 plane will be divided into a number of rectangular bins. For every rectangular bin depending upon the fractions of the samples falling under that bin I will compute what will be the height of the bar in that bin and the height of the bar gives you the probability density estimate.

Here I will use a Kernel based technique or window based technique, so let us assume that we use rectangular windows obviously because this is in 2 dimensional case. So, this rectangular window will actually be a box, so it will be something like this and this box will have certain volume under the probability density surface. And as we have used and as we have done in case of one dimension I have to find out what will be the volume of every such box, volume under every such box. The volume will be 1 upon the total number of samples which are given for probability density estimate. So, if I have total number of samples which is equal to N, then volume of individual box will be obviously be $\frac{1}{N}$.

So, here you see that I have 1, 2, 3, 4, 5, 6, 7 because here there are two samples, 7, 8, 9, 10, 11, 12, 13, so there are total 13 samples, so I have $N = 13$ and as I have 13 samples volume of every box will be $\frac{1}{13}$. So, from this volume I have to find out what will be the height of this box or the height of the rectangular window function. So, to determine the height I have to know that what the base area of this rectangular window, so again if I use the base area the

rectangular window let me assume here that this rectangular window is of base given by let us say 3×3 .

(Refer Slide Time: 48:33)

$$\text{Base of rectangular window}$$

$$\Rightarrow 3 \times 3 = 9.$$

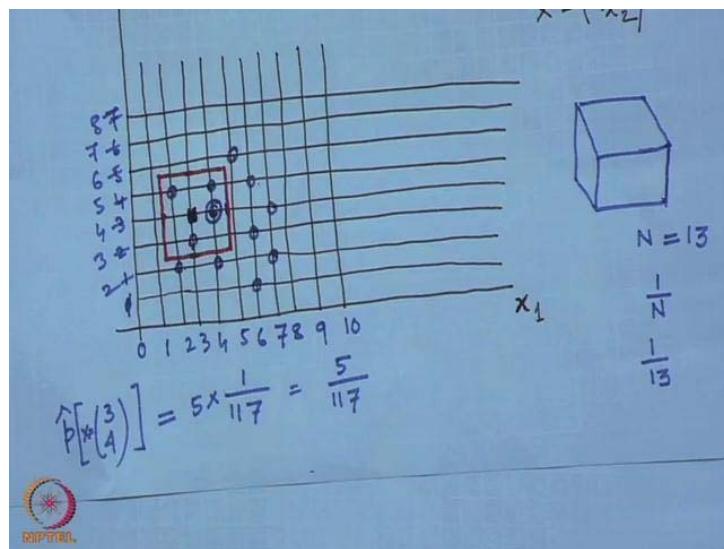
$$9 \times h = \frac{1}{13}$$

$$h = \frac{1}{9 \times 13} = \frac{1}{117}$$

So, the base of the rectangular window is 3×3 , that is area is equal to 9, so I have 9 into height of the box that has to be equal to the volume of the box and this volume we have computed that this has to be $\frac{1}{13}$ because we had total 13 samples. So, I have this $9.h = \frac{1}{13}$,

so obviously the height of the box has to be $h = \frac{1}{13 \times 9}$ which is equal to $h = \frac{1}{117}$. So, I have decided about what my Kernel function is or what my window function.

(Refer Slide Time: 49:50)



Now, from this suppose I have to estimate that what is the probability density at let us assume this location here I do not have any sample what is the probability density at this location. So, you find that this location is 3 sorry here I have made something wrong this has to be one this has to be 2, 3, 4, 5, 6, 7 and 8, so this location this is feature vector which is $\begin{pmatrix} 2 \\ 4 \end{pmatrix}$. So, I have

to find out what is the probability that $x = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, so what I do is at this location I place this box whose base is of size 3x3. So, 3x3 means it will be 1.5 on this side and 1.5 on this side, so this box will span from here to here and here to here.

So, this is the box that I have and now I count that how many training samples are actually falling in this box. So, you find that the total number of samples which are falling in this box is given by 1, 2, 3 and there two samples, so total five samples are falling in this box, so as total five number of samples are falling in this box. So, what I have to do is this probability estimate $p(x)$ at $x = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$ this will be 5 into height of the box as we have computed which is

$$\frac{1}{117}.$$

So, this is $\frac{1}{117}$ which is equal to $\frac{5}{117}$, so this is the probability density this is the estimated probability $\hat{p}(x)$, at location $x = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$. So, this is in the vector space, so how we have been able to do it simply by placing the box and counting the number of samples within the box. It is equivalent to if I place a box over here, place a box over here, place a box over here plus two boxes over here then find out how many of these boxes are actually superimposing at location $\begin{pmatrix} 3 \\ 4 \end{pmatrix}$ as I said this operation is same as convolution operation. Alternatively, I can

place a box at location $\begin{pmatrix} 3 \\ 4 \end{pmatrix}$ and then I can find out that how many samples are falling within the box at location $\begin{pmatrix} 3 \\ 4 \end{pmatrix}$ and those samples multiplied by the height of the individual box, that

gives me the probability estimate at that particular location and that is what exactly I have done. Now obviously as we have done before that this can be done for classification purpose.

So, if I have assuming that these are the samples which are coming from say class ω_1 and I will have another set of samples coming from class ω_2 and I want to classify an unknown data which is may be same location at location $\begin{pmatrix} 3 \\ 4 \end{pmatrix}$. So, by considering only the samples from class ω_1 if I compute what is $p(x)$ at $x = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, and now because now I am considering the samples from class ω_1 , so that will be the class conditional probability density function.

So, it will be $\hat{p}(x)$, at $x = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$ given ω_1 . Similarly, by considering only the samples from class ω_2 I can also estimate what is $\hat{p}(x)$, at $x = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$ given class ω_2 and if the apriori probabilities of classes ω_1 and ω_2 is same. Then out of this $p\left(\frac{x}{\omega_1}\right)$ and $p\left(\frac{x}{\omega_2}\right)$ whichever is higher x will be classified to that corresponding class if the apriori probabilities are not same. They are different, then $p\left(\frac{x}{\omega_1}\right)$ will be weighted by apriori probability $p(\omega_1)$, so this is $p(x)$ within $p\left(\frac{x}{\omega_1}\right) \cdot p(\omega_1)$.

Similarly, $p\left(\frac{x}{\omega_2}\right) \cdot p(\omega_2)$ which is the apriori probability and out of these products whichever is more x the vector, unknown feature vector x will be classified to the corresponding class. So, this we have done in case of one dimensional feature, in case of two dimensional features and the same concept can also be extended in case of three dimensional features in case of four dimensional features, in case of N dimensional features. By simply by extending this concept of window function to three dimensional window functions to four dimensional windows function to N dimensional window function and so on.

And because I cannot visualize this, I cannot draw that on a piece of paper over here I will not discuss this, but the concept we have already explained when we talked about the histogram based technique for probability density estimation. So, it is the same concept which will be extended which can be used for this window based technique for probability density estimation also, with this we come to an end to our discussion on probability density

estimation. And you find that when we have estimated this probability density function we have not assumed any parametric form of the probability density function.

So, unlike in case of say normal distribution where you have mean and standard deviation which are the parameters of the probability density function, I am not used any such parameters. So, the probability density that you estimate is the non-parametric density and in many cases because we are not aware of what will be the parameters or what form the probability density function is going to take, this non parametric estimation of the probability density function is extremely useful, so with this I come to the end of our discussion of probability density estimation next class we will start some other topic.

Thank you.

Pattern Recognition and Application
Prof. P.K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 16
Dimensionality Problem

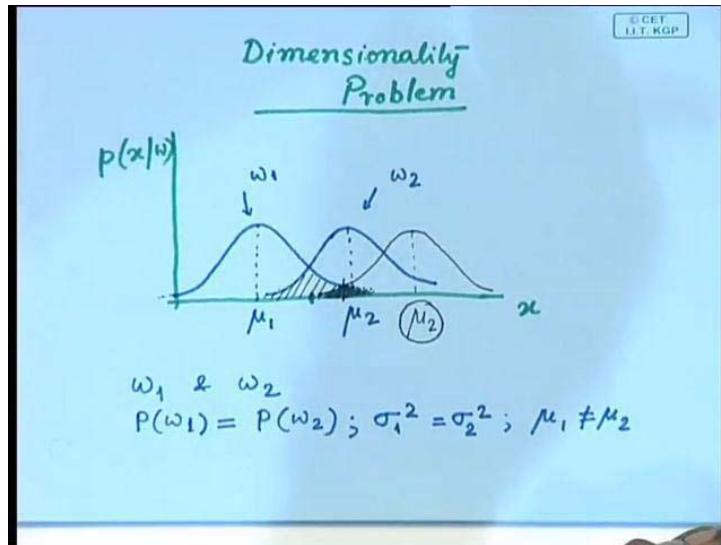
Good morning. So, in the last class we have talked about that given the probability density function which has a known parametric form, where it is not necessary that the probability density function has to be Gaussian or normal. But if it is having any other form but which can be parameterized, and that is we have the known parametric form of the probability density function. Then it is possible to estimate the value of the parameters from the training samples and the kind of estimation that we have talked about is maximum likelihood estimation of the parameter vector. That is the value of the parameter which is best supported by the training samples those are given.

Simultaneously we can also have non parametric estimation of the probability density function, because in many cases I do not need to know what is the exact form of the probability density function? But what we have to find out is $P(x/\omega)$ or $P(\omega/x)$. So, given a value of x , I have to estimate what is the value of the probability density probability for the corresponding value of x . So, I need not have the knowledge of the entire parametric form of the probability density function, rather if I know what is the value of x , for a value of x and this value of $p(x)$ has to estimate from the given set of samples. That is what is called non parametric form. I can have both types of estimation whether it is in the non-parametric form or in the non-parametric form.

Now, today I will discuss about a problem, which is dimensionality problem. That is, we said that the feature vectors are of dimension d or every component of the feature vector represents certain aspects of the pattern. Or it gives some description of the pattern which we discussed in our earlier classes when we talked about feature extraction processes. So, when I have d numbers of components of the feature vector, each of these components give you some characteristics of the pattern. And when all this different characteristics or d number of characteristics are taken together, I can have some sort of discriminating power given by this feature vectors among different patterns. So, today we will discuss that, what is the problem if we have more number of features in the feature

vector or if the dimensionality of the feature vector increases. So, that is what is the dimensionality problem? Now, let us go to the Baye's decision theory or minimum error rate classification where we said that if I know the probability density function.

(Refer Slide Time: 03:16)



So, let me assume that we are having two classes. One is class ω_1 and the other one is class ω_2 . So, I consider two classes, ω_1 and ω_2 and I also assume that say $p(\omega_1)$, that is apriori probability of class ω_1 and the apriori probability of class ω_2 , $p(\omega_2)$ they are same. So, by assuming this if I plot the probability density functions $P(x/\omega_1)$ and $P(x/\omega_2)$. And let me also assume that the variance of these two classes are also same. That is $\sigma_1^2 = \sigma_2^2$.

So, I also assume that along with this, so $p(\omega_1) = p(\omega_2)$. I also assume that $\sigma_1^2 = \sigma_2^2$ for simplicity. That means both the probability density functions, they have the same size and same shape, but the difference is the means are different. That is $\mu_1 \neq \mu_2$, obviously because μ_1 and μ_2 are also same that means I have the same probability density functions, there is not different. So, given this if I plot $P(x/\omega_1)$ and $P(x/\omega_2)$, so let me plot in this side $P(x/\omega)$ and suppose the probability density functions are something like this.

So, this is one, this extends infinitely and the other one, something like, they have the same form. So, they have the same shape and same size, only thing is one shifted from the other because the means are different, μ_1 is different than μ_2 . So suppose this is for class ω_1 and this is for class ω_2 where this is μ_1 and this is μ_2 . Then we have said earlier when we talked about Bayes decision theory, that whenever the two probabilities are same then that is my boundary. So, for all values of x which are greater than this, that is $P\left(\frac{\omega_1}{x}\right) > P\left(\frac{\omega_2}{x}\right)$, obviously it is nothing but $P\left(\frac{x}{\omega_1}\right) \cdot P(\omega_1)$, which is greater than $P\left(\frac{x}{\omega_2}\right) \cdot P(\omega_2)$. That was our Bayes decision theory, but because here we are assuming that $p(\omega_1) = p(\omega_2)$.

So, effectively it comes to be $P\left(\frac{x}{\omega_1}\right) > P\left(\frac{x}{\omega_2}\right)$. So, for all values of x lying on the right hand side of this point, we will decide that it belongs to the class ω_2 . And for all values of x lying on the left hand side of this point we will decide that it comes to class ω_1 . Then we have also said when we talked about the Bayes or minimum error rate classification, that the error in taking this decision because whenever I decide in favor of class ω_2 , that is a finite probability. It may belong to class ω_1 as well.

So, the error involved in taking this decision and that we said earlier is nothing but the area under this curve and it extends indefinitely. So, this is what is the error in taking decision, whether I decide in favor of ω_1 or I decide in favor of ω_2 . So, this area under this probability curve, that gives me the total amount of error. Now, we find that if I shift this μ_2 , that is instead of μ_2 being here, if μ_2 if this class ω_2 probability density function is something like this.

So, suppose this becomes my μ_2 , if this is the μ_2 then this is my decision boundary and if this is the decision boundary then the error will be reduced to this. So, you find that earlier my error was bounded under this curve, now the error is bounded under this curves. So, definitely the total error in the earlier case is higher than the total error I get in this case when the separation between μ_1 and μ_2 becomes large. So when the separation between the two means μ_1 and μ_2 is low then the total error increases. If the separation between the two means μ_1 and μ_2 is large then the total error decreases.

So, that clearly indicates what are the features, which are best suitable for classification purpose. It is the features whose means for different classes are widely apart. Those are the

features best suited for classification. If the means are quite near, very near, then those features does not give you much of discriminating problem, though even if the means differ by some amount. However small it is, they still have some discriminating power, but the discriminating powers of those features, where the means are closer, means for different classes are closer is much less than the discriminating power of those features whose means are widely apart.

So, usual practice in pattern recognition problem is that you start with some features find out what is the performance of classification. If the performance of classification, that means misclassification rate is not satisfactory, that is the performance is poor. You try to incorporate more and more features because I said if the features means of the features are different, they are not same. They have some discriminating power, but the extent of the discriminating power depends upon what is distance between the means. As the distance increases, the discriminating power increases. As the distance decreases, the discriminating power also decreases. So if I find with certain number of features the performance of classification is not very good, you try to incorporate more and more features.

If I try to incorporate more and more features, then effectively what I am doing is the dimensionality of the feature vectors I am going to increase. So, the value of d will increase as we incorporate more and more features in the feature vector, but that leads to another problem. Should we go on increasing the dimensionality of the feature vector without any control? So That is what is the dimensionality problem? So, let us see what kind of problem that we face if we go on increasing the dimensionality of the feature vector, ok?

(Refer Slide Time: 11:50)

The image shows handwritten mathematical notes on a whiteboard:

$$g(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma_i| + \ln P(\omega_i)$$

$$\mu_i = \frac{1}{n} \sum_{k=1}^n x_k \approx (nd)$$

$$\Sigma_i = \frac{1}{n} \sum_{k=1}^n (x_k - \mu_i)(x_k - \mu_i)^T \approx \underline{\underline{(nd^2)}}$$

$$O(nd^2) \quad O(cnd^2).$$

A small logo in the top right corner reads "© GET I.I.T. KGP".

So, you find that earlier we said that function $g(x)$ or discriminating function of different classes is $g(x) = -\frac{1}{2}(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma_i| + \ln P(\omega_i)$. So, this was discriminating function for the i^{th} class, right?

Now, see that what is the amount of computation involved in this particular case, out of this we said that this term $-\frac{d}{2} \ln 2\pi$. This is class independent, this does not depend upon the class. So, we can as well remove it from the definition of discriminating function, because for all the classes this value will remain. So, this does not give you any discriminating power, but the other terms they really discriminate among different classes. Now, from here when I try to compute the value of μ_i , you assume that I have n number of samples, training samples belonging to class ω_i .

So, when I compute the value of μ_i , value of μ_i is nothing but $\mu_i = \frac{1}{n} \sum_{k=1}^n x_k$, right? So, I have

to incorporate n number of summations followed by one division, and this x_k , that is the feature vector of dimension d for every component I have to compute n number of summations, that means the total number of summations that I have to perform is nxd , right? And I can assume that for this division it takes constant time.

So, the computational complexity to find out the mean vector is $n \times d$, which is dimension dependent. It depends upon the number of vectors, at the same time it depends upon d which is the dimensionality of the vector, right? Now, coming to the other term, computation of this

Σ_i or Σ_i^{-1} , ok? You find that this Σ_i , the definition of $\Sigma_i = \frac{1}{n} \sum_{k=1}^n (x_k - \mu_i)(x_k - \mu_i)^t$, where k varies from 1 to n , right?

So, this is vector of dimension $d \times 1$ and this is a vector $1 \times d$, right? This is a vector of dimension $d \times 1$. This is the vector of dimension $1 \times d$, when I compute I get a matrix of dimension $d \times d$. That is d^2 and following similar manner if I try to find out is the amount of computation will be involved in this case for the computation of the Σ_i . You will find that these steps are the computation of the order of nd^2 . So, to find the mean vector I need a computation of dimension $n \times d$ to find the covariance matrix, I need a computation of the order nd^2 .

Similarly, over here this also takes a computation of order nd^2 . Of course, this one is constant, not constant. It depends upon the value of n because when I want to compute ω_i , I have to take n number of samples in picture. So, this is of the complexity n . So, overall you find the dimensional complexity of this discriminant function because this is the highest one. So, I can take that the overall complexity is of the order nd^2 . Not only that I have seen number of classes, right? For every class I have to compute this discriminating function and then I have to find out which one is maximum.

So, before I get the maximum, I have to compute the discriminating function for each and every class and have c number of classes. For every class, I have a computational complexity of n into d squared. So, for c number of classes the total computational complexity will be of order cnd^2 . So, here you find that the complete computational complexity linear in terms of c , that is the number of classes linear in terms of n . That is the number of samples in the class, but it is quadratic or square in terms of the dimensionality, right?

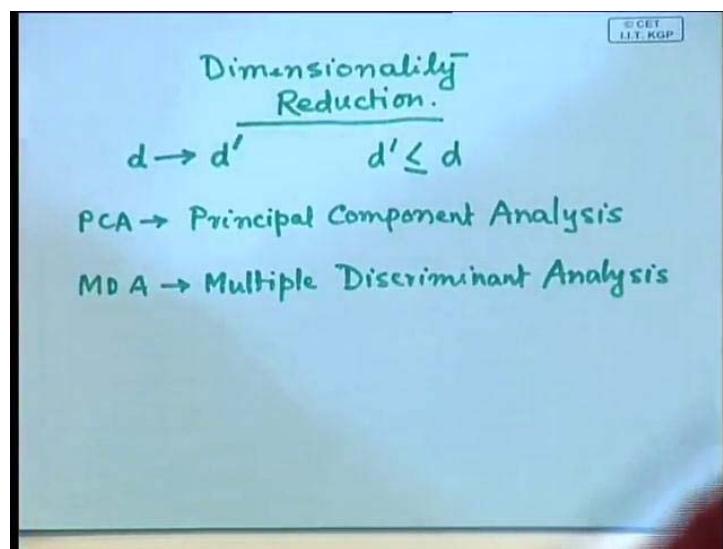
If the dimensionality is increased by a factor of 2, the computational complexity is raised by a factor of 4. If it is increased by a factor of 3, computational complexity becomes 9 times, right? So, this dimensionality of the feature vector is very, very crucial determined

how efficient your classifier will be? So, it is not always good that you go on increasing the feature of the dimensional vector by incorporating more and more features. It is true that if I incorporate more and more features, the classification performance might be better, but the same time the classifier becomes inefficient.

So, we have to think of the ways in which the dimensionality of the feature vectors can be reduced. That means whether it is possible that initially I start with feature vectors of higher dimension, but whether it is possible to reduce the dimensionality, but still maintaining the separability power of the selected features. So that means what we have to do is given a feature vector of higher dimension, I have to take the projection of those higher dimensional feature vectors on to a lower dimension. But this projection has to be in such a way that the space on which, the space your dimension on which I am projecting this feature vectors, that space should be as orthogonal as possible.

That means the different features they should not be correlated because one of the factors when I incorporate, more and more features are arbitrarily. I do not guarantee those features will be orthogonal, that means one feature may have some correlation with another feature. But when I try to project onto a lower dimensional space, I should make sure that this lower dimensional space is orthogonal. That means whatever feature I try to capture, one feature should not depend on the other feature. So, these features should be independent. So, coming to this concept of projection with the aim to reduce the dimensionality of the feature vectors, I can have two types of projections or two types of concepts.

(Refer Slide Time: 20:44)



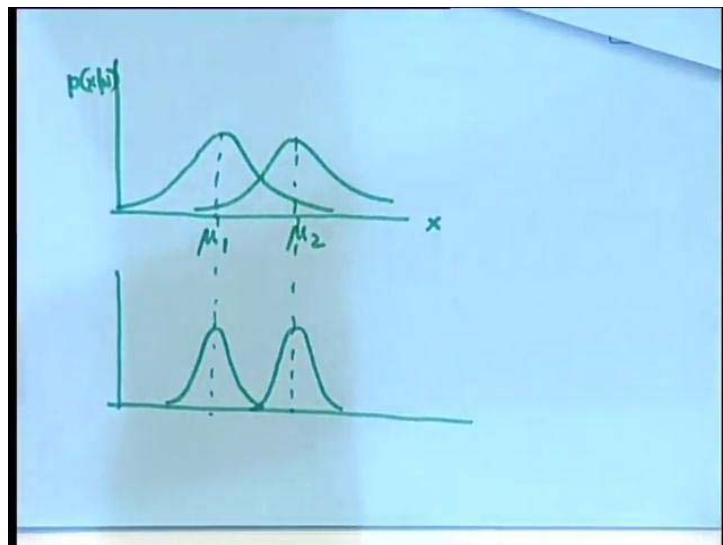
So, what we are discussing now is the reduction of dimensionality. That means initially I have a feature vector of dimension d . I want to reduce this to feature vectors dimension of d' , where $d' \leq d$. Obviously, I do not want d' to be larger than d . At most it can be equal to d , but I target for $d' < d$ so that the dimension is reduced.

Now, while I try to do that there are two approaches, one is PCA or principal component analysis. The principal component analysis or PCA approach is that, you try to project your higher dimensional feature vectors on to a space of lower dimension in such a way that, in the lower dimension whatever feature vector I get or projected vector that I get, that best represents the initial feature vector, that I have to get the original feature vector, that I have to get.

So, when I say best represents, it is the best representation in terms of least squared error. That is the list square error between your initial feature vectors or original feature vectors and the reduced feature vectors that should be minimum. And the other approach is called MDA that is multiple discriminant analyses. So the aim of multiple discriminant analyses is, after all why we are doing or trying to manipulate all these feature vectors, because we want to separate among the feature vectors which belong to different classes.

So, the multiple discriminant analyses, what it tries to do is when take a projection on to a different space, on to a different feature space, it tries to separate the mean vectors of different classes and at the same time tries to make the samples belonging to the same class more compact. That means within class scatter is reduced. Whereas, between classes scatter is increased, ok? That is quite natural because here what we said is coming to this diagram or let me redraw this diagram.

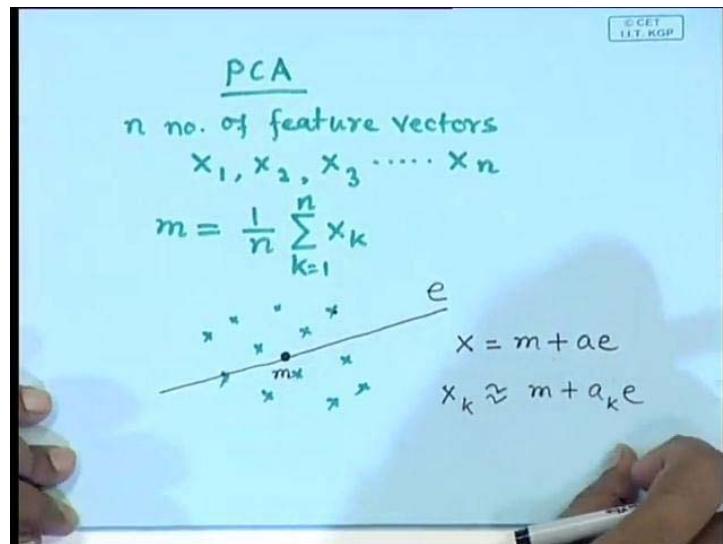
(Refer Slide Time: 24:13)



This is the x , this is $P(x/\omega)$, ok? So, suppose the distribution is something like this. So, this is μ_1 and this is μ_2 , we said that if we increase the distance between the μ_1 and μ_2 . My classification error will be less, that is one and at the same time if I try to reduce the variance of the samples within a class, that means keeping μ_1 and μ_2 same in the same location. So, this remains μ_1 and this remains μ_2 , but now I want to have a distribution something like this. So, you find that the distance between the means, μ_1 and μ_2 remains the same, but within class scatter or the variance of the samples belonging to the same class are reduced. Then also my classification will be better, that is error rate will be less.

So, this MDA approach or multiple discriminant analyses what it does is, it tries to increase the distance between the means. That is the inter class scatter, that it tries to increase and at the same time it tries to reduce the within class scatter or tries to reduce the variance of the feature vectors on to the projected space. So, you benefit in both ways, one is increase the distance between the means and simultaneously reduce the variance within the class so that the overall error is minimized. So, these are two different approaches, one is principal component analyses, other one is multiple discriminant analyses, ok? Today, what we will discuss is the principal component analysis approach.

(Refer Slide Time: 26:36)



So, we will talk about the PCA or principal component analyses. So, our problem is like this, that we have been given n number of feature vectors, every feature vector is of dimension d .

So, I have n number of feature vectors. The feature vectors are x_1, x_2, x_3 up to x_n , each of the feature vector is of dimension d , right? Obviously, the mean of these set of feature vectors is given by $m = \frac{1}{n} \sum_{k=1}^n x_k$. That is the mean of the feature vector.

Now, let us try to see that initially I have the dimensionality of the feature vector which is equal to d . Let me take extreme case that I want to represent these by a zero dimensional feature vector, right? So, if I try to represent this by a zero dimensional feature vector that means within feature vectors I do not have any variance. So, that is nothing but this mean. So, this entire set of feature vectors is represented by mean vector, but while doing that what I will lose is the variability within the data which is also not good, right?

So, instead of zero dimensional features, let us assume that I want to represent this dimensional feature vectors by 1-d feature vectors, pardon meaning the feature, but the entire n number of features is represented by only one feature vector. While doing that I lose the variance because I have a single representation. Now, what I am saying is I do not want to do that I represent this d dimensional feature vectors by one dimensional feature vector. I will still have n number of feature vectors; the number of feature vectors will remain the same.

So, there will be variability, but the dimension of every feature vector instead of d it will be 1. So, now what I am trying to do is a vector getting mapped to a scalar, because I am reducing the dimensionality to 1. Let me take step by step approach so that what should be such best type of projection. If I want to project in one dimension and then whether I can project on to a multiple dimension. If I want to project on to multiple dimensions what should be nature of such space? So, if I want to project on to a single dimension. So, suppose I have the set of points something like this, they are arbitrarily distributed. I have n number of such points and the mean m is somewhere over here, ok?

So, what I do is, if I want to represent this d dimensional feature vectors by a one dimensional feature vector. That means I have to take, I have to represent this feature vectors somehow on to a line. The d dimensional feature vector will somehow mapped to different points on a line, right? So, let us take a line passing through the mean of this feature vectors and let me assume that a unit vector in the direction of this line is given by e , right? So, I am in space x , within the space I have a line passing through the mean of the sample given feature vectors and the direction of this line I am saying it is e , right?

So, when I have done this, you find this the equation of this line will be given by $x = m + ae$, because it passes through m . That is the mean of the given feature vectors and equation of this line if $a = 0$. Then $x = m$, that means I am at this point m , right? For different values of a , I am moving from m in the direction of e . That means I am moving along this line.

So, this equation of this straight line in the direction of e , where e is an unit vector passing through the mean m is given by $x = m + ae$, right? This a now represents positions of different points on this line, given this I can represent a point x_k on this line and that x_k , point x_k I can represent by $x_k = m + a_k e$. We will see what will be the value of a_k , the best possible value of a_k . So for different points I will have different values of this a_k . So, this a_k represents that a point x_k , where is it mapped on this line. Now, while doing this definitely I have corporate some amount of error, because x_k is of dimension d . I am representing this x_k by a point on the line. So, obviously while doing so I have incorporated some amount of error, and what is that error? This is nothing but $m + a_k e - x_k$, x_k is the original one and this is the one that I have represented. So, what I can do is, for taken all these lines together or all these points together I can define an error function or a criteria function, ok?

(Refer Slide Time: 33:27)

The image shows a handwritten derivation of the error function J on a whiteboard. The derivation starts with the expression for J as a sum of squared distances between the line $(m + a_k e)$ and the points x_k :

$$J(a_1, a_2, \dots, a_n, e) = \sum_{k=1}^n \| (m + a_k e) - x_k \|^2$$

This is then simplified using the properties of dot products:

$$= \sum_{k=1}^n \| a_k e - (x_k - m) \|^2$$

$$= \sum a_k^2 \| e \|^2 - 2 \sum a_k e^t (x_k - m) + \sum \| x_k - m \|^2$$

Finally, the derivative of J with respect to a is set to zero:

$$\frac{\partial J}{\partial a} = 0$$

So, that error function or the criteria function is nothing but we represent it as this is the initial way, you represent it by J . Obviously, it will be a function of a_1, a_2 up to a_n and it will be also the function of direction of the line which is e , right? This is nothing but

$$J(a_1, a_2, \dots, a_n, e) = \sum_{k=1}^n \|m + a_k e - x_k\|^2$$

So, this is the squared error within summation, this

becomes sum of squared error, right? What we try to do is, we try to minimize this sum of squared error in terms of a , right?

So, I want to find out that what are these a 's or what are the positions of the straight line by which my d dimensional vector should be represented, assuming e to be fixed. I do not touch e for the time being, that is the direction of the line to be faced. So, given a line how these points are to be mapped on to the different points on the line. How the feature vectors have to be mapped on to different points on the straight line so that your sum of squared error will be minimized. So, this expression now I can write by slight reorientation. This can be written as $\sum_{k=1}^n \|a_k e - (x_k - m)\|^2$. This x_k is same as the previous

one, I just, I have simply rewritten this. It is same x_k , I am representing as $x_k = m + a_k e$. Otherwise, how do I get the error? This x_k is my original point and this is the projected point.

So, the difference between the original point and the projected point, that is my error. I am trying to estimate the error, right? So, this is the same as my original x_k . So, this if this square I expand, and it simply becomes $\sum a_k^2 \|e\|^2 - 2 \sum a_k e' (x_k - m) + \sum \|x_k - m\|^2$. I am not writing the limits of summation. The limit is k varying from 1 to n . So, this is my sum of squared error. What I want to do is I want to reduce, minimize the sum of squared error by varying the values of a .

So, what I have to do for that is take the differentiation of this with respect to a , equate that to zero, solve for values of a effectively. What I want to do is, I want to take $\frac{\partial J}{\partial a} = 0$. And by solving this I get the values of different a , that comes later. I said I will go stage by stage. Initially I am saying that assuming the direction of the line to be fixed, let us not try to handle all together. Dimensionality of e is same as d because it is a unit vector in the line in d dimensional space. It is a vector in d dimensional space. So, obviously the dimensionality of e will also be d . If I want to represent a vector in 3 dimensional space, the dimensionality of the vector is obviously 3, ok?

(Refer Slide Time: 38:38)

$$\frac{\partial J}{\partial a} = 2 \sum a_k - 2 \sum e^t (x_k - m) = 0$$
$$a_k = e^t (x_k - m) \leftarrow$$

So, when I take the differentiation of this with respect to a , what I get is effectively $\frac{\partial J}{\partial a}$

nothing but $\frac{\partial J}{\partial a} = 2 \sum a_k - 2 \sum e^t (x_k - m)$, ok? So, if I take the differentiation of this with

respect to a , then what I get is if I take the differentiation of this. And what I forgot is the $\|e\|=1$, because it is unit vector in the direction of the line.

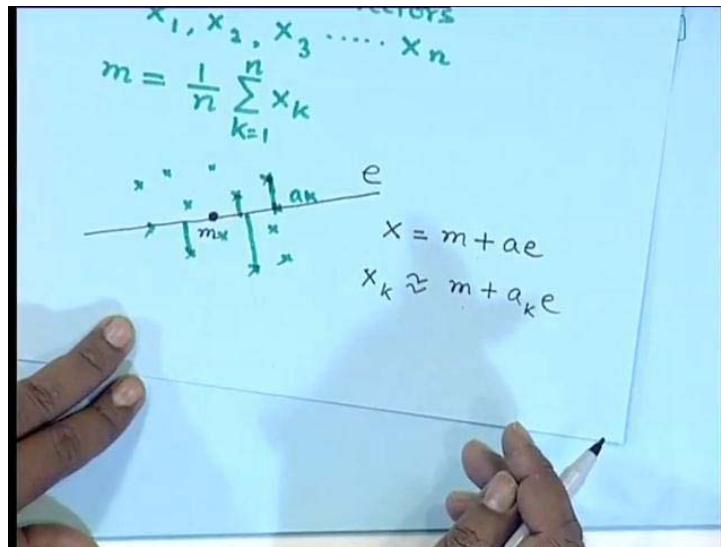
So, $\|e\|=1$. So, effectively what I have is $\sum a_k^2$. Take the differentiation that becomes

$2 \sum a_k$, take the differentiation of this, I get $\sum e^t (x_k - m)$ because it is $\sum a_k e^t (x_k - m)$. If I take to differentiate with respect to a . So, differentiate this with respect to a , it becomes 0, right? So, after differentiating this term, assuming by making use of the fact that $\|e\|=1$, I get this output, right? I have to equate this to 0 for minimization.

So, when I equate this to 0, I get simply $a_k - e^t (x_k - m)$, right? Equating this to 0, I get $a_k = e^t (x_k - m)$. Now, what is this? e is the unit vector in the direct from the line, right? So, this simply says that a_k is nothing but orthogonal projection of x_k on to a line, the direction of p passing through the mean m . It is just orthogonal projection of point x_k on to a line in the

direction of p passing through the mean point m, right? So, simply what we get is when I have these sets of points, this a_k represents these projected point on to this line.

(Refer Slide Time: 41:20)



And, how do I complete this? This is nothing but projections of different a_k 's, projections of different a_k 's on to this line and these projections are nothing but orthogonal projection and that gives the position a_k . So, the best kind of representation in one dimension is to project the d dimensional feature vectors, take the orthogonal projection of the d dimensional feature vectors on to a line passing through the centroid, ok?

So Now, you see that our assumption was we started with a given line, whose direction was known as e and for the given line the best type of projection is taken orthogonal projections on to that line. So, given a line and the best representation is orthogonal projection onto the line. Now, which given line is best, that is the question that he has raised, so far we have considered that our line is given. Now, which particular direction of the line is best, that is what direction is best? Let us try to answer that. So, for doing this I start with the same one that.

(Refer Slide Time: 42:48)

The image shows a handwritten derivation of a function $J(e)$. It starts with the expression $J(e) = \sum a_k^2 - 2 \sum a_k e^t (x_k - m) + \sum \|x_k - m\|^2$. This is then rearranged into $= - \sum [e^t (x_k - m) (x_k - m)^t e] + \sum \|x_k - m\|^2$. Finally, it is simplified to $= - e^t [\sum (x_k - m) (x_k - m)^t] e + \sum \|x_k - m\|^2$.

Now, I have to find the direction of the line, that is e . Now, I have determined that to find out what it should be orthogonal projection. Now, I have to determine that which direction of the line, which line will be best, that means what should be best e . So, I write this J , the criteria function as a function of e , you remember? This e is not error, I am not saying this as error. This is the unit vector in the direction of the line. So J $J(e) = \sum a_k^2 - 2 \sum a_k e^t (x_k - m) + \sum \|x_k - m\|^2$. When I say differentiation, it is gradient because in the vector space I cannot differentiate in the vector space. I have to take gradient which are nothing but components of that gradient comes as partial differentiation with respect to individual components.

So, I had $J(e) = \sum a_k^2 - 2 \sum a_k e^t (x_k - m) + \sum \|x_k - m\|^2$. I am not writing the limit of summation. Now, this a_k we have found that it is the same $e^t (x_k - m)$. So, this I can simply write as this one. So, this will be $\sum a_k^2 - 2 \sum a_k e^t (x_k - m) + \sum \|x_k - m\|^2$. So, I can rewrite this expression as $- \sum [e^t (x_k - m) (x_k - m)^t e] + \sum \|x_k - m\|^2 ..$

. If I break that, I can write this as $- \sum [e^t (x_k - m) (x_k - m)^t e]$, it is in the matrix expression. So, this $+ \sum \|x_k - m\|^2$. Now, this e is independent of k . So, this e terms I can take out of summation expression.

So, this I can write $-e^t \sum [(x_k - m)(x_k - m)^t] e + \sum \|x_k - m\|^2$, because e is independent of k . So, I can take this e out of the summation expression, because this summation is over k , k varying from 1 to n , right? So, it remains the same, right? Now, you find what is this term, $\sum [(x_k - m)(x_k - m)^t]$.

(Refer Slide Time: 47:37)

$$\Sigma = \frac{1}{n} \sum_{k=1}^n (x_k - m)(x_k - m)^t$$

If you remember from our earlier expression, what is our covariance matrix Σ , $\Sigma = \frac{1}{n} \sum_{k=1}^n (x_k - \mu)(x_k - \mu)^t$. Take the summation $k = 1$ to n and here you find that this term is nothing but a scaled version of this term. If I multiply the covariance matrix by n , which is the number of samples over which I am trying to compute the covariance matrix. I get this term and this is what is called scatter matrix.

So, it simply represents that how the data is spread in the space because you remember that from your one dimensional Gaussian function a variance is more, the curve becomes flat if the variance is less, the curve becomes sharp. That means the point is distributed over a smaller space, if the variance is small, the points are distributed over a larger space if the variance is large, same is for covariance matrix. The covariance tells you shape as well as size, that means to what extent that data is distributed and whether the data distribution is oriented in some particular directions. Only that is what is captured by covariance matrix, same is the case of scatter matrix. Scatter matrix gives you the same information, but with a

scale factor of n. So, if I multiply co variance matrix by the number of samples n, I get the scatter matrix, right?

(Refer Slide Time: 49:40)

The image shows handwritten mathematical notes on a whiteboard. At the top, the formula for the scatter matrix $J(e) = -e^T S e + \sum \|x_k - m\|^2$ is written, with a note below it stating "maximization of $e^T S e$ ". Below this, the text "Lagrangian \rightarrow " is followed by the expression $u = e^T S e - \lambda(e^T e - 1)$. Then, the derivative is shown as $\frac{\partial u}{\partial e} = 2Se - 2\lambda e = 0$. A note below states $Se = \lambda e \Rightarrow$ Eigenvalue Expression. A hand is visible pointing at the board.

So, this expression now becomes this $J(e)$, see if I simply write this, rewrite this it becomes $J(e) = -e^T S e + \sum \|x_k - m\|^2$, S is the scatter matrix. Now, what is our aim? I want to minimize this $J(e)$ by varying e , this should be within bracket, as a function of e , $J(e)$ by varying e and you find this term is independent of e . So, if I want to minimize $J(e)$, what I have to do is, I have to maximize this. It is a variability sign, minimization of $J(e)$ effectively means maximization of $e^T S e$.

So, if I maximize this $e^T S e$, effectively I am minimizing this criteria function $J(e)$. So, what I have to look for is the maximization of this $e^T S e$ and for this maximization, if I make use of Lagrangian. So, I can have an expression of the form $u = e^T S e - \lambda(e^T e - 1)$ So, I have to maximize this using Lagrangian, subject to the constraint $\|e\| = 1$.

So, to maximize what I have to do is I have to take the differentiation of this with respect to e . So, what I have to compute is $\frac{\partial u}{\partial e}$ and if compute $\frac{\partial u}{\partial e}$ then over here I simply get expression $\frac{\partial u}{\partial e} = 2Se - 2\lambda e$. Differentiate this with respect to e , I get $\frac{\partial u}{\partial e} = 2Se - 2\lambda e$. I have to equate this to 0, right? If I equate this to 0, I get simple expression $Se = \lambda e$. What is this expression? This is an Eigen value expression.

So, this simply an Eigen value expression, where the vector e is an Eigen vector of S and what is λ ? It is the corresponding Eigen value. Now, you remember I have to maximize. Our aim is to maximize $e^T S e$, right? where $S e = \lambda e$, that means I have to maximize $e^T \lambda e$. Our constraint is $\|e\|=1$. So, if I want to maximize $e^T S e$, that means I have to take the maximum value of λ , that effectively says that when I am considering the direction of the line in the direction of the vector e . From here I find that this e is nothing but the Eigen vector and which Eigen vector I have to consider?

The Eigen vector corresponding to the maximum Eigen vector value and by using whatever value of a , I get because my ultimate aim is I have to find out to get the value of a_k , that is what represents the point. So, that value of a_k is called the principal component. That is why it is principal component analyses, people also call it K-L transformation.

Now, you remember how this K-L transformation came into picture? You had the co variance matrix, take the Eigen values of the Eigen vectors of the co variance matrix. Corresponding to the largest Eigen values, from that you form a transformation matrix and the transformation you get is the K-L transformation. That is not like this and there we said whenever we talk about the K-L transformation, we talk about principal transformation analyses and why it is principal component. This is the reason it is principal component.

So, here you find that if I take only one Eigen vector, corresponding to the largest Eigen value, then the corresponding principal component is a scalar one and effectively my d dimensional feature vector I am converting to one dimensional feature vector. Now instead of considering only single Eigen vector, if I consider multiple Eigen of vector corresponding to largest Eigen values.

So, instead of considering only one Eigen vector if I consider, d' number of Eigen vectors, $d' \leq d$. Corresponding to d' number of largest Eigen values, then take the orthogonal projection of your feature vector, d dimensional feature vector on to those Eigen vectors. The corresponding principal components or the projection values, projections that you get that gives you d' dimensional feature vector and because my scatter matrix is real and symmetric the Eigen vectors are orthogonal. That means, the principal components that I get they are also orthogonal, and because they are orthogonal one is uncorrelated from the other, ok?

So, if I go for this principal component analyses, I can reduce the dimensionality. Simultaneously, I also ensure that my d' dimensional feature vector that I generate those feature vectors will be where the components will be uncorrelated. That means every component of the feature vector gives you independent information that no other component will give. So, let us stop here today. We will continue with MDA approach in the next class.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 17
Multiple Discriminant Analysis

Good morning. So, in the last class we started our discussion on the problem involved with dimensionality of the feature vectors, and we have said that when the feature vector dimension is quite large then the computational complexity becomes very high. Because computation of the discriminant function for different classes, for each of the classes is proportional to d^2 , where d is the dimension of the feature vector. And when we have c number of such classes then obviously the amount of time that has to be spent to decide the class, we have to compute the discriminant function for each and every class.

So, the total computation time becomes of the order of c into d square and you can easily understand that in most of the real life situations the dimension of the feature vectors becomes of the order of 100 or 100 or so. And in that case what will be the amount of computation involved to classify an unknown feature vector, because for every unknown feature vector, we have to compute the discriminant function corresponding to each and every class. And then we have to decide to which class that unknown feature vector has to be classified, based on which particular discriminant function gives the maximum value.

So, in the last class we have discussed about this dimensionality reduction problem. And we have said PCA or Principle Component Analysis is one of the techniques by which the dimension of feature vectors can be reduced. Because the principle component basically gives you those principle components which are most useful are most useful to describe the feature vectors.

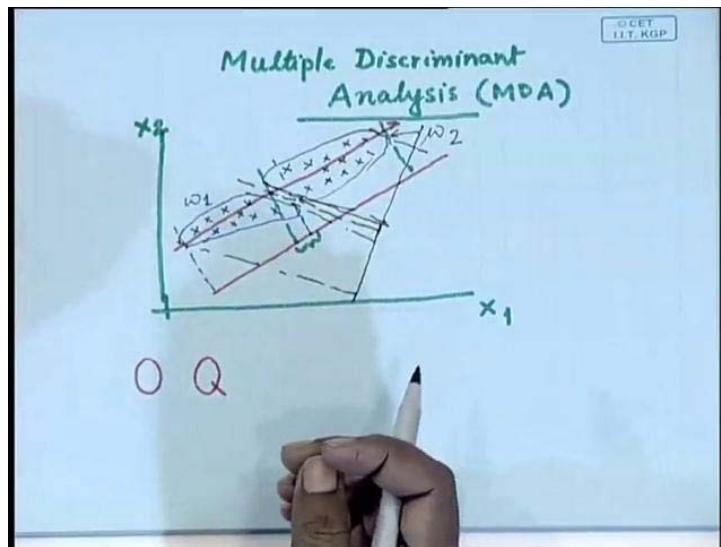
So, though the principle component analysis that gives you the feature that reduces the dimensionality of the features by taking projection on lower dimensional space either in single dimension or in multiple dimension. And we have seen that those directions are nothing but the directions of the Eigen values of the scatter matrix or co-variance matrix of the data set. And we take those Eigen vectors corresponding to the maximum Eigen values.

So, the error while doing the projection is minimized. Now, while doing so these feature vectors in lower dimension that we get they are the best representations of the original feature

vector, in terms of the squared error sense. We ensure that the sum of squared error when we project that higher dimensional feature vectors onto the lower dimension following principle component analysis, the sum of squared error will be minimum. However, it does not necessarily mean that those projections will be the best projections for separating different classes, because when we go for classification we have to ensure that the feature vectors belonging to the different classes they are well separated.

So, while the principle component analysis tries to find out the projection directions which will give you best representation of the data set in minimum squared error sense. But MDA or Multiple Discriminant Analysis tries to find out the projection direction so that if I take projections in those directions, I try to ensure that the data belonging to different classes or the sample vectors belonging to different classes, they are well separated in that projected space. So, today what we are going to discuss is this multiple discriminant analysis or MDA.

(Refer Slide Time: 04:12)



So, this is also a projection technique, but when we try to find out the projection directions we try to ensure that the data in the projected space they are well separated. Now, why the principle component analysis does not guarantee you that sort of separation? Let us just take a sample situation, say something like this. Suppose, I have two dimensional feature vector, one is feature component X_1 , the other feature component is say X_2 and the data distribution, let us assume is something like this.

So, this is the data belonging to this space. The training sample that you get that belongs to one class. Let us say this is class ω_1 , and the other set of training samples which are given

they may be something like this. Suppose, this set of samples belong to class ω_2 . Now, when you do principle component analysis, the principle component analysis takes the projections of data onto a direction which are aligned with the Eigen vectors. So, in this case the direction of the Eigen vector having corresponding to the maximum Eigen value will be something like this. Because this is the dimension direction in which the spread is maximum.

So, on this if I take the projection, let me draw the diagram somewhere over here. So, if I take the projection you find that this set of data will be projected into this space. Whereas, the data classified as labelled as belonging to ω_2 will be projected into this space. So, as a result you find that I have region where the data both from class ω_1 and class ω_2 they are intermingled. That means taking this example we can say that when I take the projections along the directions of principle, along the directions of Eigen vectors, it does not necessarily mean that the data will be well separated in the projected space.

Whereas, if I take the projection direction like this, suppose I take this as the projection direction we have discussed earlier that for best representation projection should be orthogonal projection. So, if I take this projection direction in that case we find that this set of data will be projected into this space. Whereas, this set of data will be projected into this space and now these two sets of projected data, they are well separated in the projected space. So, while principle component analysis tries to find out this direction, this projection direction, the multiple discriminant analysis tries to find out this projection direction.

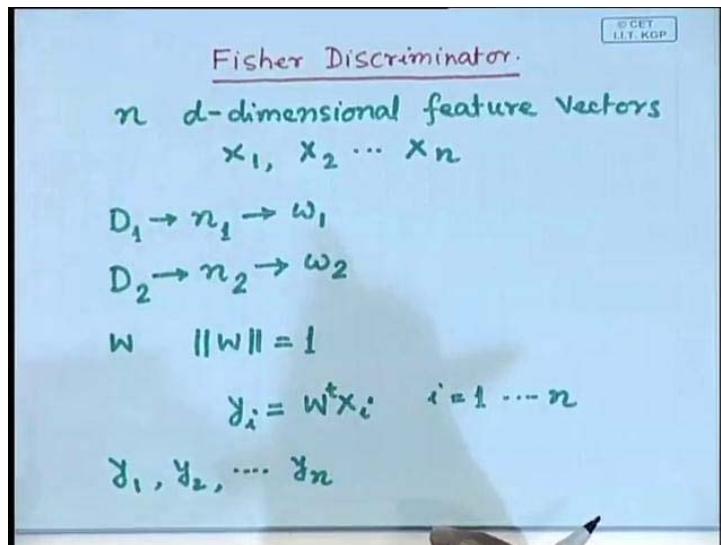
So, what does it practically mean? Let us assume that we have got two printed characters, one is say printed character O, other one is printed character Q. So, when I take the principle component analysis, the principle components actually tries to capture the gross features of the patterns and while doing so it may try, it may ignore the feature, this subtle feature corresponding to this tail whereas, this tail is most important to discriminate

between O and Q. Whereas, principle component may ignore the feature corresponding to this tail part and unless I have that information I cannot discriminate between O and Q.

So, that is the drawback of principle component analysis, though it gives you the best representation of data onto the projected space in minimum square sense, but it may not be the best projection. So, far as separability of the class are concerned so that is the aim of the multiple discriminant analysis. And when we discussed today, I will take a specific case of multiple discriminant analysis that I will consider that. Suppose, I have got only two classes

and the approach taken for this multiple discriminant analysis to separate between two classes, and or to design a classifier which can classify between two classes after taking the projections onto lower dimensional space, that is what is called Fisher discriminator.

(Refer Slide Time: 09:47)



So, let us see what is this. Yes, they will pass through the mean of the samples, but what I have shown is just the projection direction. Actually, the line will pass through the mean of the samples, I have just shown the projection direction, not its exact position and then try to say see that. Because even if I pass this line to the mean of the samples, even then the same problem will occur. Either the projection will be downwards or the projection will be upwards, but the same problem will exist. I just have done it for the clarity of the figure that is all.

So, let us assume that we have n number of d dimensional feature vectors which are say X_1, X_2 up to X_n . And these feature vectors are partitioned into two different sets, one is set D_1 consisting of n_1 number of feature vectors which are labelled as class ω_1 , because we are going for supervised learning though. So, I have out of this n number of feature vectors, n_1 number of feature vectors where obviously n_1 is less than n number of feature vectors. I say that these feature vectors are in the set D_1 and these vectors are labelled to belong to class ω_1 . Similarly, I have another set D_2 consisting of n_2 numbers of feature vectors and these feature vectors are labelled as class ω_2 .

So, obviously $n_1 + n_2$ in our case will be equal to n , n is the total number of feature vectors partitioned into two subsets, one consisting of n_1 number of feature vectors coming from class ω_1 . I am calling that set of feature vectors as D_1 and the other one is n_2 number of feature vectors coming from class ω_2 , and I say that this set is set D_2 . And suppose I take a projection direction, W direction of projection where I assume that $\|W\|=1$, that means this is an unit vector in the direction of projection line.

So, given this a feature vector X_i when I take its orthogonal projection, because as we said that the orthogonal projections are the best projections. So, whenever we will talk about projection, we will talk about orthogonal projection. So, when I take orthogonal projection of the feature vectors onto this line. Suppose, I take a feature vector X_i , take the orthogonal projection onto this projection line, I get projected vector y_i . So, here this y_i will be nothing but $y_i = W^T X_i$, because W is the unit vector in the direction of line of projection and X_i is the feature vector in my original space. When I take the projection of this in the direction of W , I get $W^T X_i$ and the projected vector is y_i .

So, when I have this n number of samples, X_1 to X_n , the projections of this onto a projecting line will be y_1, y_2 up to y_n . So, each of this I can compute following this formula, so here i varies from 1 to n . So, these are the projected vectors that I can have after taking projection in the direction W . Now, coming to the sets D_1 and D_2 .

(Refer Slide Time: 14:59)

$$m_1 = \frac{1}{n_1} \sum_{x \in D_1} x$$

$$m_2 = \frac{1}{n_2} \sum_{x \in D_2} x$$

$$\tilde{m}_1 = \frac{1}{n_1} \sum_{y \in R_1} y$$

I can find out the mean m_1 which is nothing but $m_1 = \frac{1}{n_1} \sum_{x \in D_1} x$ mean of the samples in class

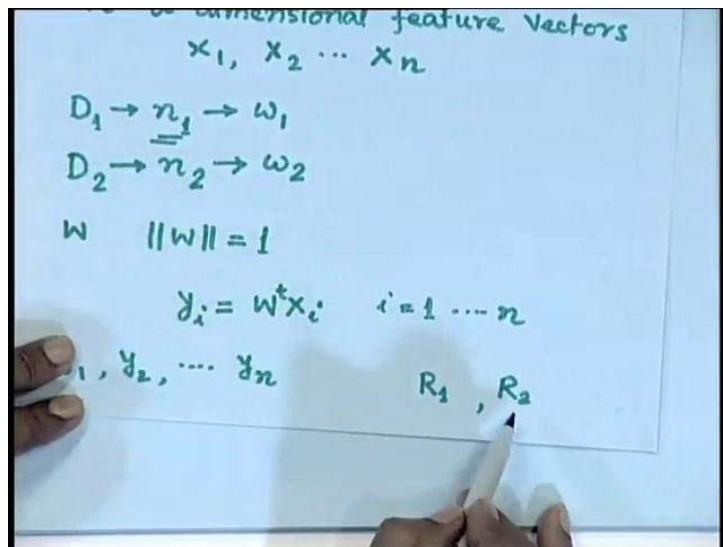
D_1 , and obviously this $m_1 = \frac{1}{n_1} \sum_{x \in D_1} x$, because n_1 is the number of samples in class D_1 . Then

summation of x for all x belonging to set D_1 , I take all the samples from set D_1 and average of that gives me the mean m_1 . Similarly, m_2 is nothing but $m_2 = \frac{1}{n_2} \sum_{x \in D_2} x$. Now, when I compute the projections of this mean vectors.

Suppose, the projected mean vector m_1 , I write as \tilde{m}_2 , this \tilde{m}_1 is nothing but $\tilde{m}_1 = \frac{1}{n_1} \sum_{y \in R_1} y$,

for all y belonging to say set R_1 . Now, what is this R_1 when I have all these projected samples, projected vectors y_1 to y_n . Now, the projected vectors which are projection of these n_1 number of samples.

(Refer Slide Time: 16:49)



I call that set as set R_1 . So, this set is the projected vectors of these n_1 number of samples. Similarly, I say set R_2 which is the set of projected vectors, set of this n_2 number of projected vectors. I call this set R_2 , right? So, here this \tilde{m}_1 which is the projected mean is nothing but

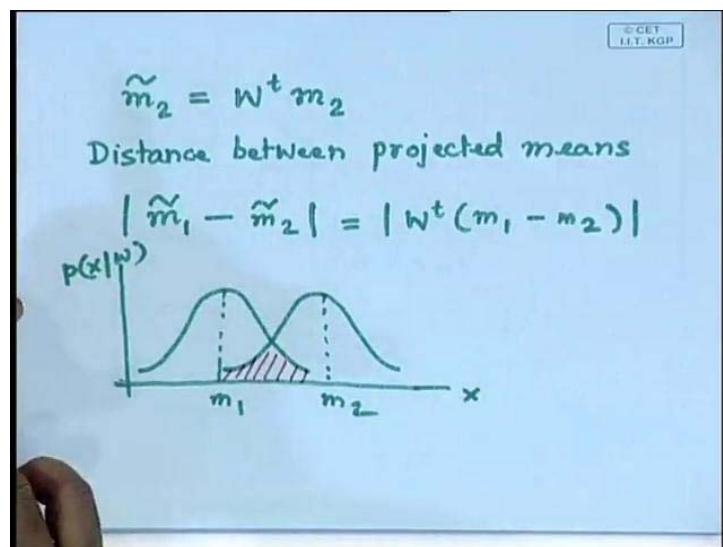
$$\frac{1}{n_1} \sum_{y \in R_1} y.$$

(Refer Slide Time: 17:35)

The image shows a handwritten derivation on a whiteboard. It starts with the formula for the mean of set D_1 :
$$m_1 = \frac{1}{n_1} \sum_{\forall x \in D_1} x$$
Then it shows the formula for the mean of set D_2 :
$$m_2 = \frac{1}{n_2} \sum_{\forall x \in D_2} x$$
Next, it derives the formula for the projected mean \tilde{m}_1 :
$$\begin{aligned}\tilde{m}_1 &= \frac{1}{n_1} \sum_{\forall y \in R_1} y = \frac{1}{n_1} \sum_{\forall x \in D_1} W^t x \\ &= \frac{1}{n_1} W^t \sum_{\forall x \in D_1} x = W^t m_1\end{aligned}$$

And this y is nothing but $W^t x$, for all X belonging to set D_1 . I can take this W^t out of this summation, because it does not depend upon the samples, okay? I simply get $\frac{1}{n_1} W^t \sum_{\forall x \in R_1} x$ which is nothing but $W^t m_1$. So, that is quite obvious. So, projection of mean of the vectors in set D_1 is nothing but $W^t m_1$.

(Refer Slide Time: 18:48)



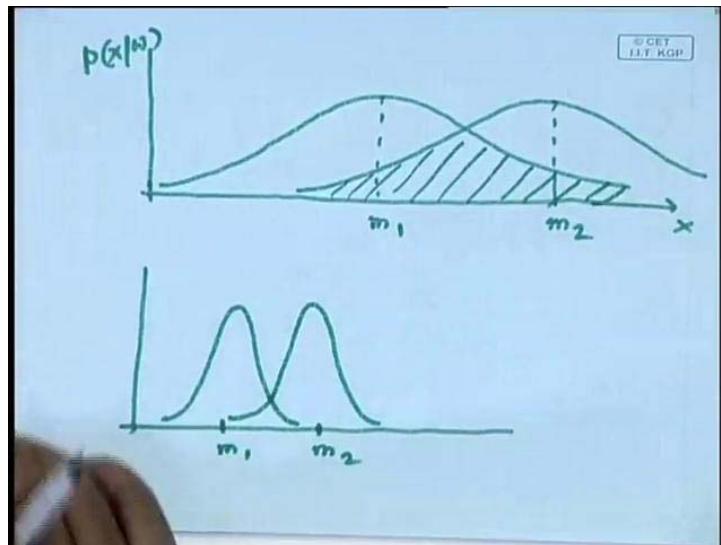
In the same manner I can compute the projection of mean of samples in set D_2 which is nothing but \tilde{m}_2 . If I follow the same procedure it will simply be $W^t m_2$. So, \tilde{m}_2 that is the

projection of m_1 onto the direction of W is nothing but $W^t m_1$, projection of m_2 onto the direction of W is nothing but \tilde{m}_2 which is nothing but $W^t m_2$. Now, what is the distance between these two projected means? That is nothing but so the distance between projected means is nothing but $|\tilde{m}_1 - \tilde{m}_2|$. This is the distance between these two projected means or difference between the two projected means, which will be simply $|W^t(m_1 - m_2)|$.

So, this is quite simple. Now, you find that I can increase the distance between \tilde{m}_1 and \tilde{m}_2 by simply scaling up W . Here, I have assumed that modulus of W is equal to 1, if modulus of W is more than 1 then the distance between \tilde{m}_1 and \tilde{m}_2 will go on increasing. As I increase the scale factor of W , the distance between \tilde{m}_1 and \tilde{m}_2 will increase, and that ensures that I can increase the separability between the two classes. But the question is how much should I increase the separation between the two classes?

Should I go on increasing indefinitely? Obviously that is not justified or not very logical, because how much difference between \tilde{m}_1 and \tilde{m}_2 that I need for good classification depends upon, what is the variance of the samples within set D_1 and what is the variance of samples within set D_2 ? This variance will indicate that what should be the between \tilde{m}_1 and \tilde{m}_2 , the reason is very simple, if I take that distribution suppose I have two distributions like this. This is m_1 , this is m_2 , this is my x and this is say $p(x/\omega)$. We said the error of classification is nothing but the area bounded between these two density curves, probability density curves which is this shaded portion. Now, if the variance of the data distribution is quite large that means I have variance something like this.

(Refer Slide Time: 22:46)



The other one is like this. So, this is my x , this is $p(x/\omega)$, this is m_1 and this is m_2 , here the variance or standard deviation of the data within the two different classes are quite large. So, the curve has become very flat and the error is this. So, when I have this sort of distribution, you find that I should have the difference between m_1 and m_2 to be quite large. So, that the error of classification is reduced, that means these two curves should be wide apart, right? Whereas if I have distribution of this form where the variance is very small, m_1 is somewhere over here, m_2 is somewhere over here.

So, in this situation I do not need that much separation which I need in the first case, here the separation between m_1 and m_2 can be much less than this, but even then my error the total error will be under control. So, how much should be the difference between two projected means in the reduced space that should be relative to the variance or standard deviations of data within different classes.

So, it should not be an absolute value, it should be relative to the variance of data. So, accordingly I can make use of the scatter or variance to have some criteria function, the criteria function which has to be either maximized or minimized. And when it is maximized or minimized I can say that the separation between the two means that I have got that is sufficient for classification. So, we can define the scatter of the projected data like this.

(Refer Slide Time: 25:32)

$$\text{Scatter of Projected Data}$$

$$\tilde{s}_i^2 = \sum_{y \in R_i} (y - \tilde{m}_i)^2$$

$$\frac{1}{n} (\tilde{s}_1^2 + \tilde{s}_2^2)$$

So, let us see what is scatter of projected data. So, for an i^{th} class I define the scatter of the projected data as $\tilde{s}_i^2 = \sum_{y \in R_i} (y - \tilde{m}_i)^2$, where this summation has to be taken over all y belonging

to set R_1 , is that you find that this is something similar to variance. What we have not done is we have not normalized this. So, when I define such a kind of scatter, for the samples belonging to set D_i or the samples taken from class ω_i , then I can define the total within class scatter. So, the total within class scatter will be nothing but \tilde{s}_i^2 or since we are considering only two classes, let me make it $\tilde{s}_1^2 + \tilde{s}_2^2$. Let us normalize this by the number of samples I have which is equal to n . So, this tells you the total within class scatter of the projected samples, right? And as I said that I want the separation between the two classes or the distance between two means should be relative to this total scatter, right?

(Refer Slide Time: 27:39)

$J(W) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$

$J(\cdot)$

s_i^2 s_w^2

\uparrow

$s_i^2 = \sum_{x \in D_i} (x - m_i)^T (x - m_i)$

$s_w^2 = \sum_i s_i^2 = s_1^2 + s_2^2$

So, I can define now a criteria function which is something like this, $J(W) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$ So, this

$\tilde{s}_1^2 + \tilde{s}_2^2$, it is the total within class scatter which takes into consideration the scatter of the samples taken from set R_1 plus the scatter of samples taken from set R_2 . And what I have at the numerator is $|\tilde{m}_1 - \tilde{m}_2|^2$ which is nothing but the square of the distance between the projected means m_1 and m_2 .

So, what I should try to achieve is this $|\tilde{m}_1 - \tilde{m}_2|^2$ should be as large as possible, with respect to the total within class scatter. Or effectively I want to maximize this criteria function $J(W)$,

which is the ratio $\frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$.

So, I want to maximize this. So, I can see effectively that this being the difference between the two classes or two means. I can say equivalently that this is something similar to inter class scatter and this is something similar to intra class scatter. So, if intra class scatter is less that indicates that your data distribution is very compact. If it is compact then inter class scatter I need not have very large value to give the separation between two classes, right? Whereas if this intra class scatter is large that means the data are distributed sparsely or widely within every class. If it is so then this inter class scatter should be quite large so that I can have satisfactory classification performance or I can discriminate between the classes very easily.

So, effectively what I have to have is I have to maximize this criteria function $J(W)$, which is nothing but something like ratio of inter class scatter and intra class scatter. And Fisher's discriminator actually maximizes this and the value of W which maximizes this gives you the projection direction. So, how do I find optimal value of W which will maximize this? If I want to do that in that case, this expression I have to write as an explicit function of W and then I take the gradient. Or I take the differentiate it with respect to W equate that to 0 and you get the solution.

So, if I want to explicitly write this criterion function J in terms of W , let us do this. First we define a scatter matrix, scatter matrices, one is s_i and other one is s_w , I will say s_i is the scatter within the i^{th} class. And s_w is the total within class scatter, right? So, as before I can define this s_i that is scatter of the samples within i^{th} class which is nothing but

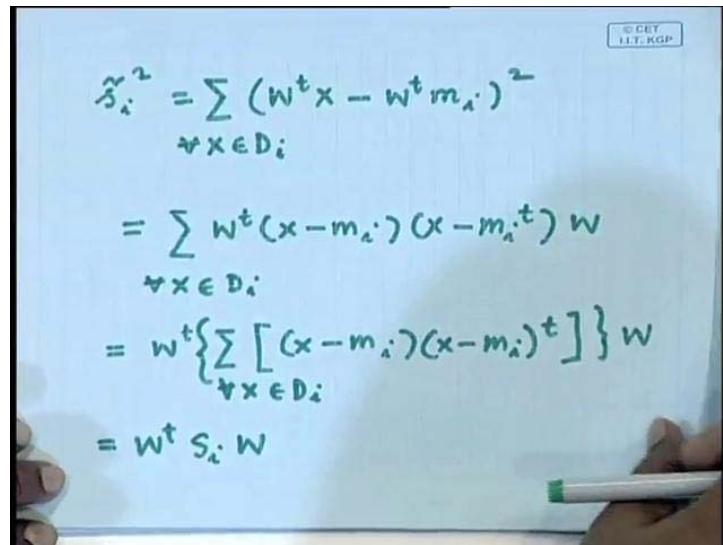
$$s_i = \sum_{x \in D_i} (x - m_i)(x - m_i)^t.$$

So, this gives me the scatter of the samples in set D_i or the scatter of the samples which are taken from class ω_i . So, once I get the scatter for individual classes or individual sets then I can define total within class scatter, because this is within class scatter. So, I can define total within class scatter as $s_w = \sum_{\omega_i} s_i$ and here we are considering only two classes. So, this will

be $s_w = \sum_{\omega_i} s_i = s_1 + s_2$, where s_1 is that total within class scatter s_1 is the scatter of the samples

in set D_1 , s_2 is the scatter of the samples in set D_2 . If I take these two sums that gives me total within class scatter or simply it is called within class scatter. Next, let us try to see that what will be the scatter of the projected samples.

(Refer Slide Time: 33:46)



A handwritten derivation on a whiteboard showing the calculation of the scatter matrix \tilde{s}_i^2 . The derivation starts with the formula $\tilde{s}_i^2 = \sum_{x \in D_i} (w^t x - w^t m_i)^2$, where $w^t x$ represents the projected value of sample x onto the line defined by vector w , and $w^t m_i$ represents the projected mean of the samples in class i .

$$\begin{aligned}\tilde{s}_i^2 &= \sum_{x \in D_i} (w^t x - w^t m_i)^2 \\&= \sum_{x \in D_i} w^t (x - m_i) (x - m_i)^t w \\&= w^t \left\{ \sum_{x \in D_i} [(x - m_i)(x - m_i)^t] \right\} w \\&= w^t S_i w\end{aligned}$$

So, for that I define \tilde{s}_i^2 which is nothing but $\tilde{s}_i^2 = \sum_{x \in D_i} (W^t x - W^t m_i)^2$. So, earlier we have

taken the scatter of the original samples. Now, we are taking the scatter of the projected samples, this expression can be written as just by some re-organization. I can write this expression as $\tilde{s}_i^2 = \sum_{x \in D_i} W^t (x - m_i) (x - m_i)^t W$. Now, given this you will find that this W which

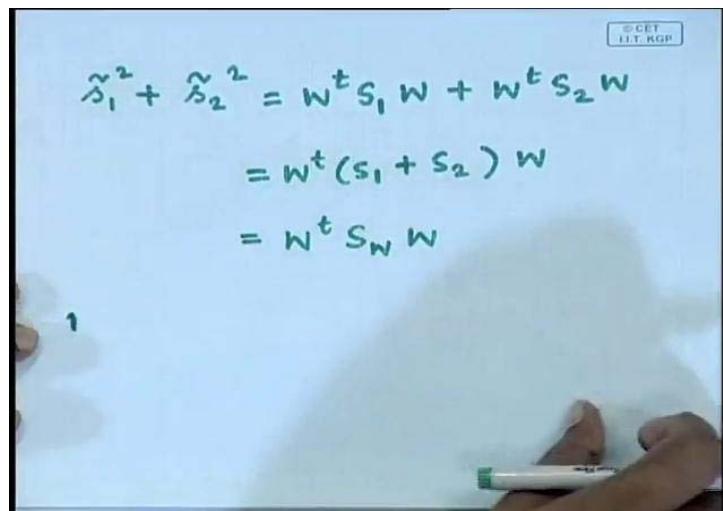
is independent of x can take outside the summation, right?

So, if I take outside the summation then what I will have is

$$\tilde{s}_i^2 = W^t \left\{ \sum_{x \in D_i} [(x - m_i)(x - m_i)^t] \right\} W. \text{ Now, what is this term within this summation? This is}$$

nothing but s_i , that is the scatter of the samples in the original space. So, what I simply have is this is nothing but $\tilde{s}_i^2 = W^t s_i W$.

(Refer Slide Time: 36:09)



A handwritten derivation on a whiteboard showing the calculation of the total scatter matrix $\tilde{s}_1^2 + \tilde{s}_2^2$. The derivation starts with the formula $\tilde{s}_1^2 + \tilde{s}_2^2 = W^t S_1 W + W^t S_2 W$, where S_1 and S_2 are the scatter matrices for classes 1 and 2 respectively.

$$\begin{aligned}\tilde{s}_1^2 + \tilde{s}_2^2 &= W^t S_1 W + W^t S_2 W \\&= W^t (S_1 + S_2) W \\&= W^t S_W W\end{aligned}$$

So, when I have this, then the sum of this scatter over two classes that is $\tilde{s}_1^2 + \tilde{s}_2^2$, that will be simply $\tilde{s}_1^2 + \tilde{s}_2^2 = W^t S_1 W + W^t S_2 W$. This is nothing but $\tilde{s}_1^2 + \tilde{s}_2^2 = W^t (S_1 + S_2) W$ as we have already defined that. This is total within class scatter $S_1 + S_2$ is nothing but S_w , which is total within scatter. So, this simply becomes $\tilde{s}_1^2 + \tilde{s}_2^2 = W^t S_w W$, right? So, in the same manner when I compute the difference between the two means, so that is nothing but the difference between the two means.

(Refer Slide Time: 37:30)

The image shows a handwritten derivation on a whiteboard. At the top, it says "Separation of the projected means". Below that, the equation is written in blue ink:

$$(\tilde{m}_1 - \tilde{m}_2)^2 = (W^t m_1 - W^t m_2)^2$$

$$= W^t (m_1 - m_2) \underbrace{(m_1 - m_2)^t}_{(m_1 - m_2)^t} W$$

$$= W^t S_B W$$

A red arrow points from the term $(m_1 - m_2)^t$ to the text "Between Class Scatter." at the bottom. Below this, in purple ink, it says $S_B W \rightarrow \text{Vector in direction of } (m_1 - m_2)$.

Or separation of the projected means which is $(\tilde{m}_1 - \tilde{m}_2)^2$. Where \tilde{m}_1 is nothing but $\tilde{m}_1 = W^t m_1$. I can re-orient or re-organize this expression as $W^t (m_1 - m_2) (m_1 - m_2)^t W$, and you find that this is something like between class scatter. So, we had this S_1 to be within class scatter. Because for computation of this I am considering the samples belonging to individual classes computing their scatters and then adding these intra class scatters.

So, this is what is called within class scatter, and following the similar expression, this expression $(m_1 - m_2) (m_1 - m_2)^t$. This tells you something like between class scatter. So, I can write this expression as $W^t S_B W$, where this S_B we will call as between class scatter. Now, you will find that this is S_w or within class scatter. This is symmetric and positive semi definite and so is this between class scatter S_B that is the property of this scatter matrices and not only that, when I am writing this expression as $W^t S_B W$, where this

$S_B = (m_1 - m_2)(m_1 - m_2)^t$. It is nothing but a vector, column vector and $(m_1 - m_2)^t$ transpose is a row vector.

So, this is actually the outer product of two vectors. So, as it is outer product of two vectors so rank of this scatter matrix S_B will be at the most one, not more than 1. And not only that, I can have some more observation about S_B or I can have some more observation about $S_B W$. You find that when I consider this part. So this portion $(m_1 - m_2)^t W$, this is the inner product of two vectors. Because this is a row vector, this is a column vector of same dimension.

So, this term is inner product of two vectors. As it is inner product of two vectors so it is a scalar, and when this is scalar this total term $(m_1 - m_2)(m_1 - m_2)^t W$. This being a scalar, this whole portion will be a vector and the vector in the direction of $(m_1 - m_2)$. So, when I am writing this as $S_B W$ you find that this S_B will be a vector in the direction of m_1 , sorry $S_B W$ will be a vector in the direction of $(m_1 - m_2)$, because $S_B W$ is nothing but this, right?

So, this $S_B W$ is a vector in direction of $(m_1 - m_2)$. So, we will make use of this while solving for our projection direction W . So, I have these different scatter matrices and the corresponding expressions. So, once I have this, I have this within class scatter, I have this between class scatter and what we said is that is should define a criteria function, which is a ratio of measure of between class scatter and measure of within class scatter, okay?

(Refer Slide Time: 43:20)

The image shows a whiteboard with handwritten notes in blue ink. At the top, there is a formula: $J(W) = \frac{W^t S_B W}{W^t S_W W} \Rightarrow \text{generalized Rayleigh Coefficient.}$ Below this, another equation is written in a box: $S_B W = \lambda S_W W$, followed by the text "→ Generalized Eigenvalue problem." Further down, it says "If S_W is nonsingular." and then another box contains the equation $S_W^{-1} S_B W = \lambda W$, with the text "→ Eigenvalue problem" next to it. In the top right corner of the whiteboard, there is a small logo that reads "© GET IIT-KGP". A person's hand is visible at the bottom right corner of the whiteboard.

So, now if I redefine our criteria function as $J(W) = \frac{W^t S_B W}{W^t S_W W}$, in earlier case it was

$J(W) = \frac{\left| \tilde{m}_1 - \tilde{m}_2 \right|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$, right? And we have found that this $\left| \tilde{m}_1 - \tilde{m}_2 \right|^2$ is nothing but $W^t S_B W$ and

$\tilde{s}_1^2 + \tilde{s}_2^2$ is nothing but $W^t S_W W$. So, I get an expression of this criteria function J explicitly in

terms of W is given by $J(W) = \frac{W^t S_B W}{W^t S_W W}$, where S_B is between class scatter and S_W is total

within class scatter.

So, what I have to do is we have to maximize this ratio by varying the vector W . And W gives you the projection direction and that is the tedious mathematics. So, without going for tedious mathematics, let us see what is the solution. So, a W which will maximize this ratio must satisfy $S_B W = \lambda S_W W$ for some constant λ . So, the W which maximizes this ratio must satisfy this condition for some constant λ . This expression, this ratio is known as generalized Rayleigh coefficient and this expression $S_B W = \lambda S_W W$. It is nothing but a generalized Eigen value problem.

So, you will find that if S_W is non-singular then this expression will be simply in the form $S_W^{-1} S_B W = \lambda W$. So, this is our well known Eigen value problem, where this W is nothing but Eigen vector of $S_W^{-1} S_B$ and λ is the corresponding Eigen value, but we can simplify our solution instead of trying to solve the Eigen values and Eigen vectors simply because of the fact that $S_B W$ as we said that this $S_B W$ is a vector in the direction of $m_1 - m_2$.

(Refer Slide Time: 47:52)

Handwritten notes on a whiteboard:

$$S_B W = \lambda S_W W$$

$$W = S_W^{-1} (m_1 - m_2)$$

$$K(m_1 - m_2) = \lambda S_W W$$

$$K_1(m_1 - m_2) = S_W W$$

$$W = (K_1) S_W^{-1} (m_1 - m_2)$$

$$W = S_W^{-1} (m_1 - m_2)$$

So, by using this and from the relation that $S_B W = \lambda S_W W$, I can find out what should be the direction W , because as I said the scale factor of W is not much important. What I want to see is what is the direction of W , that is my projection direction. So, from this you can easily find that W is nothing but $S_W^{-1} (m_1 - m_2)$. Why is it so? Because as we said that $S_B W$ is a vector in the direction of $m_1 - m_2$ so I can write this $S_B W$ as some scale factor $K(m_1 - m_2)$. Because it is a vector in the direction of $(m_1 - m_2)$, right?

So, I can write this as some scale factor $K(m_1 - m_2) = \lambda S_W W$, λ is the constant. So, I can put this constant under this constant K , because it simply becomes K/λ . So, equivalently I have some $W = K_1 S_W^{-1} (m_1 - m_2)$, right? Which clearly says that this W is nothing but $W = K_1 S_W^{-1} (m_1 - m_2)$, is it. And as I said that this scale factor is not that important to me, because I am interested in the direction of W . The scale factors will indicate that how much will be the separation between the two classes and in the projected domain how the points will be distributed.

If I increase the scale factor, separation will be more. The points will be more spreaded if I reduce the scale factor. The separation will be less points, will be less spreaded. So, both of them the separation between the classes as well as the variance in the projected domain, both of them vary simultaneously depending upon the value of K , in the same scale. So, this scale K or K_1 is not that important. What is important is what is the direction W . So, as such I can ignore this scale K_1 and if I ignore this scale K_1 , I simply have $W = S_W^{-1} (m_1 - m_2)$. Where this S_W is total within class scatter and $(m_1 - m_2)$ is a vector in the direction of line joining m_1 and m_2 , right?

So, if I use this projection direction for projecting the data into lower dimensional space, I get, I can maintain the separation among the classes and features data for classification purpose. However, this particular projection may not be the best in terms of data representation as far as minimum squared error is concerned. So, what I have done is, till now is a two class problem, and your classification design gets complete because what I have done is from multiple dimension, I have projected onto single dimension and in the single dimension I have to find out a threshold so that if this one dimensional vector. So, the scalars are less than the threshold, I will put them into one class. If it is more than the threshold, I have to put them into another class.

So, I have to determine the threshold value that is all, and the classifier which makes use of this concept is as I said is called Fisher's classifier or Fisher's linear discriminator. Now, when I have seen number of classes, I have to extend from two class problem to C class problem. And it can be shown that in that case it simply becomes $C-1$ linear classifiers or the projection that now I am taking into one dimension. I have to take projections onto $C-1$ number of different directions.

So, the problem remains the same, simply it is extension from two dimension to C dimension or the dimensionality of the feature vectors. Here it is one dimensional feature vector and there we have to go for $C-1$ dimensional feature vector and obviously I will gain if $C-1 < D$. Where D is the original dimensionality of the feature vectors, and $C-1$ is the reduced dimensionality of the feature vectors.

So, we have discussed about this dimensionality problem and two ways to solve, to tackle the dimensionality problem. One approach tries to find out a projection which is the best in terms of data representation that is the principle component analysis approach. And the other approach that is this MDA or multiple discriminant analysis is the one which is again a projection onto a lower dimensional space. But it tries to guarantee that when you take the projection, the separability of the feature vectors between different classes that will be maintained which is not guaranteed by principle component analysis. So, we will stop here today. Next day we will start some other topic.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 18
Multiple Discriminant Analysis
(Tutorial)

Hello. So, for last two three classes, we have discussed about the problem of dimensionality of the feature vectors. We have seen that in case of pattern recognition, depending upon the complexity of the patterns that we want to classify or we want to recognize, we have to capture different types of features from the patterns. And the number of features in some cases can be very large it can be two features, it can be three features it can be ten features, it can be hundred features or it can be even thousands of features.

So, we have seen through our analytical analysis, that as the dimension of the feature vector increases, the training complexity or the time required to train the classifiers also increases to a large extent. So, it is always better that if we can reduce the feature dimensions. So, we have said we have discussed earlier that one way to reduce the dimension of the feature vectors is to project the feature vectors into lower dimensional space.

And when I project a higher dimensional feature vector onto a lower dimensional space, then there is always a risk that if I have 2 different classes, say ω_1 and ω_2 . And say in higher dimensional space in n dimensional space, if the feature vectors belonging to ω_1 , and the feature vectors belonging to ω_2 , they are separable when I reduce the number of dimensions, it is quite possible that in the reduced dimension the feature vectors will not remain separable anymore.

And the other thing is that what should be the best projection direction or what is the best projection space. So, we have seen that. So, far as data representation is concerned in a lower dimensional space, the best way of projection is to project the higher dimensional feature vectors onto the Eigen vectors as computed from the feature vectors. And we have also seen that if I simply project the higher dimensional feature vectors onto Eigen vectors or onto the Eigen space, then the separability among the feature vectors may be lost.

The Feature vectors belonging to different classes may not remain separable anymore. So, for that we have gone for Fisher's linear discriminator. So, in case of Fisher's linear discriminator, we have seen that we try to maximize the between classes scatter and at the same time we try to minimize the within class scatter. That means when you project the higher dimensional feature vectors onto a lower dimensional space, we try to find out the projection space in such a way that the clusters or the patterns belonging to different classes they will be as apart as possible. And simultaneously, the patterns belonging to the same class they will be as compact as possible.

That is what we mean by, it tries to maximize the between class scatter and it tries to minimize the within class scatter. And accordingly, we have computed different projection directions. So, in the first case when simply data representation is our aim onto a lower dimensional space the projection directions are actually the Eigen vectors. And this simply Eigen vector projecting onto the Eigen vector may not maintain the separability. So, which is not very suitable for classification purpose because in case of classification it is not only data representation onto the lower dimensional space, but it is also the separability of the data belonging to different classes that is very important.

So, for classification our aim is that we want to find out the projection directions, which maximizes the between class scatter and it tries to minimize the within class scatter. So, today we will take up an example, we will try to solve a problem which will illustrate this fact that if I take projection onto an Eigen vector, the separability may not be maintained, but if I project onto another space following the Fisher's linear discriminator the separability of the data may be maintained.

(Refer Slide Time: 05:30)

Tutorial on
Multiple Discriminant Analysis

$\left\{ \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 6 \end{pmatrix}, \begin{pmatrix} 7 \\ 5 \end{pmatrix} \right\} \in \omega_1$

$\left\{ \begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 10 \\ 1 \end{pmatrix}, \begin{pmatrix} 12 \\ 3 \end{pmatrix}, \begin{pmatrix} 13 \\ 6 \end{pmatrix} \right\} \in \omega_2$

Find out the projection direction that maintains
separability between the two classes.



NPTEL

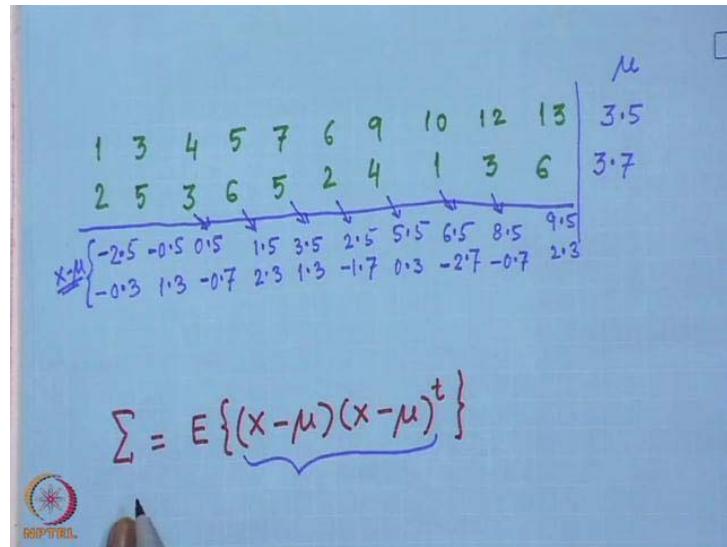
So, we are going to have today a tutorial on multiple discriminant analysis. So, we will take a number of feature vectors from 2 different classes, a set of feature vectors from class ω_1 another set of feature vectors from class ω_2 . So, suppose we have been given some feature vectors, let us take them let feature vectors say, $\begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 6 \end{pmatrix}, \begin{pmatrix} 7 \\ 5 \end{pmatrix}$. So, we are considering two dimensional feature vectors. So, suppose these are the five feature vectors which belong to class ω_1 .

So, these are the feature vectors which are taken from class ω_1 . And we have another set of feature vectors say $\begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 10 \\ 1 \end{pmatrix}, \begin{pmatrix} 12 \\ 3 \end{pmatrix}, \begin{pmatrix} 13 \\ 6 \end{pmatrix}$. And say, these are the feature vectors which are taken from class ω_2 . And we want to find out the projection direction so that when these feature vectors are projected onto a lower dimensional space projected on to a line, we want to find the direction of the line. So, that when this feature vectors are projected onto that line the separability of the feature vectors between the two classes ω_1 and ω_2 that will be maintained.

So, what we want to do is find out the projection direction that maintains separability between the two classes. And of course, what we will do is before finding out such a projection direction which maintains the separability, we will find out the best projection direction. So, far as data representation into a lower dimensional space is concerned and then we will solve this problem. We will find out the projection direction that will maintain the separability. So, that we can compare the performance of the two. So, that we can demonstrate that in one case the separability is maintained and in the other case the separability is not maintained. So, we have these four, five feature vectors from class ω_1 and we have the 5 feature vectors from class ω_2 , right.

So, first if I want to find out the projection direction for best representation, then what we have to do is we have to find out the Eigen vectors of the co-variance matrix of these data elements. So, let us see how we can find out the co-variance matrix first and from the covariance matrix we have to find out the Eigen vector corresponding to the maximum Eigen value. So, first let us find out the co-variance matrix. So, to find out the best projection direction, we have to consider all these feature vectors together forgetting about their class belongings

(Refer Slide Time: 10:30)



So, we have the feature vectors $\begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 6 \end{pmatrix}, \begin{pmatrix} 7 \\ 5 \end{pmatrix}$ from class ω_1 and

$\begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 10 \\ 1 \end{pmatrix}, \begin{pmatrix} 12 \\ 3 \end{pmatrix}, \begin{pmatrix} 13 \\ 6 \end{pmatrix}$. These are the feature vectors from class ω_2 and considering all

these feature vectors I have to find out the co-variance matrix. So, to find the co-variance matrix first what I have to do is I have to find out the mean vector μ . And this mean vector is nothing but mean of all these feature vectors. So, when I try to find out the mean of all these feature vectors, I have to find out the mean of the first component I also have to find out the mean of the second component. And the mean of the first component and mean of the second component these two taken together gives you the mean vector.

So, when I compute mean of all these ten feature vectors, the first components of all this ten feature vectors this mean you can compute. This will come out to be something like say 3.5. And the mean of all the second component of all the ten feature vectors that will come out to

be something like 3.7. So, this is my mean vector $\mu = \begin{pmatrix} 3.5 \\ 3.7 \end{pmatrix}$. As you know that when I

compute the co-variance matrix, the co-variance matrix is defined as $\Sigma = E\{(X - \mu)(X - \mu)^T\}$,

where X's are individual feature vectors and this μ is the mean vector.

So, I have to compute $(X-\mu)$. So, from every feature vector x , I have to subtract the mean vector which is μ . So, when I subtract the mean vector from every feature vector what I have is, let me put here I will put $(X-\mu)$ for individual feature vectors. So, I will put $x - (X-\mu)$ here. So, here we find that, when I subtract this mean vector μ from the first feature vector which is $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$, this will simply be $\begin{pmatrix} -2.5 \\ -0.3 \end{pmatrix}$. So, this is the first feature vector minus the mean vector. For the second feature vector it will be $\begin{pmatrix} -0.5 \\ 1.3 \end{pmatrix}$. For the third vector it will be $\begin{pmatrix} 0.5 \\ -0.7 \end{pmatrix}$. For the fourth vector it will be $\begin{pmatrix} 1.5 \\ 2.3 \end{pmatrix}$, for this vector it will be $\begin{pmatrix} 3.5 \\ 1.3 \end{pmatrix}$. So, these are $X - \mu$ for all the feature vectors belonging to class ω_1 .

Now, coming to other set of feature vectors this is $\begin{pmatrix} 2.5 \\ -1.7 \end{pmatrix}$. Similarly, for this is it $\begin{pmatrix} 5.5 \\ 0.3 \end{pmatrix}$. Similarly, for this it will be $\begin{pmatrix} 6.5 \\ -2.7 \end{pmatrix}$. For this vector it will be $\begin{pmatrix} 8.5 \\ -0.7 \end{pmatrix}$. And for the last one it will be $\begin{pmatrix} 9.5 \\ 2.3 \end{pmatrix}$. So, these two row's actually give me the feature vectors, $X - \mu$ where $X - \mu$ have been computed by subtracting the mean vector from the individual vectors.

And then for each of these vectors after subtraction of the mean vector I have to compute the $(X-\mu)(X-\mu)^t$. And then I have to take the average of all those matrices, which I get that gives me the expectation value of $(X-\mu)(X-\mu)^t$ which is nothing but my co-variance matrix.

So, if I do that for the first one, you find that the first one is $\begin{pmatrix} -2.5 \\ -0.3 \end{pmatrix}$.

(Refer Slide Time: 17:15)

Handwritten notes showing matrix calculations:

$$\begin{pmatrix} -2.5 \\ -0.3 \end{pmatrix} \begin{pmatrix} -2.5 & -0.3 \end{pmatrix} = \begin{pmatrix} 6.25 & 0.75 \\ 0.75 & 0.09 \end{pmatrix} \rightarrow M_1$$

$$\begin{pmatrix} -0.5 \\ 1.3 \end{pmatrix} \begin{pmatrix} 0.5 & 1.3 \end{pmatrix} = \begin{pmatrix} 0.25 & -0.65 \\ -0.65 & 1.69 \end{pmatrix} \rightarrow M_2$$

$$\begin{pmatrix} 0.5 \\ -0.7 \end{pmatrix} \begin{pmatrix} 0.5 & -0.7 \end{pmatrix} = \begin{pmatrix} 0.25 & -0.35 \\ -0.35 & 0.49 \end{pmatrix} \rightarrow M_3$$

$$M_4 = \begin{pmatrix} 2.25 & 3.45 \\ 3.45 & 5.29 \end{pmatrix}$$

So, what I have to compute is $\begin{pmatrix} -2.5 \\ -0.3 \end{pmatrix} \begin{pmatrix} -2.5 & -0.3 \end{pmatrix}$, this is my $(X-\mu)(X-\mu)^t$. So, this is a column vector which is of dimension 2×1 and this is a row vector of dimension 1×2 , obviously I can do matrix multiplication. And the product matrix that the resultant matrix will be of dimensional 2×2 . And if I simply compute this you find that the value of this matrix will be $\begin{pmatrix} 6.25 & 0.75 \\ 0.75 & 0.09 \end{pmatrix}$. So, this is one of the matrices similarly, for the second one which is $\begin{pmatrix} -0.5 \\ 1.3 \end{pmatrix} \begin{pmatrix} 0.5 & 1.3 \end{pmatrix}$ when I do this multiplication this matrix would be $\begin{pmatrix} 0.25 & -0.65 \\ -0.65 & 1.69 \end{pmatrix}$, for the third one which is $\begin{pmatrix} 0.5 \\ -0.7 \end{pmatrix} \begin{pmatrix} 0.5 & -0.7 \end{pmatrix}$, this will be simply $\begin{pmatrix} 0.25 & -0.35 \\ -0.35 & 0.49 \end{pmatrix}$.

So, this is my third matrix and this way if I compute all the matrices, once I have said how to compute individual matrices $(X-\mu)(X-\mu)^t$, this can be done for all the vectors $X-\mu$ and I can get similar such matrices. So, as I have 10 different vectors I will have 10 such matrices. So, let us put those 10 matrices one by one. So, this is my matrix one M_1 , this is my matrix M_2 , this is my matrix M_3 . Similarly, M_4 with the fourth vector, that will be simply will be

$$M_4 = \begin{pmatrix} 2.25 & 3.45 \\ 3.45 & 5.29 \end{pmatrix}.$$

(Refer Slide Time: 20:58)

$$M_5 = \begin{pmatrix} 12.25 & 4.55 \\ 4.55 & 1.69 \end{pmatrix} \quad M_9 = \begin{pmatrix} 72.25 & -5.95 \\ -5.95 & 0.49 \end{pmatrix}$$

$$M_6 = \begin{pmatrix} 6.5 & -4.25 \\ -4.25 & 2.89 \end{pmatrix} \quad M_{10} = \begin{pmatrix} 90.25 & 21.85 \\ 21.85 & 5.29 \end{pmatrix}$$

$$M_7 = \begin{pmatrix} 30.25 & 1.65 \\ 1.65 & 0.09 \end{pmatrix}$$

$$M_8 = \begin{pmatrix} 42.25 & -17.55 \\ -17.55 & 7.29 \end{pmatrix}$$

M_5 the fifth matrix, $M_5 = \begin{pmatrix} 12.25 & 4.55 \\ 4.55 & 1.69 \end{pmatrix}$, the sixth matrix $M_6 = \begin{pmatrix} 6.5 & -4.25 \\ -4.25 & 2.89 \end{pmatrix}$, M_7 seventh matrix will be $M_7 = \begin{pmatrix} 30.25 & 1.65 \\ 1.65 & 0.09 \end{pmatrix}$, $M_8 = \begin{pmatrix} 42.25 & -17.55 \\ -17.55 & 7.29 \end{pmatrix}$, $M_9 = \begin{pmatrix} 72.25 & -5.95 \\ -5.95 & 0.49 \end{pmatrix}$ and the last one, $M_{10} = \begin{pmatrix} 90.25 & 21.85 \\ 21.85 & 5.29 \end{pmatrix}$, right? So, these are the 10 matrices that we have. So, the first matrix is this, which $(X_1 - \mu)(X_1 - \mu)^t$.

The second one $(X_2 - \mu)(X_2 - \mu)^t$ that gives this, third one $(X_3 - \mu)(X_3 - \mu)^t$ that gives this, the fourth one $(X_4 - \mu)(X_4 - \mu)^t$ that gives this. Continuing this way up to the tenth, $(X_{10} - \mu)(X_{10} - \mu)^t$ that gives me this. So, I have this total of 10 matrices, now out of this 10 matrices when I combine them I add them and divide by the number of matrices that I have what I get is the expectation value

(Refer Slide Time: 24:10)

The image shows a handwritten derivation of the covariance matrix formula. It starts with the definition of the covariance matrix as the expected value of the outer product of a vector and its transpose, $\Sigma = E\{(x-\mu)(x-\mu)^T\}$. This is then equated to $\frac{1}{10} \sum_{i=1}^{10} (x_i - \mu)(x_i - \mu)^T$. The summation is then simplified to $\frac{1}{10} \sum_{i=1}^{10} M_i$, where each M_i is a matrix. Finally, the result is given as a 2x2 matrix: $= \begin{pmatrix} 2.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix}$, labeled as the Covariance Matrix.

So, as per our definition, the co-variance matrix $\Sigma = E\{(X-\mu)(X-\mu)^t\}$. And over here because we have 10 different vectors. So, this expression will simply be $\Sigma = \frac{1}{10} \sum_{i=1}^{10} (X_i - \mu)(X_i - \mu)^t$. Where the summation has to be taken over $i = 1$ to 10 as I have 10 different feature vectors and accordingly I have 10 different matrices. And each of this X_i gives me a matrix M_i the one that we have computed here, M_1, M_2, M_3 and so on. So, each of $(X_i - \mu)(X_i - \mu)^t$ gives me a matrix M_i . So, this expression will simply $\Sigma = \frac{1}{10} \sum_{i=1}^{10} M_i$.

So, effectively what I have to do is, I have to add all this 10 matrices, M_1 to M_{10} all these different matrices have to be added and then the matrix has to be, the summation matrix has to be divided by 10. And when I do that, you will find that this simply becomes $\begin{pmatrix} 26.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix}$. So, this is my co-variance matrix. So, this is what is the co-variance matrix,

now the next step that I have to do is, I have to find out the Eigen vector corresponding to the maximum Eigen value of this co-variance matrices.

So, first let us try to find out what will be the Eigen value or the maximum Eigen value and then let us try to find out what will be the Eigen vector, corresponding to that maximum Eigen value. So, you all know, that if λ is an Eigen value of this matrix then I can simply find out the Eigen value λ by using this equation.

(Refer Slide Time: 27:58)

The image shows a handwritten derivation of eigenvalues from a 2x2 matrix equation. The matrix is given as:

$$\begin{vmatrix} 26.275 - \lambda & 0.35 \\ 0.35 & 2.53 - \lambda \end{vmatrix} = 0$$

This is simplified to:

$$\Rightarrow (26.275 - \lambda)(2.53 - \lambda) - (0.35)^2 = 0$$

Further simplified to a quadratic equation:

$$\Rightarrow \lambda^2 - 28.805\lambda + 66.3532 = 0$$

The solutions are boxed as:

$$\lambda_1 = 26.28$$

$$\lambda_2 = 2.525$$

The NPTEL logo is visible at the bottom left.

So, the equation that I have to $\begin{vmatrix} 26.275 - \lambda & 0.35 \\ 0.35 & 2.53 - \lambda \end{vmatrix} = 0$. And you solve this equation, I get the different Eigen values of λ . So, simply this equation leads to $(26.275 - \lambda)(2.53 - \lambda) - 0.35^2 = 0$, which is reduced to $\lambda^2 - 28.805\lambda + 66.3532 = 0$. So, here we find that, I get a quadratic equation in terms of λ , where λ is an Eigen value of the co-variance matrix.

And as this equation is a quadratic equation. So, obviously the λ will have two solutions, which indicates that this matrix, this co-variance matrix will have two Eigen values and corresponding to each of this Eigen values I have one Eigen vector. So, this co-variance matrix will have two Eigen vectors one corresponding to each Eigen value. So, when I solve this equation this is a quadratic equation the solution is very simple. You can find out that one of the solutions, $\lambda_1 = 26.28$ and the other solution $\lambda_2 = 2.525$.

So, these are the two Eigen values of my co-variance matrix. And out of these two I have to choose that particular Eigen value which is maximum. So, you find that this Eigen value $\lambda_1 = 26.28$, that is the maximum one. So, once I have this maximum Eigen value, I have to find out what is the Eigen vector corresponding to the maximum Eigen value. And you also know what is this Eigen equation, Eigen value equation which simply says that, if x is an Eigen vector of a matrix A , A is my matrix and suppose X is the Eigen vector then $A X = \lambda X$.

(Refer Slide Time: 31:16)

The image shows handwritten mathematical work on a blue background. At the top, it says $\underline{AX} = \underline{\lambda X}$. Below this, a covariance matrix is given as $\begin{pmatrix} 26.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix}$. To its right is an eigenvalue equation: $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 26.28 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. This leads to a system of linear equations: $\begin{cases} 26.275 x_1 + 0.35 x_2 = 26.28 x_1 \\ 0.35 x_1 + 2.53 x_2 = 26.28 x_2 \end{cases}$. Below these equations, the solution is given as $x_1 = 70$ and $x_2 = 1$, with an arrow pointing to the vector $\begin{pmatrix} 70 \\ 1 \end{pmatrix}$. To the right of the vector, the value 26.28 is underlined.

Where this λ is the Eigen value and X is the Eigen vector of matrix A . So, following the same one, here we can find out what is my Eigen vectors corresponding to this maximum Eigen value λ , which is equal to 26.28. So, let us try to find out the Eigen vector corresponding to this maximum Eigen value. So, in order to do that we know that our co-variance matrix was

given by $\begin{pmatrix} 26.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix}$. This was my co-variance matrix sigma and if I assume that the

Eigen vector corresponding to the maximum Eigen value of this co-variance matrix is X . And this being a two dimensional matrix which has got two components x_1 and x_2 , right?

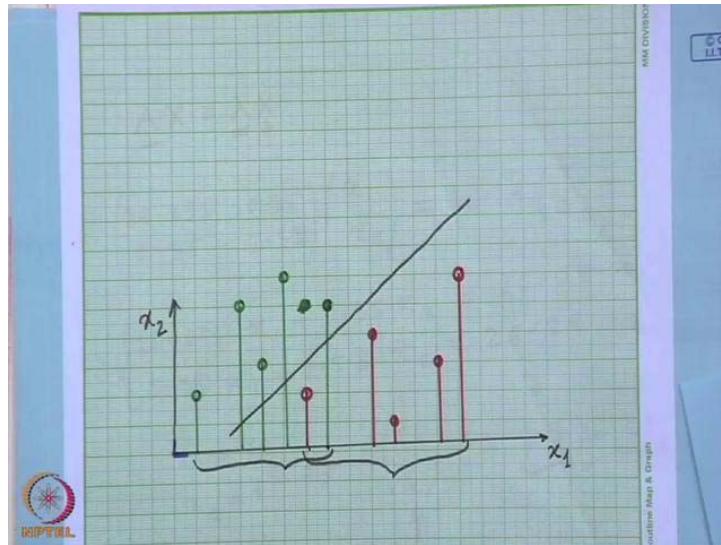
So, this has to be $\begin{pmatrix} 26.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ and where this λ , the maximum Eigen value

is 26.28. So, this has to be simply $\begin{pmatrix} 26.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 26.28 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. So, if I reduce this

equation into linear equations I get two equations, one is $26.275x_1 + 0.35x_2 = 26.28x_1$ and the other one is $0.35x_1 + 2.53x_2 = 26.28x_2$. So, I get these two simultaneous equations. So, I have to find out the value of x_1 and the value of x_2 by solving these two linear equations, simultaneous equations. And the solution also is very simple, and if you solve this you will find that I will get x_1 is equal to something around 70 and x_2 will be something around 1.

So, that clearly indicates that $\begin{pmatrix} 70 \\ 1 \end{pmatrix}$ is the Eigen vector of the co-variance matrix corresponding to the Eigen value $\lambda = 26.28$. So, corresponding to this Eigen value the Eigen vector is $\begin{pmatrix} 70 \\ 1 \end{pmatrix}$. What we have to do is we have to project the feature vectors which are given onto this vector $70\ 1$. So now, let us plot these points on a graph paper to see that what kind of projection we get right. So, I will use a graph paper to this job. So, what we have is, let us assume that we have the origin somewhere over here.

(Refer Slide Time: 35:24)



This is our origin, and I will put these different feature vectors which are given onto this graph paper. And over there if you remember that we said these are the feature vectors $1\begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \begin{pmatrix} 4 \\ 5 \end{pmatrix}, \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \begin{pmatrix} 7 \\ 6 \end{pmatrix}$, these first 5 feature vectors they belong to class ω_1 and the second 5 feature vectors $\begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 10 \\ 1 \end{pmatrix}, \begin{pmatrix} 12 \\ 3 \end{pmatrix}, \begin{pmatrix} 13 \\ 6 \end{pmatrix}$ they belong to class ω_2 . So, when I plot these feature vectors onto my graph paper I will plot them into two different colors. So, the first one I will use the green color. So, the first 1 is $\begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ that comes somewhere over here

then $\begin{pmatrix} 3 \\ 5 \end{pmatrix}$, $\begin{pmatrix} 3 \\ 5 \end{pmatrix}$ is somewhere over here then $\begin{pmatrix} 4 \\ 3 \end{pmatrix}$, $\begin{pmatrix} 4 \\ 3 \end{pmatrix}$ is this, then comes $\begin{pmatrix} 5 \\ 6 \end{pmatrix}$ over here and then $\begin{pmatrix} 7 \\ 5 \end{pmatrix}$. 1, 2, 3, 4, 5, okay?

So, these are the feature vectors which belong to class ω_1 . Now, let us plot the feature vectors that belong to class ω_2 and I put them into red color. So, the first one is $\begin{pmatrix} 6 \\ 2 \end{pmatrix}$ from here the feature vectors are $\begin{pmatrix} 6 \\ 2 \end{pmatrix}$, $\begin{pmatrix} 9 \\ 4 \end{pmatrix}$, $\begin{pmatrix} 10 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 12 \\ 3 \end{pmatrix}$ and $\begin{pmatrix} 13 \\ 6 \end{pmatrix}$.

So, these are the feature vectors belonging to two different classes. And we have computed that our Eigen vector is $\begin{pmatrix} 70 \\ 1 \end{pmatrix}$, what is that one. Anyway, so the Eigen vector was $\begin{pmatrix} 70 \\ 1 \end{pmatrix}$. So, I want to plot that Eigen vector which is $\begin{pmatrix} 70 \\ 1 \end{pmatrix}$. So, it will pass through a, somewhere over here. So, I will take a scale to get this direction of this Eigen vector which is this. So, this is my direction of the Eigen vector corresponding to the maximum Eigen value

And now, if I take perpendicular projection onto this Eigen vector, the perpendicular projection of the feature vectors onto this Eigen vector, what I will have is, the different projections will be like this. This feature vector will be projected here, this feature vector will be projected here, this feature vector will be projected here. So, we find that as we said that all the feature vectors belonging to class ω_1 , they are represented by green color and all the feature vectors belonging to class ω_2 , they are represented by red color. And when I project these feature vectors onto the Eigen vector corresponding to the maximum Eigen value.

So, these are my projections. So, here you find that feature vectors belonging to class ω_1 they are projected. So, what we have used is, this is my axis x_1 and this was my axis x_2 . So, the feature vectors belonging to class ω_1 , they are projected onto a points in the line which is in this range, the feature vectors belonging to class ω_2 they are projected over here. So, it clearly says that when I take the projection onto the Eigen vector the projected points, the projected vectors are no more separable.

The projections from class ω_1 and the projections from class ω_2 they get intermixed which is quite obvious in this particular case. So, the feature vector from class ω_1 is over here, where as the feature vector from class ω_2 is over here. Whereas, the rest of the feature vectors

which are projected onto the other side, the rest of the feature vectors from class ω_1 they are projected onto the other side. So, I have a mixing of the feature vectors, the projected feature vectors onto a lower dimensional space. Whereas, you can clearly say see that I can obviously draw a straight line over here separating the feature vectors in class ω_2 and the feature vectors belonging to class ω_1 by a linear straight line.

So, in our original space, in the two dimensional space they were linearly separable, but when I take projections onto the Eigen vector they are no more linearly separable, right? So, this a problem that we face when we project feature vectors from a higher dimensional space onto feature vectors onto a lower dimensional space. And this is the problem which is addressed in the Fisher's linear discriminator, where we try to find out the projection direction which maintains the separability, which tries to maintain the separability by increasing the between class scatter and by decreasing, by minimizing the within class scatter.

So, if you remember the Fisher's linear discriminator, so now let us talk about the Fisher's linear discriminator. So, Fisher's linear discriminator tries to find out the mean of the individual classes. So, it considers the feature vectors belonging to class ω_1 and the feature vectors belonging to class ω_2 , finds out what is the mean of feature vectors in class ω_1 and what is the mean of the feature vectors in class ω_2 . So, one is μ_1 other one is μ_2 . And then it computes what is the within class scatter or what is scatter of the samples of the feature vectors in class ω_1 and what is the scatter of the samples in class ω_2 . Then it defines what is called total within class scatter.

(Refer Slide Time: 45:35)

$$S_1 \quad S_2$$

$$S_W = S_1 + S_2$$

$$W = \underbrace{S_W^{-1}}_{=} (\underbrace{\mu_1}_{=} - \underbrace{\mu_2}_{=})$$

So, if you remember we have said that if S_1 is the scatter of the feature vectors in class ω_1 and S_2 is the scatter of the feature vectors in class ω_2 . Then the total within class scatter is given by S_w which is nothing but $S_1 + S_2$. So, this is the total within class scatter. And if μ_1 is the mean of the feature vectors in class ω_1 and μ_2 is the mean of the feature vectors in class ω_2 , then when you discussed about the Fisher's linear discriminator, we have said the direction, the projection direction is given by a vector W which is nothing but $W = S_w^{-1}(\mu_1 - \mu_2)$. Where S_w is the total within class scatter, μ_1 is the mean of the feature vectors in class ω_1 and μ_2 is the mean of the feature vectors belonging to class ω_2 .

And this expression was obtained by solving a generalized Eigen value equation and that Eigen value equation took care of maximization of the between class scatter. That means pushing the clusters or the classes as apart as possible. And by minimizing the within class scatter that is S_1 or S_2 by making those classes as compact as possible. So, we try to make the feature vectors belonging to a single class very compact. And simultaneously, try to maintain the distance between two different classes. So, accordingly we had obtained an criteria function and by optimization of that criteria function, we could come to a solution something like this, which says that the projection direction which maintains the separability is given by $W = S_w^{-1}(\mu_1 - \mu_2)$. Where S_w is the total within class scatter.

So, if I have 2 different classes one is S_1 and other one is S_2 , S_1 is the scatter of class one, S_2 is the scatter of class two. So, total within class scatter $S_w = S_1 + S_2$. So, let us try to see that by using this Fisher's linear discriminator approach, what projection direction we get and how does it differ from the Eigen direction? So, here again I will consider same set of feature vectors belonging to class ω_1 and ω_2 . So, that I can compare the performance of projection onto a Eigen vector direction, onto a Eigen vector and the projections onto the Fisher's discriminator. So, as we said that we have two sets of feature vectors, two dimensional feature vectors one set taken from class ω_1 the other set taken from class ω_2 .

(Refer Slide Time: 49:21)

Handwritten notes on a whiteboard:

- Feature vectors for class ω_1 : $\omega_1 \rightarrow \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 6 \end{pmatrix}, \begin{pmatrix} 7 \\ 5 \end{pmatrix}$
- Mean of ω_1 : $\mu_1 = \begin{pmatrix} 4 \\ 4.2 \end{pmatrix}$
- Covariance matrix $S_1 = \sum (x_i - \mu_1)(x_i - \mu_1)^T$
- Feature vectors for class ω_2 : $\omega_2 \rightarrow \begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 10 \\ 1 \end{pmatrix}, \begin{pmatrix} 12 \\ 3 \end{pmatrix}, \begin{pmatrix} 13 \\ 6 \end{pmatrix}$
- Mean of ω_2 : $\mu_2 = \begin{pmatrix} 10 \\ 3.2 \end{pmatrix}$
- Covariance matrix S_2

So, from class ω_1 the feature vectors we have taken are $\begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 6 \end{pmatrix}, \begin{pmatrix} 7 \\ 5 \end{pmatrix}$. So, these are

the feature vectors which are taken from class ω_1 . So, each of them is a two dimensional vector $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ is a vector, $\begin{pmatrix} 3 \\ 5 \end{pmatrix}$ is a vector, $\begin{pmatrix} 4 \\ 3 \end{pmatrix}$ is a vector, $\begin{pmatrix} 5 \\ 6 \end{pmatrix}$ is a vector and $\begin{pmatrix} 7 \\ 5 \end{pmatrix}$ is a vector.

Similarly, I had other set of feature vectors which are $\begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 10 \\ 1 \end{pmatrix}, \begin{pmatrix} 12 \\ 3 \end{pmatrix}, \begin{pmatrix} 13 \\ 6 \end{pmatrix}$, right? And

these are the feature vectors which are taken from class ω_2 . So, first what we have to do is, we have to compute the mean of the feature vectors from class ω_1 and the mean of the feature vectors from class ω_2 . So, here I will compute μ_1 and here I will compute μ_2 .

So, if you compute the mean of these feature vectors which is μ_1 , you can easily find out that this is, $20/5$ as I have 5 different feature vectors. So, the first component would be 4 of μ_1 . And similarly, sum of this second components divided by 5 that will simply be 4.2. So μ_1 the

mean of the feature vectors belonging to class ω_1 is $\begin{pmatrix} 4 \\ 4.2 \end{pmatrix}$. Similarly, μ_2 , the mean of the

feature vectors belonging to class ω_2 , that also I can compute from here.

So, mean of the feature vectors belonging to class ω_2 , μ_2 will simply be this is $50/5$ that will be 10. So, that is the first component and in the same manner the second component you can

compute 3.2. So, this is the mean vector of the feature vectors belonging to class ω_1 , and this is the mean vector of the feature vectors belonging to class ω_2 . So, once I have these two means, then I have to compute the scatter of the feature vectors belonging to class ω_1 and scatter of the feature vectors belonging to class ω_2 . That means I have to compute what is S_1 and what is S_2 ? And S_1 is nothing but $S_1 = \sum_{\forall X_i \in \omega_1} (X_i - \mu_1)(X_i - \mu_1)^t$,

(Refer Slide Time: 53:21)

The image shows handwritten mathematical notes on a blue background. On the left, there are two sets of feature vectors:

- Set 1 (Class ω_1): $\begin{array}{ccccc} 1 & 3 & 4 & 5 & 7 \\ 2 & 5 & 3 & 6 & 5 \end{array}$
- Set 2 (Class ω_2): $\begin{array}{ccccc} 6 & 9 & 10 & 12 & 13 \\ 2 & 4 & 1 & 3 & 6 \end{array}$

Next to each set, the mean vector μ is calculated:

- $\mu_1 = \begin{pmatrix} 4 \\ 4.2 \end{pmatrix}$
- $\mu_2 = \begin{pmatrix} 10 \\ 3.2 \end{pmatrix}$

To the right of the vectors, the formula for the scatter matrix S_1 is given as:

$$S_1 = \sum_{\forall X_i \in \omega_1} (X_i - \mu_1)(X_i - \mu_1)^t$$

Below it, the formula for the scatter matrix S_2 is given as:

$$S_2 = \sum_{\forall X_i \in \omega_2} (X_i - \mu_2)(X_i - \mu_2)^t$$

for all $X_i \in \omega_1$. So, you compute $(X_i - \mu_1)(X_i - \mu_1)^t$ for all the feature vectors belonging to class ω_1 . So, each of them gives me a matrix. So, I will get a number of matrices and because here I have 5 different feature vectors. So, I will have 5 different matrices, sum of all those matrices gives me scatter S_1 . Similarly, $S_2 = \sum_{\forall X_i \in \omega_2} (X_i - \mu_2)(X_i - \mu_2)^t$. For all

$X_i \in \omega_2$. So, we have to compute these two. So, let me stop this lecture here, in the next lecture I will complete this task, this problem. And then we will compare the performance of this and the projection onto the Eigen vector.

Thank you.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 19
Multiple Discriminant Analysis (Tutorial)

Hello, so in the last class we have started a tutorial. We have taken up a problem of a few feature vectors belonging to class ω_1 and few feature vectors belonging to class ω_2 . Actually, the problem that we were trying to solve is the dimensionality reduction problem that from a higher dimensional space if I project onto a lower dimensional space. Then in many cases the problem that we face that in higher dimensional space if the feature vectors belonging to different classes are linearly separable, when I project that into a lower dimensional space they may not remain linearly separable anymore and we have also discussed that the best way to project is onto the Eigen space. So, we have taken this particular problem in the previous class.

(Refer Slide Time: 01:28)

Tutorial on
Multiple Discriminant Analysis

© CET
I.I.T. KGP

$$\left\{ \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 6 \end{pmatrix}, \begin{pmatrix} 7 \\ 5 \end{pmatrix} \right\} \in \omega_1$$
$$\left\{ \begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 10 \\ 1 \end{pmatrix}, \begin{pmatrix} 12 \\ 3 \end{pmatrix}, \begin{pmatrix} 13 \\ 6 \end{pmatrix} \right\} \in \omega_2$$

Find out the projection direction that maintains
separability between the two classes.

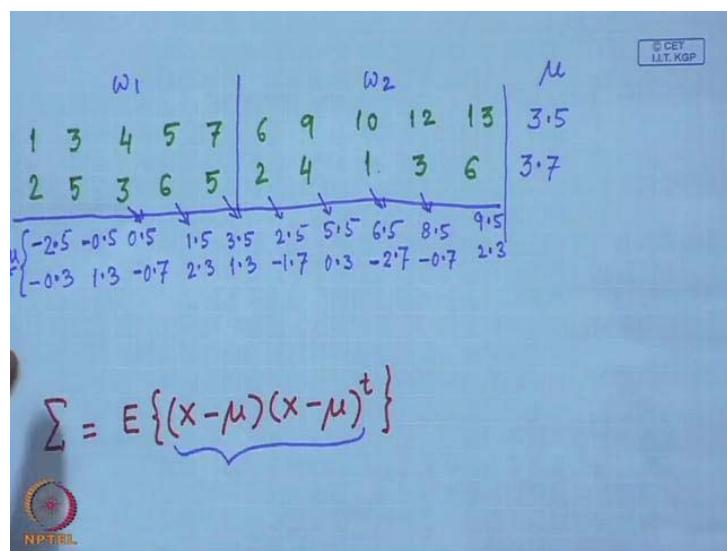
IIT Kharagpur

That there are 5 feature vectors which are taken from class ω_1 and there are 5 features which are taken from class ω_2 . And we want to find out a projection direction for reduction of the dimensionality that maintains separability between the two classes that is, even after taking the projection onto a line, the projections of the vectors belonging to class ω_1 and the projection of the vectors belonging to class ω_2 .

These two sets of projected points should still be linearly separable.

So, before solving this problem we have taken up the issue that, if I project for the best representation of the multi-dimensional data. And the for that we have seen earlier that the best way of projection is to project onto the Eigen vector. So, the first problem that we have tried to solve is what will happen to this projected points, if the projection is taken on an Eigen vector. So, for that let us briefly recapitulate what we have done.

(Refer Slide Time: 02:51)



So, this is the entire set of feature vectors. And I want to project this feature vectors onto the Eigen vector. So, I have to find out what is Eigen vector. So, for this feature vector we have computed the min vector. And we know that the co-variance matrix the definition of the co-variance matrix is $\Sigma = E\{(X-\mu)(X-\mu)^t\}$, where μ is this mean vector. So, for every vector we have computed $X-\mu$. So, these are $X-\mu$ for different feature vectors. And once I have this $X-\mu$, for every individual vector the $X-\mu$ we have computed $(X-\mu)(X-\mu)^t$ giving me a number of matrices.

(Refer Slide Time: 03:46)

© CET
I.I.T. KGP

$$\begin{pmatrix} -2.5 \\ -0.3 \end{pmatrix} \begin{pmatrix} -2.5 & -0.3 \end{pmatrix} = \begin{pmatrix} 6.25 & 0.75 \\ 0.75 & 0.09 \end{pmatrix} \rightarrow M_1$$

$$\begin{pmatrix} -0.5 \\ 1.3 \end{pmatrix} \begin{pmatrix} 0.5 & 1.3 \end{pmatrix} = \begin{pmatrix} 0.25 & -0.65 \\ -0.65 & 1.69 \end{pmatrix} \rightarrow M_2$$

$$\begin{pmatrix} 0.5 \\ -0.7 \end{pmatrix} \begin{pmatrix} 0.5 & -0.7 \end{pmatrix} = \begin{pmatrix} 0.25 & -0.35 \\ -0.35 & 0.49 \end{pmatrix} \rightarrow M_3$$

$$M_4 = \begin{pmatrix} 2.25 & 3.45 \\ 3.45 & 5.29 \end{pmatrix}$$

M₅

So, the first one $(X_1 - \mu)(X_1 - \mu)^t$ that gives me a matrix M_1 , $(X_2 - \mu)(X_2 - \mu)^t$ gives me another matrix M_2 . Similarly, $(X_3 - \mu)(X_3 - \mu)^t$ gives me another matrix M_3 . Similarly, I get M_4 and as I have total 10 number of feature vectors. So, I get 10 such matrices.

(Refer Slide Time: 04:16)

© CET
I.I.T. KGP

$$M_5 = \begin{pmatrix} 12.25 & 4.55 \\ 4.55 & 1.69 \end{pmatrix} \quad M_9 = \begin{pmatrix} 72.25 & -5.95 \\ -5.95 & 0.49 \end{pmatrix}$$

$$M_6 = \begin{pmatrix} 6.5 & -4.25 \\ -4.25 & 2.89 \end{pmatrix} \quad M_{10} = \begin{pmatrix} 90.25 & 21.85 \\ 21.85 & 5.29 \end{pmatrix}$$

$$M_7 = \begin{pmatrix} 30.25 & 1.65 \\ 1.65 & 0.09 \end{pmatrix}$$

$$M_8 = \begin{pmatrix} 42.25 & -17.55 \\ -17.55 & 7.2 \end{pmatrix}$$

M₅

So, once I have 10 such matrix I can compute the covariance matrix which is nothing but expectation value of all these 10 matrices or effectively, these are the average or mean of all this 10 matrices, which gives me the co-variance matrix.

(Refer Slide Time: 04:45)

The image shows a handwritten derivation of a covariance matrix. It starts with the formula for the covariance matrix $\Sigma = E \{(x - \mu)(x - \mu)^T\}$, where x is a vector and μ is the mean. This is then expanded as $= \frac{1}{10} \sum_{i=1}^{10} (x_i - \mu)(x_i - \mu)^T$. This sum is then equated to $= \frac{1}{10} \sum_{i=1}^{10} M_i$, where each M_i represents one of the 10 matrices. Finally, the resulting matrix is shown as a 2x2 matrix with elements 26.275 and 0.35 in the top row, and 0.35 and 2.53 in the bottom row, labeled as the Covariance Matrix.

$$\Sigma = E \{(x - \mu)(x - \mu)^T\}$$

$$= \frac{1}{10} \sum_{i=1}^{10} (x_i - \mu)(x_i - \mu)^T$$

$$= \frac{1}{10} \sum_{i=1}^{10} M_i$$

$$= \begin{pmatrix} 26.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix} \quad \text{Covariance Matrix}$$

So, I compute the co-variance matrix by taking the mean of all these 10 matrices. And the co-variance matrix came out to be this that is $\begin{pmatrix} 26.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix}$. Now, you note an interesting over here is, in all the matrices that we have computed starting from M_1 to M_{10} , this was the matrix. You find that elements (1, 2) and (2, 1) is same. And obviously in the co-variance matrix, also element, (1, 2) and (2, 1) is same, because this is the co-variance between the first element and the second element, x_1 and x_2 .

So, these two elements have to be. So, I get the co-variance matrix. And to get the Eigen vector of this co-variance matrix, I have to solve the Eigen value equation. So, first I have to find out the Eigen values of this co-variance matrix, out of that I have to choose the maximum Eigen value. And find out the Eigen vector corresponding to the maximum Eigen value. So, to find out the Eigen values of the co-variance matrix the equation is the procedure is well known you form a determinant of this form.

(Refer Slide Time: 06:17)

The image shows a handwritten derivation of eigenvalues for a 2x2 matrix. The matrix is given as:

$$\begin{vmatrix} 26.275 - \lambda & 0.35 \\ 0.35 & 2.53 - \lambda \end{vmatrix} = 0$$

This leads to the characteristic equation:

$$\Rightarrow (26.275 - \lambda)(2.53 - \lambda) - (0.35)^2 = 0$$
$$\Rightarrow \lambda^2 - 28.805\lambda + 66.3532 = 0$$

Solving the quadratic equation, the eigenvalues are found to be:

$$\lambda_1 = 26.28$$
$$\lambda_2 = 2.525$$

At the bottom left, there is a logo for IIT Kharagpur.

So, along the main diagonal if λ is an Eigen value then along the main diagonal from all the elements you subtract λ from a determinant and set this is determinant value equal to zero. So, I will get a number of equations. Solve those equations to find out the value of the λ . So, that is what I get over here. I get a quadratic equation you find that it is an equation which is quadratic in terms of λ . And because it is a quadratic equation so I will get 2 values of λ , two Eigen values.

So, I get λ_1 which is 26.28 and I get λ_2 which is 2.525. So this matrix, this co-variance matrix has two such Eigen values and out of this two Eigen values, I have to consider that Eigen value which is maximum. And corresponding to this Eigen value I have to find out the corresponding Eigen vector. So, obviously the matrices of size $n \times n$ I will have n number of Eigen values. So, it depends upon what is the dimensionality of the matrices.

(Refer Slide Time: 07:48)

The image shows handwritten mathematical work on a blue background. At the top, it says $\underline{A} \underline{X} = \lambda \underline{X}$. Below this is a matrix equation: $\begin{pmatrix} 26.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 26.28 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. This leads to a system of equations: $\begin{cases} 26.275 x_1 + 0.35 x_2 = 26.28 x_1 \\ 0.35 x_1 + 2.53 x_2 = 26.28 x_2 \end{cases}$. Solving this system, it is shown that $x_1 = 70$ and $x_2 = 1$. The final result is $\underline{X} = \underline{26.28}$.

So, for this Eigen value I try to compute the Eigen vector and for that we have this well-known Eigen value equation that for matrix A, if X is the Eigen vector and λ is the corresponding Eigen value then, this equation must be satisfied that is $A X = \lambda X$. In our case this matrices

A is the co-variance matrix which is this, right? $\begin{pmatrix} 26.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix}$. And this Eigen vector X is

$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ which will be equal to, so I took the maximum Eigen value which is 26.28.

So, $\begin{pmatrix} 26.275 & 0.35 \\ 0.35 & 2.53 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 26.28 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ I get two simultaneous equations. Solve this equation

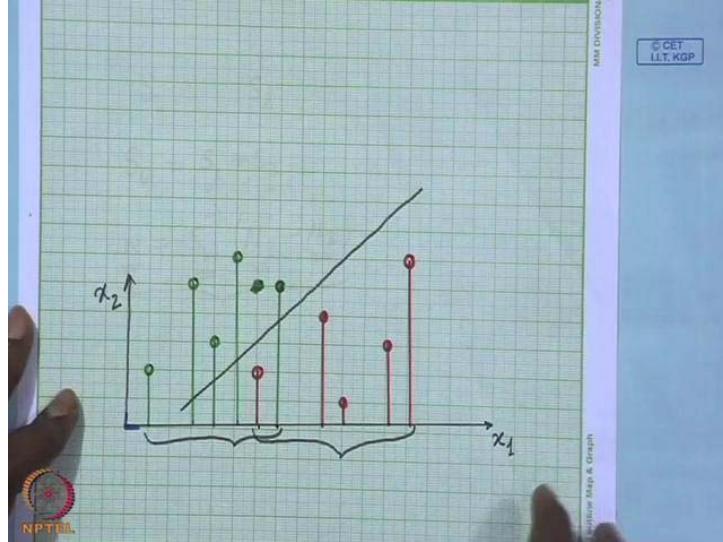
to get the values of x_1 and x_2 . So, here you find that x_1 would be is equal to 70, x_2 would be equal to 1. If I try to get the value from the other one the values will be slightly different and the values difference, the difference in values from these two equations is because of the truncation error. In many cases the decimal values beyond a certain positon are ignored.

So, I get a truncation error. So, because of the truncation error the $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ values that I get from

here. And the $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ values that I get from here, ideally this should be same, but I get a

difference because of the truncation error, but anyway the values will be very close instead of 70 here we may get 68 69 or something like that. So, this is my Eigen vector, Eigen vector is $\begin{pmatrix} 70 \\ 1 \end{pmatrix}$ then what we have done is...

(Refer Slide Time: 09:42)



We have plotted all the feature vectors onto a graph paper. And this is the Eigen vector $\begin{pmatrix} 70 \\ 1 \end{pmatrix}$

projected on all those feature vectors onto the Eigen vector and these are my projected points. And here as expected because we did not consider the separability of the feature vectors belonging to two different classes. We just wanted to find out projections directions that are a projections space that best represents the higher dimensional feature vectors into a lower dimensional feature vectors.

So, we said that it may not be possible to maintaining the separability between the classes. And exactly that is what has happened. You find that all this feature vectors belonging to class ω_2 their projected within this space. And all these feature vectors belonging to class ω_1 , they are projected within this region and these 2 regions now get merged. So, though in the original 2 dimensional space the feature vectors belonging to two classes ω_1 and ω_2 .

They were separable on the projected space they are no more separable. So, this is a problem that we face if we go for best way of representation or project the original set of data onto the Eigen space for reduction of dimensionality. So, to solve this problem what Fisher linear discriminator does is it tries to find out a projections space which maximizes the separability

between the classes or separation between different classes. And at the same time it tries to form compact clusters of the feature vectors belonging to individual classes.

So, what happens if I have two points clouds belonging to two different classes. When I project into a lower dimensional space, I try to see that these two classes are wide apart and at the same time the feature vectors belonging to individual class they are very compact. So, if I can maintain this, I can maintain the separability between the classes even in the projected space or in the lower dimensional space. So, in order to do that I have to consider two things one is I have to minimize the within class scatter so that every class is very, very compact.

And at the same time I have to maximize the between class scatter that is the classes should be pushed well apart. So, if I do them simultaneously then I get a sort of criteria function. So, after optimization of the criteria function I get a solution vector. So, this solution vector gives me a vector or a projection direction on which if I project higher dimensional feature vector even in the lower dimensional projected space the separability of the vectors can be guaranteed. Of course, when I tell all about this my assumption is in the initial space in the original higher dimensional space that data points are separated, they are separable.

If they are not separable in the original higher dimensional space, then whatever tricks I play I cannot guarantee that they will be separable in the lower dimensional projected space. So, that has to be kept in mind that when I talk about separability after projecting onto a lower dimensional space. I always assume that in the original higher dimensional space the feature vectors are well separated. So, then only the other things are valid. So, by optimization of that criteria functions which minimizes the within class scatter and maximizes the between class scatter. I get a solution which is of this form.

(Refer Slide Time: 14:17)

$$S_1 \quad S_2$$
$$S_w = S_1 + S_2$$
$$W = \underbrace{S_w^{-1}}_{\text{underlined twice}} (\underbrace{\mu_1 - \mu_2}_{\text{underlined}})$$

That is if S_1 is the within class scatter of the data points belonging to class ω_1 and S_2 is the scatter of the data points belonging to class ω_2 . Then $S_w = S_1 + S_2$ this is called the total within class scatter, that is the sum of the scatters of the samples belonging to class ω_1 and the samples belonging to class ω_2 . So, I take these two scatters individually then add this two I get total within class scatter which is S_w , μ_1 is the mean of the samples belonging to class ω_1 and μ_2 is the mean of the samples belonging to class ω_2 .

So, once I have this then, the direction of the projection or the projection vector which maintains separability will be given by $W = S_w^{-1}(\mu_1 - \mu_2)$. So, this solution has been obtained by solving a generalized Eigen value expression. So, to illustrate this I will take up the same problem for which we have gone for projection onto the Eigen space. So, that is we have two sets of samples, two sets of feature vectors one set of feature vector belonging to class ω_1 , the other set of feature vector belonging to class ω_2 .

(Refer Slide Time: 16:05)

The image shows handwritten mathematical notes on a blue background. On the left, there are two sets of feature vectors labeled $\omega_1 \rightarrow$ and $\omega_2 \rightarrow$. The first set, $\omega_1 \rightarrow$, contains vectors $\begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 6 \end{pmatrix}, \begin{pmatrix} 7 \\ 5 \end{pmatrix}$. The second set, $\omega_2 \rightarrow$, contains vectors $\begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 10 \\ 1 \end{pmatrix}, \begin{pmatrix} 12 \\ 3 \end{pmatrix}, \begin{pmatrix} 13 \\ 6 \end{pmatrix}$. To the right of these vectors, the mean vectors $\mu_1 = \begin{pmatrix} 4 \\ 4.2 \end{pmatrix}$ and $\mu_2 = \begin{pmatrix} 10 \\ 3.2 \end{pmatrix}$ are listed. Next to the mean vectors, the scatter matrices are defined as $S_1 = \sum_{x_i \in \omega_1} (x_i - \mu_1)(x_i - \mu_1)^t$ and $S_2 = \sum_{x_i \in \omega_2} (x_i - \mu_2)(x_i - \mu_2)^t$. In the top right corner, there is a small logo for "DGET IIT-KGP".

So, the feature vectors which belong to class ω_1 are $\binom{1}{2}, \binom{3}{5}, \binom{4}{3}, \binom{5}{6}, \binom{7}{5}$. So, these are the 5 feature vectors which are taken from class ω_1 . Similarly, $\binom{6}{2}, \binom{9}{4}, \binom{10}{1}, \binom{12}{3}, \binom{13}{6}$. These are the 5 feature vectors which are taken from class ω_2 , and the scatter of data points belonging to a particular class they are defined by this. It is $S_1 = \sum_{x_i \in \omega_1} (X_i - \mu_1)(X_i - \mu_1)^t$ where μ_1 is the mean of the feature vectors belonging to class ω_1 .

So, for every vector belonging to class ω_1 , I have to compute every vector X belonging to class ω_1 , I have to compute $(X - \mu_1)(X - \mu_1)^t$. So, when I have n number of feature vectors in belonging to class ω_1 I will have each of this $(X - \mu_1)(X - \mu_1)^t$ gives me matrix. So, if I have n number of feature vectors in class ω_1 , I will have n number of such matrices.

And sum of all those matrices gives me the scatter of the samples in class ω_1 . Similarly, here for every vector belonging to class ω_2 , $(X - \mu_2)(X - \mu_2)^t$ that gives me one matrix.

And If there are m numbers of feature vectors in class ω_2 , I will get m such matrices. And sum of all these m matrices gives me the scatter of the samples belonging to class ω_2 . So, that is

how I compute scatter for class 1 and I compute the scatter from for class 2, S_1 and S_2 . And you find that there is this scatter and the co-variance matrix they are very, very similar.

If I normalize the scatter matrix by the number of samples that I have, I get a co-variance matrix because co-variance matrix is nothing but expectation value of this term, which is nothing but take the summation of all these matrices normalize by the number of data elements I have or number of matrices that I found I get a co-variance matrix. So, they give more or less same information in one case it is normalized in the other case it is not normalized.

So, what I do is I consider this feature vectors belonging to class ω_1 and I compute the mean of all this feature vectors because finally, I have to compute this term. So, I compute the mean

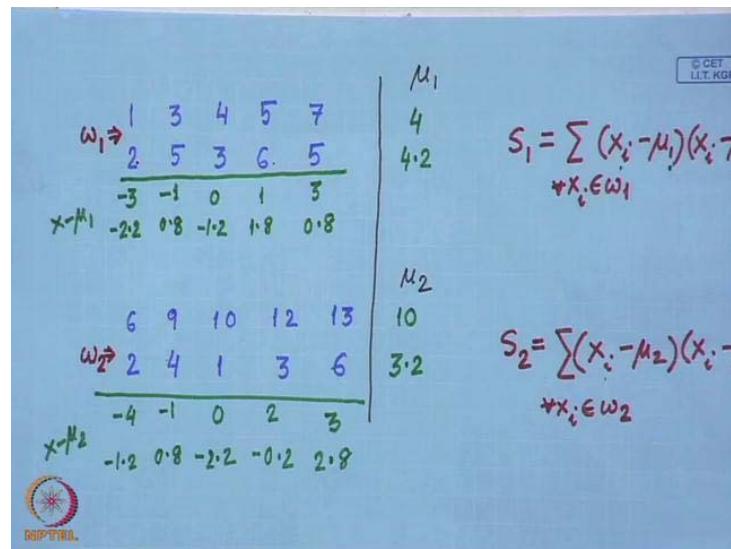
of this feature vectors which is μ_1 and μ_1 in this case will be $\begin{pmatrix} 4 \\ 4.2 \end{pmatrix}$. And then considering

these vectors from class ω_2 , the mean of this feature vectors which is $\mu_2 = \begin{pmatrix} 10 \\ 3.2 \end{pmatrix}$ which you

can easily compute from here. Take the summations of all these components divide by 5, I get 10.

Take the summation of all these second components divide by 5 you get 3.2. And similarly, from here and the next one I have to compute is for each of the data points, for each of this feature vectors I have to compute $(X - \mu_1)$. For each of the feature vectors over here I have to compute $(X - \mu_2)$. So, that I can compute $(X - \mu_1)(X - \mu_1)^t$ In the other case $(X - \mu_2)(X - \mu_2)^t$.

(Refer Slide Time: 20:28)



So, over here you find that for all the samples belonging to class ω_1 , $(X - \mu_1)$ when I compute.

So, $(X - \mu_1)$ over here it will be simply $\begin{pmatrix} -3 \\ -2.2 \end{pmatrix}$. Similarly, over here for the second one it is

$\begin{pmatrix} -1 \\ 0.8 \end{pmatrix}$. Similarly, over here this will be $\begin{pmatrix} 0 \\ -1.2 \end{pmatrix}$. Here it will be $\begin{pmatrix} 1 \\ 1.8 \end{pmatrix}$ and here it will be $\begin{pmatrix} 3 \\ 0.8 \end{pmatrix}$

. So, these are $(X - \mu_1)$ for individual vectors X. In the same manner I have to find out $(X - \mu_2)$ for individual vectors X belonging to class ω_2 .

So, let us see what those values will be for the second one. So, here I have to compute $(X - \mu_2)$

. So, the first one it will be $\begin{pmatrix} -4 \\ -1.2 \end{pmatrix}$, then $\begin{pmatrix} -1 \\ 0.8 \end{pmatrix}$, $\begin{pmatrix} 0 \\ -2.2 \end{pmatrix}$. Similarly, here $\begin{pmatrix} 2 \\ -0.2 \end{pmatrix}$, $\begin{pmatrix} 3 \\ 2.8 \end{pmatrix}$.

So, I have got $(X - \mu)$ for every individual X belonging to class ω_2 . And Now, I have to go for computation of the scattered matrices S_1 and S_2 . So, for computation of S_1 I have to find out

what is $\sum_{x_i \in \omega_1} (x_i - \mu_1)(x_i - \mu_1)^t$. So, the first one is $\begin{pmatrix} -3 \\ -2.2 \end{pmatrix}$.

(Refer Slide Time: 23:17)

So, let us see that what will be this, first one is $\begin{pmatrix} -3 \\ -2.2 \end{pmatrix}$, this is $(X_1 - \mu)(X_1 - \mu)^t$ will be

$\begin{pmatrix} -3 \\ -2.2 \end{pmatrix} \begin{pmatrix} -3 & -2.2 \end{pmatrix}$. And if you compute this value the matrix will be $\begin{pmatrix} 9 & 6.6 \\ 6.6 & 4.84 \end{pmatrix}$. This is one

of the matrices, the second one was

$\begin{pmatrix} -1 \\ 0.8 \end{pmatrix}$. If you remember this it was $\begin{pmatrix} -1 \\ 0.8 \end{pmatrix}$. So, it is $\begin{pmatrix} -1 \\ 0.8 \end{pmatrix} \begin{pmatrix} -1 & 0.8 \end{pmatrix}$. It will be simply

$\begin{pmatrix} 1 & -0.8 \\ -0.8 & 0.64 \end{pmatrix}$. For the third one it is $\begin{pmatrix} 0 \\ -1.2 \end{pmatrix} \begin{pmatrix} 0 & -1.2 \end{pmatrix}$.

So, if I compute this it will be $\begin{pmatrix} 0 & 0 \\ 0 & 1.44 \end{pmatrix}$. For the fourth one it is $\begin{pmatrix} 1 \\ 1.8 \end{pmatrix}$. So, I will have

$\begin{pmatrix} 1 \\ 1.8 \end{pmatrix} \begin{pmatrix} 1 & 1.8 \end{pmatrix}$. So, if I multiply this it will be $\begin{pmatrix} 1 & 1.8 \\ 1.8 & 3.24 \end{pmatrix}$. And then fifth one $(X - \mu_1)$ was

$\begin{pmatrix} 3 \\ 0.8 \end{pmatrix}$. So, if I compute this $\begin{pmatrix} 3 \\ 0.8 \end{pmatrix} \begin{pmatrix} 3 & 0.8 \end{pmatrix}$ if I compute this it will be $\begin{pmatrix} 9 & 2.4 \\ 2.4 & 0.64 \end{pmatrix}$. So, these

are the 5 matrices that I have and from these 5 matrices I have to compute the scatter is S_1 which is nothing but sum of all these 5 matrices.

And if I take the sum you find that the first component would be $9 + 1 + 1 + 9 + 2$ which is equal to 20. Similarly, this is $6.6 - .8 + .8 + 0 + 1.8 + 2.4$. So, this will be simply 10, this term will also

be equal to 10 and here I will have 10.8 which is nothing but sum $4.84 + 0.64 + 1.44 + 3.24 + 0.64$ which comes out to be 10.8. So, this is my scatter matrix

$S_1 = \begin{pmatrix} 20 & 10 \\ 10 & 10.8 \end{pmatrix}$. Similarly, let us try to compute what will be the scatter matrix S_2 . So, for

computation of scatter matrix S_2 I have $(X - \mu_2)$ over here for every X belonging to class ω_2 so

that is given by this. So, I have to compute the scatter matrix S_2 . So, the first one is $\begin{pmatrix} -4 \\ -1.2 \end{pmatrix}$.

(Refer Slide Time: 28:21)

© CET
I.I.T. KGP

$$\begin{aligned} \begin{pmatrix} -4 \\ -1.2 \end{pmatrix} \begin{pmatrix} -4 & -1.2 \end{pmatrix} &= \begin{pmatrix} 16 & 4.8 \\ 4.8 & 1.44 \end{pmatrix} \\ \begin{pmatrix} -1 \\ 0.8 \end{pmatrix} \begin{pmatrix} -1 & 0.8 \end{pmatrix} &= \begin{pmatrix} 1 & -0.8 \\ -0.8 & 0.64 \end{pmatrix} \\ \begin{pmatrix} 0 \\ -2.2 \end{pmatrix} \begin{pmatrix} 0 & -2.2 \end{pmatrix} &= \begin{pmatrix} 0 & 0 \\ 0 & 4.84 \end{pmatrix} \\ \begin{pmatrix} 2 \\ -0.2 \end{pmatrix} \begin{pmatrix} 2 & -0.2 \end{pmatrix} &= \begin{pmatrix} 4 & -0.4 \\ -0.4 & 0.04 \end{pmatrix} \\ \begin{pmatrix} 3 \\ 2.8 \end{pmatrix} \begin{pmatrix} 3 & 2.8 \end{pmatrix} &= \begin{pmatrix} 9 & 8.4 \\ 8.4 & 7.84 \end{pmatrix} \end{aligned} \quad \left| \quad S_2 = \begin{pmatrix} 30 & 12 \\ 12 & 14.8 \end{pmatrix} \right.$$

So, the first matrix that I get is $\begin{pmatrix} -4 \\ -1.2 \end{pmatrix} \begin{pmatrix} -4 & -1.2 \end{pmatrix}$. So, that simply becomes $\begin{pmatrix} 16 & 4.8 \\ 4.8 & 1.44 \end{pmatrix}$. So,

this is what I get from the first vector. The second vector is minus 1 1.8. So, the matrix that I

get from the second vector is $\begin{pmatrix} -1 \\ 0.8 \end{pmatrix}$. So, it is $\begin{pmatrix} -1 \\ 0.8 \end{pmatrix} \begin{pmatrix} -1 & 0.8 \end{pmatrix}$. So, this matrix will be

$\begin{pmatrix} 1 & -0.8 \\ -0.8 & 0.64 \end{pmatrix}$. For the third one which is $\begin{pmatrix} 0 \\ -2.2 \end{pmatrix}$. So, it is $\begin{pmatrix} 0 \\ -2.2 \end{pmatrix} \begin{pmatrix} 0 & -2.2 \end{pmatrix}$. So, this will

simply be $\begin{pmatrix} 0 & 0 \\ 0 & 4.84 \end{pmatrix}$.

So, this is my third matrix. To compute the fourth matrix, it is $\begin{pmatrix} 2 \\ -0.2 \end{pmatrix}$. So, what I have is

$\begin{pmatrix} 2 \\ -0.2 \end{pmatrix}(2 \quad -0.2)$. And if I multiply these two what I get is $\begin{pmatrix} 4 & -0.4 \\ -0.4 & 0.04 \end{pmatrix}$, this is the fourth

matrix. And then to get the fifth matrices I have to take this one $\begin{pmatrix} 3 \\ 2.8 \end{pmatrix}$. So, this is

$\begin{pmatrix} 3 \\ 2.8 \end{pmatrix}(3 \quad 2.8)$, multiply these two what I get is $\begin{pmatrix} 9 & 8.4 \\ 8.4 & 7.84 \end{pmatrix}$. So, these are the 5 matrices that

I get from the samples, the 5 samples belonging to class ω_2 .

Now, sum of all these 5 matrices will give me the scatter S_2 . So, if you take the sum scatter S_2 will be given by $\sum_{\forall X_i \in \omega_2} (X_i - \mu_2)(X_i - \mu_2)^t$. So, I have S_2 which is nothing but sum of all these 5

matrices. So, over here it will be $16+1+4+9=30$. So, the first component will be 30, second components $4.8-0.8-0.4=4.4$. So, this will be 12. Similarly, here it will be 12 and this component will be $1.44+0.64+4.84+0.04+7.84$. So, this will be simply 14.8.

So, you find that I get two scattered matrices one for class ω_1 which is S_1 the other scatter matrix for class ω_2 which is S_2 . So, for class ω_1 the scatter matrix S_1 is given by this and for class ω_2 the scatter matrix S_2 is given by this. Now, from S_1 and S_2 , I have to find out what is the total within class scatter that is S_w which is nothing but S_1+S_2 .

(Refer Slide Time: 33:43)

G.CET
I.I.T. KGP

$$S_1 = \begin{pmatrix} 20 & 10 \\ 10 & 10.8 \end{pmatrix} \quad S_2 = \begin{pmatrix} 30 & 12 \\ 12 & 14.8 \end{pmatrix}$$

$$S_w = S_1 + S_2$$

$$= \begin{pmatrix} 50 & 22 \\ 22 & 25.6 \end{pmatrix}$$

$$W = S_w^{-1} (\underline{\mu}_1 - \underline{\mu}_2)$$

S_w^{-1} ?

So, what I had is I had S_1 , I will rewrite here for convenience, I had $S_1 = \begin{pmatrix} 20 & 10 \\ 10 & 10.8 \end{pmatrix}$ and

$S_2 = \begin{pmatrix} 30 & 12 \\ 12 & 14.8 \end{pmatrix}$. So, I have to compute the total within class scatter which is S_w , which is

nothing but $S_1 + S_2$. And this will be simply if I add this two matrices $S_w = \begin{pmatrix} 50 & 22 \\ 22 & 25.6 \end{pmatrix}$. So, this

is my total within class scatter. And from this what I have to compute is S_w^{-1} because finally projection direction, I have to compute the projection direction.

And the projection direction $W = S_w^{-1}(\mu_1 - \mu_2)$ where μ_1 is the mean of the samples belonging to class ω_1 and μ_2 is the mean of the samples belonging to class ω_2 . So, from this total within class scatter is S_w , I have to compute what is S_w^{-1} . So, this is what I need to compute. Now, for a two dimensional matrix, 2 by 2 matrix computation of inverse is very simple. So, if I have a two dimensional matrices something like this.

(Refer Slide Time: 35:47)

The image shows a handwritten derivation of the inverse of a 2x2 matrix A . It starts with the definition of matrix A as:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Then, it shows the formula for the inverse of A :

$$A^{-1} = \frac{1}{|A|} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

This is then simplified to:

$$= \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

The background of the image is a blue surface with a small logo and text "NPTEL" at the bottom left.

Say, matrix A which is given by $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, this is a 2 dimensional matrices. Then

$A^{-1} = \frac{1}{|A|} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$ and this $|A|$ is nothing but $(ad - bc)$. So, which is simply

$$A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

So, this is what is the inverse of a 2x2 matrices, where the 2x2 matrix

is given by elements a, b, c, d. So, simply using this expression when I know that my total within class scatter $S_w = \begin{pmatrix} 50 & 22 \\ 22 & 25.6 \end{pmatrix}$.

(Refer Slide Time: 37:04)

$$\begin{aligned}
 S_w &= \begin{pmatrix} 50 & 22 \\ 22 & 25.6 \end{pmatrix} \\
 S_w^{-1} &= \frac{1}{\begin{vmatrix} 50 & 22 \\ 22 & 25.6 \end{vmatrix}} \begin{pmatrix} 25.6 & -22 \\ -22 & 50 \end{pmatrix} \\
 &= \frac{1}{796} \begin{pmatrix} 25.6 & -22 \\ -22 & 50 \end{pmatrix} \\
 &= \boxed{\begin{pmatrix} 0.032 & -0.03 \\ -0.03 & 0.06 \end{pmatrix}} \leftarrow S_w^{-1}
 \end{aligned}$$

This is the total within class scatter I can very easily compute the inverse of the total within class scatter that is S_w inverse. So, over here this is S_w^{-1} will be given by

$$S_w^{-1} = \frac{1}{\begin{vmatrix} 50 & -22 \\ -22 & 25.6 \end{vmatrix}} \begin{pmatrix} 25.6 & -22 \\ -22 & 50 \end{pmatrix}$$

And if you compute this determinant you find that this

determinant is $50 \times 25.6 - 22 \times 22$ which comes out to be 796. And if you complete this computation the inverse of the matrix will come out to be $\begin{pmatrix} 0.032 & -0.03 \\ -0.03 & 0.06 \end{pmatrix}$. So,

$S_w^{-1} = \begin{pmatrix} 0.032 & -0.03 \\ -0.03 & 0.06 \end{pmatrix}$. So, this is the matrix which is important to me that is S_w^{-1} . And the other

quantity that I have to compute is $(\mu_1 - \mu_2)$.

(Refer Slide Time: 39:38)

$\mu_1 = \begin{pmatrix} 4 \\ 4.2 \end{pmatrix}$

$\mu_2 = \begin{pmatrix} 10 \\ 3.2 \end{pmatrix}$

$\mu_1 - \mu_2 = \begin{pmatrix} -6 \\ 1 \end{pmatrix}$

And we know from before that $\mu_1 = \begin{pmatrix} 4 \\ 4.2 \end{pmatrix}$ and $\mu_2 = \begin{pmatrix} 10 \\ 3.2 \end{pmatrix}$. So, these were the means of the samples belonging to class ω_1 and the mean of the samples belonging to class ω_2 . So, I can easily find out that $(\mu_1 - \mu_2)$ will be equal to $\begin{pmatrix} -6 \\ 1 \end{pmatrix}$.

(Refer Slide Time: 40:30)

$$\vec{e} = S_w^{-1} (\mu_1 - \mu_2)$$

$$= \begin{pmatrix} 0.032 & -0.03 \\ -0.03 & 0.06 \end{pmatrix} \begin{pmatrix} -6 \\ 1 \end{pmatrix}$$

$$= \boxed{\begin{pmatrix} -0.222 \\ 0.24 \end{pmatrix}}$$

So, once I have this the projection direction is simply given by, if I write it as a vector $\vec{e} = S_w^{-1} (\mu_1 - \mu_2)$ that is the direction of projection where S_w^{-1} . You remember from it is

$S_w^{-1} = \begin{pmatrix} 0.032 & -0.03 \\ -0.03 & 0.06 \end{pmatrix}$. Of course, we are not considering the higher decimal spaces. So, it is

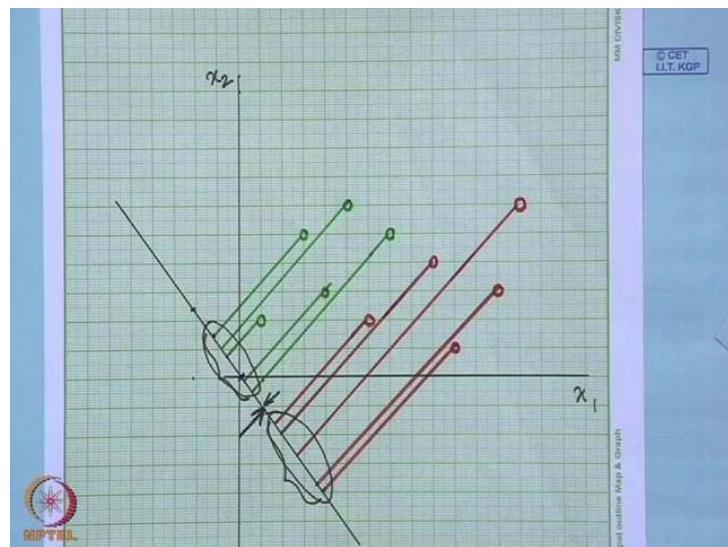
$\begin{pmatrix} 0.032 & -0.03 \\ -0.03 & 0.06 \end{pmatrix} \begin{pmatrix} -6 \\ 1 \end{pmatrix}$. So, you multiply this by the column vector $\begin{pmatrix} -6 \\ 1 \end{pmatrix}$.

And if you compute this you will find that this one will come out to be $\begin{pmatrix} -0.222 \\ 0.24 \end{pmatrix}$. So, the

projection direction or vector on which if I take the projection which will maintain the separability between the classes should be this one, which is $\begin{pmatrix} -0.222 \\ 0.24 \end{pmatrix}$. So, again let us plot

these points on a graph paper and take the projections on to this vector. And let us see what improvement we get at all if we get some improvement. So, I will take this graph paper. So, as before let me assume that say this is my origin. So, I will draw x axis and y axis over here or x_1 and x_2 axis.

(Refer Slide Time: 43:21)



So, this is x_1 axis and this is x_2 axis. Let us put the vectors belonging to class ω_1 . I will use this green color for that, the vectors are $\begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \begin{pmatrix} 4 \\ 5 \end{pmatrix}, \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \begin{pmatrix} 7 \\ 6 \end{pmatrix}, \begin{pmatrix} 5 \\ 5 \end{pmatrix}$ from class ω_1 and

$\begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 10 \\ 1 \end{pmatrix}, \begin{pmatrix} 12 \\ 3 \end{pmatrix}, \begin{pmatrix} 13 \\ 6 \end{pmatrix}$ from class ω_2 . So, for class ω_1 it is $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$, the next one is $\begin{pmatrix} 3 \\ 5 \end{pmatrix}$, next one is $\begin{pmatrix} 4 \\ 3 \end{pmatrix}$ that is over here the next one is $\begin{pmatrix} 5 \\ 6 \end{pmatrix}$ which is over here, the next one is $\begin{pmatrix} 7 \\ 5 \end{pmatrix}$.

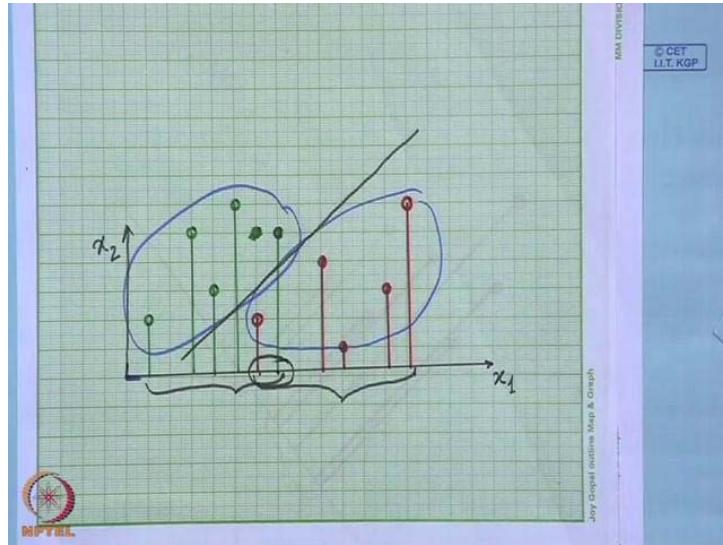
So, these are the vectors which are taken from class ω_1 , similarly if I take the vectors from class ω_2 , which is $\begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 10 \\ 1 \end{pmatrix}, \begin{pmatrix} 12 \\ 3 \end{pmatrix}, \begin{pmatrix} 13 \\ 6 \end{pmatrix}$. So, I will take $\begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 6 \\ 2 \end{pmatrix}$ is over here then $\begin{pmatrix} 9 \\ 4 \end{pmatrix}$, then comes $\begin{pmatrix} 10 \\ 1 \end{pmatrix}$, then comes $\begin{pmatrix} 12 \\ 3 \end{pmatrix}$ and then $\begin{pmatrix} 13 \\ 6 \end{pmatrix}$. So, these are the vectors from class ω_2 .

And my projections direction as we have computed is $\begin{pmatrix} -0.222 \\ 0.24 \end{pmatrix}$. So, I will put it as let us truncate this point say $\begin{pmatrix} -0.22 \\ 0.24 \end{pmatrix}$.

So, because I am interested only in the direction. So, even if I multiply this by 10 then does not matter. So, I will take it as $\begin{pmatrix} -2.2 \\ 2.4 \end{pmatrix}$. So, I will take this point -2.2 or if -22 if I multiply this by 100 even that does not matter. So, I will make it -22 and then +24. So, -22 is somewhere over here +24 is over here. So, this is the point so my vector is, this is the direction of projection.

Now, let us project or take the perpendicular projection of all the feature vectors belonging to class ω_1 and the feature vector belonging to class ω_2 on to this projection direction. So, this will be, if I take perpendicular projection. So, these are the projections. Now, we find when I projected it the feature vectors belonging to class ω_1 onto this projection vector and the feature vectors belonging to class ω_2 onto the projection vector they are not mixed anymore there are well separated. So, over here this is the region in which the feature vectors belonging to class ω_1 they get projected and this is the region over which the feature vectors belonging to class ω_2 they are projected. And these 2 regions are well separated. So, the projected points are no more mixed.

(Refer Slide Time: 50:12)



Unlike in the previous case if you remember this particular figure where the feature vectors from 2 different classes they are mixed in this region. So, the projected points are no more separable, but if I maintain that constraint that when I take try to take the projection directions I will maintain the separability I will try to maximize the separability between the classes or I will try to maximize the between class scatter. And I will try to minimize the within class scatter.

So, while doing so the solution vector or the projection direction that I get is this and here it is quite clear. This demonstrates that on this direction if I take the projections even in the projected space. The Feature vectors in different classes they are well separated. So, this solves one part of the problem that is happened able to reduce the dimensionality, but for classification application, I need to go further that is I have to design a classifier which works in this projected space.

So, how do I do that? Clearly, over here these are one dimensional variable all these projected points if I take the distance of these points from the center these are one dimensional variable. So, I get a set of one dimensional value or scalar values over here. I also get a set of scalar values over here. So, I take this set of scalar values to belong to one class and this set of scalar values that belonging to another class. And for classification purpose what I need to do is. I need to identify a point on this line which will separate between these two classes.

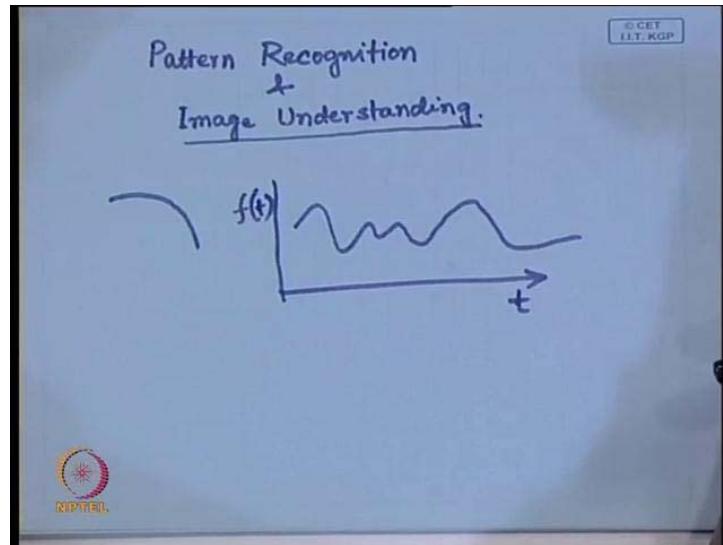
So, if this point follows a particular distribution I can find out what is the mean and variance of this set of points. I can also find out what is the mean and variance of this set of points. And from there I can identify what should be the threshold of this line. So, this is not a big problem, but it is a very simple problem. So, through what we have done is, we have reduced the problem from a higher dimensional space to a lower dimensional space and the lower dimensional space the problem is more manageable. So, I stop this lecture here.

Thank you.

Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

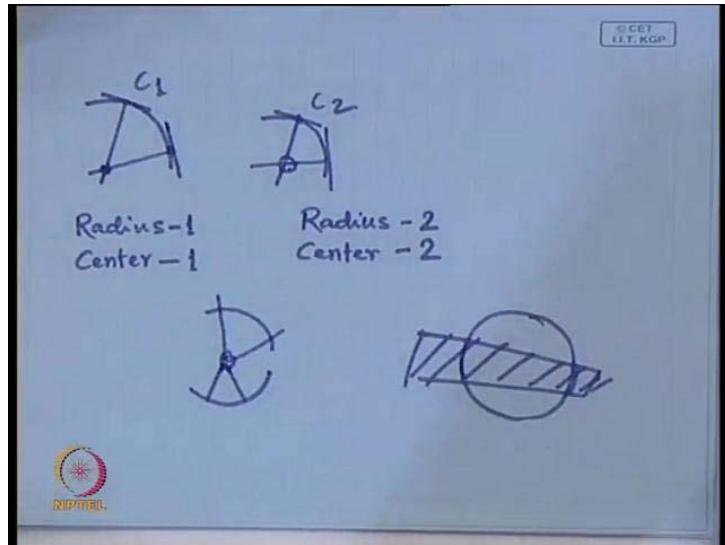
Lecture - 2
Feature Extraction – 1

(Refer Slide Time: 00:26)



So, let us take a very simple example to start with. Suppose that we have a circular arc say something like this. So, this circular arc is also a pattern. Similarly, if I have some function something like this, some variation of signal, so you can say this is a signal which varies with time. So, I can say that it is a function $f(t)$ which varies with time t . So, this is also a pattern. Now, given different such patterns, which may be similar or which may be dissimilar. How do we say that 2 patterns are similar or 2 patterns are dissimilar, that is the basic problem of pattern recognition. So, whenever I say or I can find out that one pattern is very, very similar to another given pattern, then I say that the first pattern is recognized as the second pattern, which is nothing but the knowledge that we have or something that we have in our knowledge base.

(Refer Slide Time: 01:35)



So, coming to this simple problem, say circular pattern recognition say given two circular arcs. One circular arc is like this. Another circular arc is something like this. So, how can we say that these 2 circular arcs are same or they are part of the same circle. So, as you know that if we go to parametric domain a circle is defined by a parameter, which is nothing but the radius of the circle.

And if I do not consider the case of translation in variance, then I will have 2 parameters. One is the center of the circle and other one is the radius of the circle. So, if I describe the circle only by the radius in that case, the circle is translation in variant, whereas along with radius. If I also specify the center of the circle in that case, the circle is translation invariant because the moment I translate the circle from one location to another location. Its center is going to change even though the radius will remain the same.

So, that is why we have different kinds of features, the translation invariant features, rotation invariant features. Of course, if you rotate a circle, it will, the features will also remain rotation invariant whereas once I specify the center of the circle, it is no more translation invariant because the moment you translate the circle, or you change the circle from one position to position to another, the center of the circle is going to change. So, given these 2

circular arcs, if I want to say whether these 2 circular arcs are same or part of the same circle or these circular arcs are similar, then what I have to do?

I have to find out what is the radius of the circle. I have to find out what is the location of the center of the circle. So, naturally to find out the radius of the circle, it is a matter of school level mathematics. What we will usually do is, we will try to find out a perpendicular at one point on the perimeter on the circle, so you draw a perpendicular like this. Similarly, you take another point on the perimeter of the circle. You also draw a perpendicular to the circle at this particular location and the point of intersection of these 2 perpendiculars is nothing but the center of the arc.

Once I get the center, then finding out the radius is very simple. It is nothing but the distance of a point on the perimeter from the center. So, I can easily find out the radius. Also, I can also find out the center. Similarly, if I perform the same task for the second circular arc, I can find out what is the location of the center or taking perpendiculars from 2 points on the circle, I can find out what is the location of the center. And once I have the location of the center, I can find out what is the radius of the circle.

So, in this case, let me call it this is radius 1 because it is of the first circular arc. Let me call it as center 1 because it is the center of the first circular arc. Similarly, in this case, let me call it as the radius 2 because it is the radius of the second circular arc and the center of the second circular arc. Now, if I find that radius 1 is same as radius 2, these 2 radii are equal.

In most of the cases, I cannot get the condition that these 2 are exactly equal because there is error of quantization, error of measurement. So, because of those errors and similarly, there is also error in segmentation, it is very difficult to get perfect segmentation. If the segmentation is erroneous, then obviously the position of the circular arc or the points, which belong to circular arc, may deviate by 1 or 2 pixels. So, in such case, it is extremely difficult that even if they are part of similar circle, the radii are same. But, after doing this computation, radius 1 may not be exactly same as radius 2.

So, what I have to do is I have to find out a measure of similarity, which is nothing but it is the difference between radius 1 and radius 2. So, if I find that the difference between radius 1 and radius 2 is very small that is negligible, I assume that they are same. So, in which case, I can say that this circular arc c_2 and this circular arc c_1 , they are part of the same circle or their radii are same. Whereas if the positions of the centers are different, then

though the circles are same, these 2 circular arcs are not the part of the same circle. So, one circle is translated with respect to another, whereas if I have a case something like this.

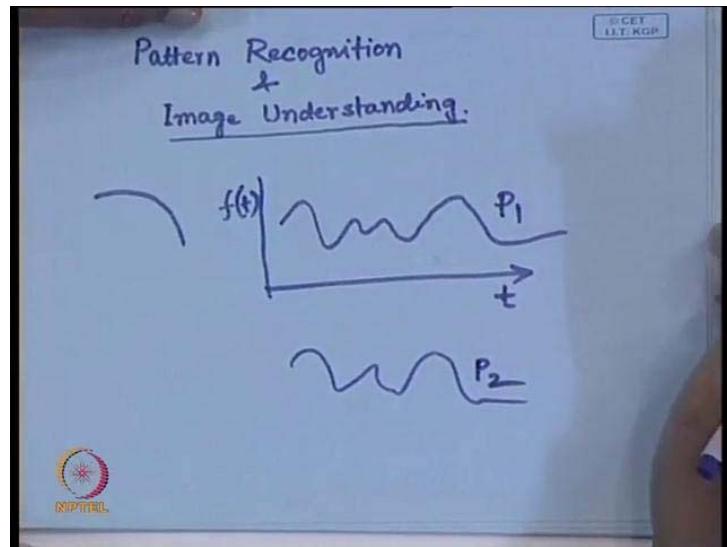
So, here what I am trying to do is I am trying to depict a case. Where I have 2 different circular arcs of the same circle, but these 2 circular arcs are not connected. They may not be connected because of various reasons. I can have a situation something like this say I have a circle. But on this circle, I have another object. So, this part is occluded. So, this object is on top of the circle. And when I take the image from the top side, naturally this part of the circular arc and this part of the circular arc or this part of the periphery will not be visible to a camera.

So, once I do this segmentation, I will get a situation something like this. So, those thought these 2 circular arcs are part of the same circle, but they become discontinuous. Now, in this case, I want to find out whether they are part of the same circle or not. I will perform similar type of operation. Find out the centroid location, location of the center. Here, you will find that the location of the center considering this arc and considering this arc, they will be same or they may not be exactly same. But they will be very close to each other or distance between these centers is very, very small.

Whereas so, it in this case, the center will be very close. And the radius, the length of the radius will also be similar. So, in this case, we can say that both this circular arcs are part of the same circle. So, what I am trying to do is, whenever I want to recognize a pattern, what I have to do is, I have to extract certain features of the pattern. And using those features in the feature domain, I have to find out whether the patterns are similar, the patterns are same or the patterns are different. So, over here in this particular example, I have taken this a very simple case of circular arc.

So, I have simply calculated the length of the radius and the position of the center. Whereas over here, this is not a simple pattern like a circular arc. So, in these cases, we have to find out that what are the kinds of features that we should be able to extract which is useful for recognition purpose or which are useful for classification purpose? So that given another pattern of similar form. I can say that this P_1 and this pattern P_2 .

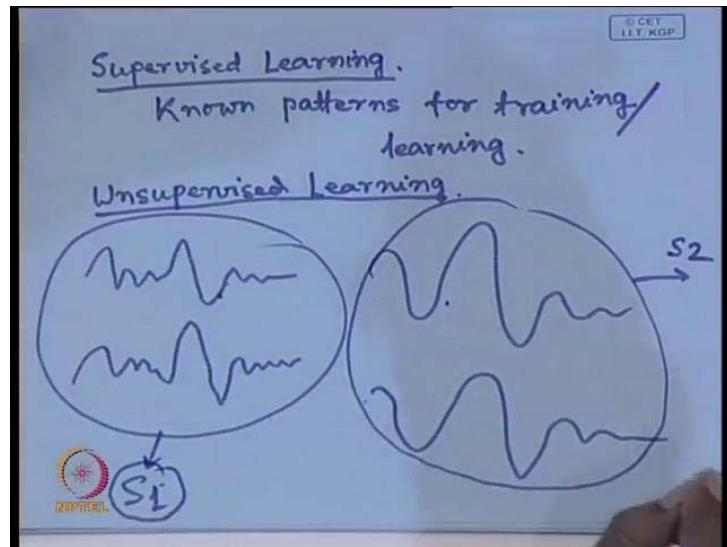
(Refer Slide Time: 09:47)



Whether these 2 patterns are same. They are similar or they are widely different. So, this pattern recognition problem is nothing but a matching problem. So, whenever you find that the similarity between 2 patterns is very high or the dissimilarity is very low, in that case, take one of the patterns as a reference or as a model, which is there in our knowledge base. I say that the second pattern is recognized or associated with the first one.

So, in any recognition problem, I have to have set knowledge or set of model. Given an unknown pattern, I have to identify that unknown pattern with one of the pattern, which is known to us. So, that is basically the pattern recognition problem. And to generate these known patterns, obviously I said that we have different kinds of learning. One is supervised learning and the other one is unsupervised learning.

(Refer Slide Time: 10:55)



So, in case of supervised learning, I have to have a set of patterns, which are known. So, a set of similar such patterns, may be this pattern is generated by some measuring instrument. So, I have to take similar such patterns from the same measuring instrument and number of such patterns and whatever feature I extract for one pattern. I have to generate similar features for all other patterns. And using these features, I have to generate a model that is a feature vector, which is representative of these class of patterns. That is the representative feature vector, which has to be kept in the knowledge first.

So, any time a new pattern comes, I will generate similar such feature vector and compare these 2 feature vectors. What is the distance between these 2 feature vectors? So, if the distance between the 2 features of vectors is very small, I will say that this unknown pattern is same as the pattern that I have in my knowledge base. If the distance is large, I will say this unknown pattern is different from the pattern that I have in my knowledge base.

So, in case of supervised learning, what I have to do is, I have to take a set of known patterns for which I have to find the feature vectors and for those feature vectors, I have to form a representative feature vector which represents the same class of patterns. So, that is what has in case of supervised learning which makes the use of known patterns for training and learning. Another class of learning or training which is unsupervised

learning. So, what is the difference between supervised learning and unsupervised learning?

In case of supervised learning, we have to make use of a number of known patterns for training purpose or for learning purpose. So, the classifier or the recognizer learns using a set of known patterns. In case of unsupervised learning, I do not have any known pattern. Say, I will have patterns like this. Similar such patterns might also be there. I can have some patterns of this form, similar pattern again of this form. In case of supervised learning you know that these 2 patterns are similar or same though there might be difference or deviation and these 2 patterns are same.

This set of pattern and this set of pattern, they are different. That is known in case of supervised learning. In case of unsupervised learning, it is not known. So, you have a mixture of such patterns. Whenever I have a mixture of such patterns naturally for all the classification problem, as I said that I walk in the feature domain, in the domain of feature vectors not in the patterns themselves. So, I generate set of feature vectors for each of these patterns. So, what I have is a mixture of feature vectors.

The features may be generated from this pattern. It might be generated from this pattern. It might have been generated from this pattern. It might have been generated from this pattern. From where ever it comes, I just have a mixture of feature vectors without having any information of what is the source of that feature vector.

In case of supervised learning, I know the source of the feature vector. So, for unsupervised learning, what I have to do is, I have to process those feature vectors. And by processing those feature vectors, I have to partition the set of feature vectors into number of subsets. Where the feature vectors in 1 set are generated from similar patterns, feature vectors in another subset are generated from another set of similar features.

So, the feature vectors generated from these 2 classes will be going to 1 set. The feature vectors generated from these patterns will move to another set. And this partitioning of feature vectors into different sets has to be done by data processing. This knowledge is not known beforehand. In case of supervised learning, this knowledge is known beforehand. So, once I have.

Once I partition the feature vectors into different subsets, then I can take the feature vectors from every set. Say from set 1, I take all the feature vectors and using those features vectors,

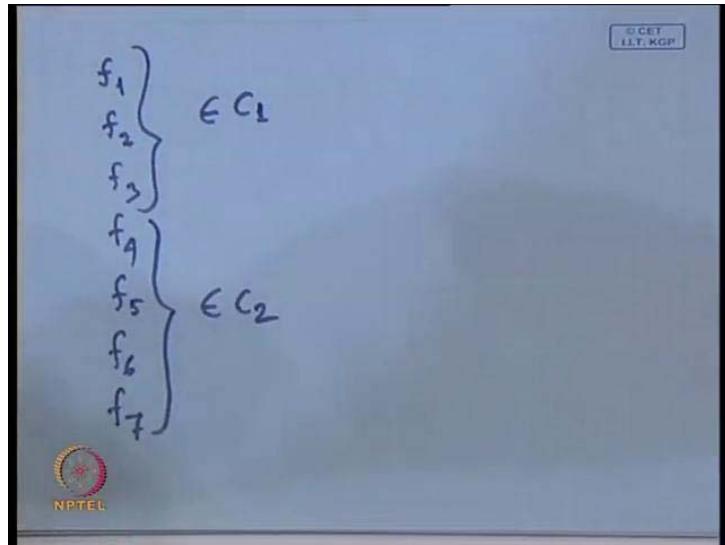
now I generate a representative feature vector, which in case of supervised learning. This is because I knew the class belongingness of the feature vector. Straight way, I could have used those feature vectors to generate the representative feature vector. In this case, the first step which was already known in case of supervised learning that is not known.

So, I have to do it first. That is partitioning the feature vectors into different subsets Once that partitioning is done, I have to take the feature vectors from empty different partition. Using feature vectors belonging to a single partition, I have to generate a representative feature vector for that class of the elements. Then going for recognition, the task in supervised learning, unsupervised learning that remains same. So, only one step will be additional in case of unsupervised learning that I have to go for partitioning of the feature vectors into different partitions depending upon similarity of the feature vectors.

So, when you do this partitioning, you have to make sure that all the feature vectors which I have put into this subset s_1 they are similar, all the feature vectors, which I have put into subset s_2 . They are also similar, whereas if I take a feature vector from s_1 and another feature vector from s_2 . These 2 features of vectors have to be widely different. They should not be similar.

In case of unsupervised learning one step is more, that is I have to do data agglomeration based on similarity, which I do not have to do in case of supervised learning because in case of supervised learning somehow that has been done. That also learns, but what is learnt? What it is learning that is known. See, in case of supervised learning, coming to the feature vector domain, suppose that I have a set of feature vectors $f_1, f_2, f_3, f_4, f_5, f_6, f_7$ and so on.

(Refer Slide Time: 18:27)



Suppose that this f_1 to f_3 , these are the feature of the vectors that belong to say class c_1 and f_4 to f_7 , these feature vectors they belong to class c_2 right? In case of supervised learning, this is already told that these feature vectors f_1 , f_2 and f_3 ; they belong to class c_1 . Similarly, f_4 , f_5 , f_6 , f_7 belong to class c_2 . That is told. So, when I am training my recognizer, recognizer has to make a decision that given a feature vector, what is the class belongingness of that feature vector?

That is the decision that the recognizer has to do, right? So, when the recognizer is taking this decision say given this feature vector f_1 or another feature vector, which is very close to f_1 , which are widely different from any of these 4 features vectors; that feature vector the recognizer should decide that it should belong to class c_1 .

So, what in case of supervised learning you have is? You have the input and for this input, what decision the recognizer has to take that is also known. So, if the recognizer takes any decision other than the correct one, you know there is an error. Is that okay? So, by making use of that error your approach will be to train your recognizer in such a way that the error is minimized, there are different techniques for that. We will discuss that later on.

So, that part, the class belongingness of the training samples or the training feature vectors are known before in case of supervised learning similarly, over here if f_4 is presented to the

recognizer, the recognizer has to take a decision that f_4 belongs to class c_2 . But if the recognizer decides that f_4 belongs to class c_1 that means there is error. So, to minimize this error, I have performed something. So, that is what is learning or training of the recognizer.

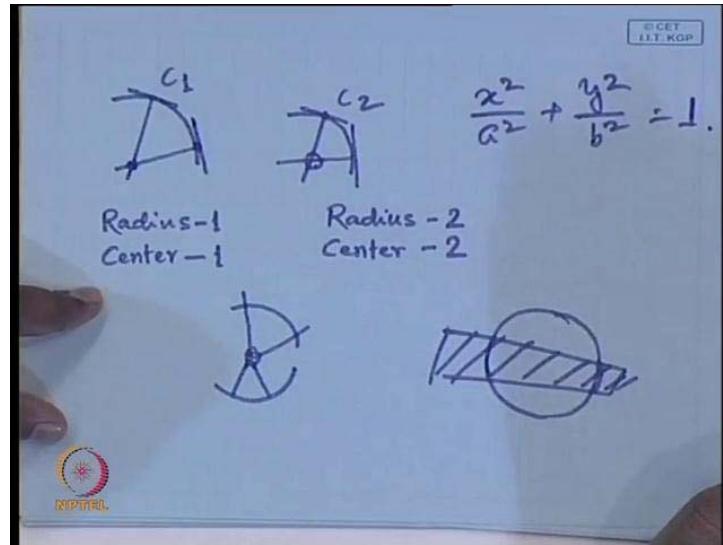
In case of unsupervised learning, I simply have this set of feature vectors. I do not know whether the feature vectors belong to class c_1 or the feature vector belong to class c_2 . So, straight way, I cannot give this f_1 to the recognizer and expect that this recognizer will give me an output, which I can understand. So, that is why, I need a pre partitioning of the features of vectors based on similarity of the feature vectors. You can say this is a sort of preprocessing. Now after doing that, all the feature vectors which are in same partition, I know they are coming from the class. Though I may not know what is that class. In case of supervised learning, I know what that class. In case of unsupervised learning, I may not know what that class is, but definitely I know that they are coming from the same class, whatever class it is. Is it okay?

So, that is how I can form while doing this partitioning. What I am doing is I am making a sort class association of a set of feature vectors. And once this partitioning is done, then by making use of this, I can go for a recognition purpose or training of the recognizer. Following the training, the recognizer can be used for pattern recognition purpose.

So, this is difference between supervised and unsupervised learning. So, in case of unsupervised learning, I have to have this job of partitioning of the feature vectors into different partitions or different subsets. Once that is done, the remaining of part of learning will be same as in case of the supervised learning operation. I am coming to that. So, for determination of this feature vectors, the simple example I have taken is for circular arcs. What feature vector you have to generate or what are the features that you should look into that depends upon your problem.

If I know that all the patterns that I am going to get, they are basically circular arcs, then features I try to generate is simply the radius and the center because given the radius and the center that uniquely determines the circle. I do not need any other information. If it is an ellipse, then radius and circle is not sufficient. Isn't it? I have to know the parameters a and b .

(Refer Slide Time: 23:50)



Because x square by a square plus y square by b square, equation of an ellipse. Is it not? I have to know what are these parameters a and b . So, what are the features or what is the feature vector that you have to generate? That depends upon your problem domain because every feature is not equally acceptable for every pattern. Depending upon the pattern, I have to decide what kind of feature I have to follow.

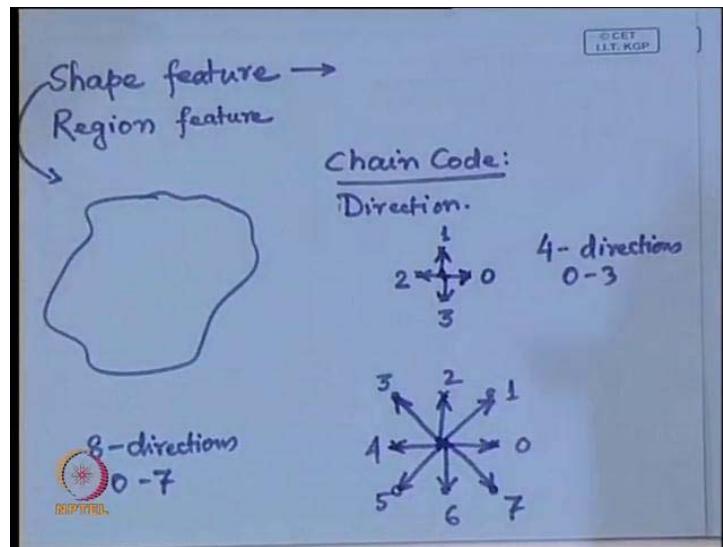
In many cases given a pattern, I can uniquely find out a feature vector. I can uniquely find out a feature. But given a feature, the same pattern may not be generated, I may not be able to generate the same pattern given a feature vector. So, your feature vector to pattern mapping that is actually one to many it is not unique. But given a pattern, if my procedure for generation of feature vector is fixed, I will always generate the same pattern, same feature vector.

So, the mapping from pattern to features that is unique, whereas mapping from feature vector to pattern it is not unique. Is it okay? So because it is not unique. So many patterns may be matched to the same feature vector. Then the question comes that if many patterns are mapped to the same feature vector, then using the feature vector, how do I distinguish the patterns?

This is the only reason that we do not talk about features, but we talk about feature vectors. Feature vectors will definitely have different components. Every individual component tries to capture some property of the pattern. So, all these components taking together in the form of feature vector to some extent pin points to a particular vector though may not be uniquely even in that case. But I can have a smaller set of pattern which will generate the feature vector.

As you reduce the dimension of the feature vector, this set goes on increasing. The size of the set will go on increasing. So, in all these pattern recognition problems, we never talk about a single pattern. But we always talk about feature vector and the feature vector has different components where every component tries to capture a certain property of the feature of the pattern. So, coming to the case of the feature extraction or the discrete feature extraction.

(Refer Slide Time: 26:55)



As we said in the last class that we can have 2 different features; one is shape, feature or shape based feature. The other one is region based feature. So, this shape feature simply tells you what is the shape of the object, whether it is a rectangle, it is a square, it is a circle, it is ellipse and so on or it has some arbitrary boundary. So that is what it is told by the shape, feature. The second kind of feature that is the region feature it tells about what is the region, what is the property of the region, which is enclosed inside the boundary by this

region. So by this region, What I mean is in simple grey level image, it may have intensity values.

If we have color images, it may be a color feature. It may also be texture feature if the region has different types of textures. If it is a colored texture, I can have color feature along with the texture feature. Even a colored texture can have different intensity values. So, when you talked about colorless processing, particularly in HSI model, we have talked about hue, saturation and intensity and we have said that hue gives you the color information. Saturation gives you the purity of the color and intensity is black and white equivalent of the intensity value.

So, if the intensity is high, the image will be bright. If the intensity is low, the image will be dark. So, I can have with in a region the texture feature, color feature, intensity feature, combination of all 3 or any one of them or any 2 taken at a time. So, that is why I said that a single feature does not give you a proper description. What I have to take is multiple numbers of features. Where every feature tries to capture certain property of the pattern and all those features take together in the form of a vector. Whenever I form a vector, the position of the element within the vector is very, very important.

So, I have to put these features in a particular order and all through, I have to maintain the same order, right? So, all these features taken together put in the form of a vector to some extent pin points to a particular pattern or to a class of pattern, where the domain of this class is very, very narrow. As you reduce the dimensionality of the feature vector, the domain will become wider. Because more and more patterns of different types will come into the same class. So, first let us concentrate on this shape feature or shape based feature.

So, suppose that we can have arbitrary shape like this and earlier, we said that whenever we want to detect a shape feature, it is not necessary that I have to know what is there inside. If I know or if I code only the boundary of this shape that is sufficient to tell me what is the shape feature. Now, one of the kind of features or the descriptors, which can be used to describe or represent this particular shape, is what is called a chain code. What is chain code? If I want to represent this boundary, an arbitrary boundary by linear segments that is some sort of linear approximation of, piece wise linear approximation of arbitrary boundaries, so if I want to represent this arbitrary boundary by piece wise linear approximation, then I can represent this boundary by linear segments of some specified length and specified direction.

So, I represent this boundary by sequence of linear segments of specified length and specified direction. That is what is called a chain code, is that okay. So when I say that I have to have linear segments of specified lengths and specified directions, so I have to specify beforehand what the length is and what is the direction?

So, firstly let me talk about how to decide the direction. This direction can be specified in either 4-connectivity or in 8-connectivity. I hope all of you know what is connectivity, 4-connectivity and 8-connectivity. So, this direction can be specified either in terms of 4-connectivity or in terms of 8-connectivity. So, if I talk about 4-connectivity, suppose I have a central pixel somewhere over here. Then these pixels in 4 connectivity has 4 neighbors one on top, one to the right, one at the bottom and one to the left, right?

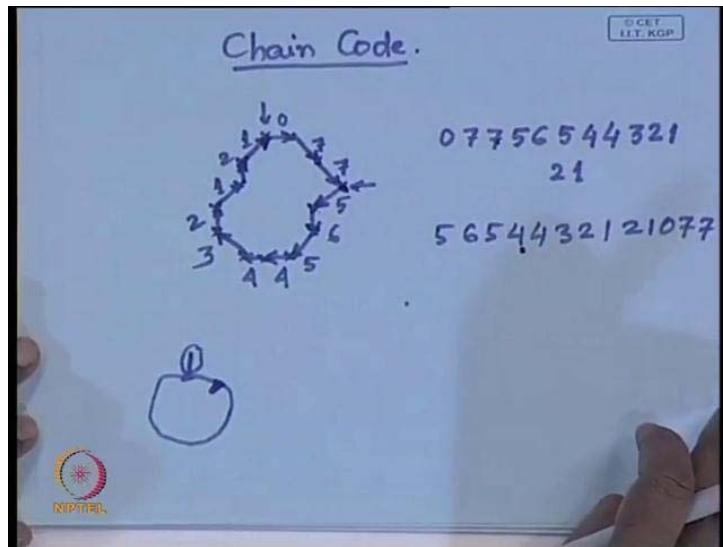
So, when I want to move from this central pixel to any of these 4 connected neighbors, I can have only 4 possible moves. I can move towards east. I can move towards north or I can move towards south or I can move towards east. So, these are the only 4 possible moves that I can make. This is what I mean by direction. So, if I specify that this I code as direction 0, this I is code as direction 1, this west I will code as 2 and south I will code as direction 3.

So, you find that every direction is now coded either as 0 or as 1 or as 2 or as 3. So, that gives me all 4 possible movements, directions in which I can move following 4-connectivity. If I go for 8-connectivity, suppose this is my neighboring pixel. I have these 4 pixels, which are 4 neighbors, one to the East, North, West and South. In addition, I have these diagonal pixels north south, north east, North West, south east and south west direction.

And now, from the center pixel, if I want to move, I can move in any of the 8 directions. I can move either in the north, in the east direction; I can move in the north east direction, I can move in the north direction, I can move in the North West direction or in the west direction or south west direction or south or south east. So, these are the 8 possible moves that I can make from the center pixel following 8-connectivity, right?

So now, if I code these movements as 0, 1, 2, 3, 4, 5, 6, 7, so following 4 connectivity, I have 4 directions, 0 to 3. In case of 8-connectivity, I have 8 directions 0 to 7. Is that okay? So, the directions are fixed. Now, we have to decide about the length of the linear segment. I will come to the length concept a bit later. Now, let me see that how using these directions and assuming length to be 1, how I can go for coding the boundary of a shape, okay? So, suppose that I have a shape something like this.

(Refer Slide Time: 36:11)



So, we are talking about chain code. Suppose I have a boundary something like this. You find that, can you see the grid? Actually, all these are placed on grid intersections. So, suppose that I start from any particular point. Let me assume that I scan the figure in raster scan fashion and whichever is the first boundary point, I reach that my starting point. Now, once I reach my starting point from the starting, I will move in the clockwise direction. I will try to trace the boundary in the clockwise direction. So, my starting point following the raster scan fashion is this one. I want to make a move from the starting point to the next boundary point in clock wise direction.

So, when I try to make a move, obviously it will move in this direction and come to the case that if I assume that I am following this 8-connectivity, this particular movement is given a code 0. So, here I will get 0, next movement over here is in the south east direction and for south east direction, following 8-connectivity the code is given as 7. So, this movement will be coded as 7. Similarly, over here, this is also 7, again south west direction. Sorry this is in the south east direction. Next movement is in the south west direction and south west movement is coded as 5.

So, I will code this as 5. Next one is in the south, which is coded as 6. Again, south west coded as 5. Then in the west, which is coded as 4. One more west which is again coded as 4. Then I have north east, sorry North West which is coded as 3. Then I have movement in the north direction, which is coded as 2. Then I have north east, which is coded as 1. Then again I have movement north coded as 2 and then again on north east, which is coded as 1. And this completes the tracing of the entire boundary. Right?

Now, once I have this, you find that this sequence of movements can be used as a descriptor of this boundary. So, the sequence becomes 0, 7, 7, 5, 6, 5, 4, 4, 3, 2, 1, 2, 1. So, this codes the directions of the linear line segments. And in this case, we have in fact, here all the pixels are more, all the pixels have one neighbor. If it is a boundary, then ideally the pixels will have only 2 numbers. If it is boundary, otherwise the third point that you have is a extraneous point, right? So, I can have situation something like this.

What you said is say, I can have a boundary of this form, so over here, something like this is it ok? So, when I go for chain code, this has to be taken as extraneous point, which has to be removed before hand and for removal of this, you must have heard about morphological filtering. Do just opening and closing operation and these extraneous points will be removed; those who have done my course, I did not talk about the morphological operations.

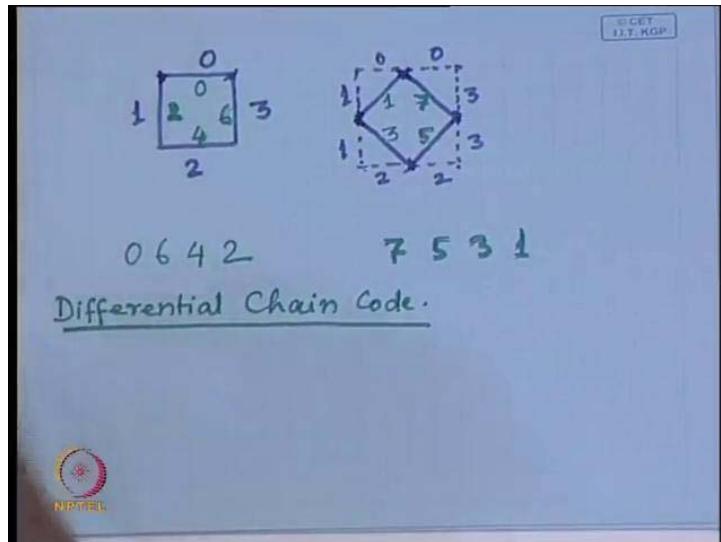
So, by using the morphological filtering operation, such extraneous points can be removed. Similarly, I can have, why something inside, also say something like this, which may come due to noise. So, again by using morphological operations, such kind of noise can be removed. So, coming back to this, boundary is represented by sequence of directions like this and over here the length of the line segment is assumed to be 1 because I have moved from every pixel to its neighboring pixels.

So, this boundary is represented by this kind of chain code. And you find that the problem with this chain code is that, instead of taking this as my starting point, if the starting point comes out to be somewhere over here, then the chain code will be totally different.

The chain code will be 5, 6, 5, 4, 4, 3, 2, 1, 2, 1, 0, 7, 7. If I take this point as my starting point, then this is the chain code. If I take this point as my starting point, then this is the chain code. So, the chain codes are totally different, not only that if the shape is rotated. Then I

have even worst situation, in this case at least even if the chain code is different, but the same symbols appear in the same sequence. But if the shape is rotated.

(Refer Slide Time: 42:48)



Let us take very simple case, say I have a rectangle. And let us assume that length of each side of the rectangle is my unit distance, right? And let me also assume to make the situation even simpler that instead of this 8-connectivity, I am considering 4connectivity. So, following this 4 connectivity and taking this as the starting point, the chain code for this particular figure assuming that every side length is of unit distance, is of unit length will be 0, 3, 2, 1.

Now, if I simply rotate this by 45 degrees, following 4-connectivity, I cannot generate the chain code for this figure Because these movements are not defined in my case or even if I take the concept of chess board movement. So, these are the boundary points. This is one boundary point. So, the kind of movement that I have to make is assuming that this is my starting point. I have to make a movement over here.

This gives me 0. I have to make a movement over here, which gives me 3. I have to make a movement over here, which gives me 3 again, movement over here 2, movement over here 2, movement over here 1, 1, 0. So, you find that here the chain code was 0, 3, 2 1. Now, my

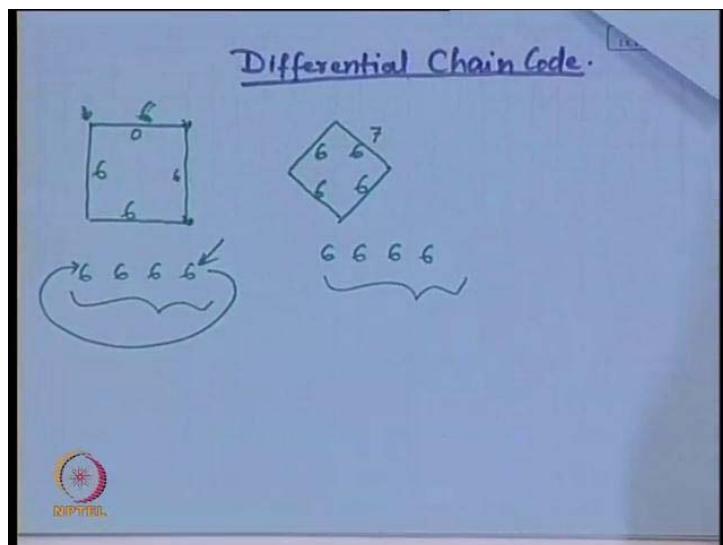
chain code is 0, 3, 3, 2, 2, 1, 1, 0. The chain codes are totally different though over here it is simply a case of scaling. Is it not? Earlier case, I had every line segment only one appearance.

Now, I have every line segment appearing twice. So, you can say that it is just a matter of scaling. I can scale it down to have the same chain code, but what happens to 8-connectivity? If I go for 8-connectivity, let me see if they have different color. If I go for 8-connectivity, then this will be coded as 0. This will be coded as 6. This will be coded as 4. This will be coded as 2, whereas if the same figure is rotated by 45 degrees; this will be coded as 7. This will be coded as 5. This will be coded as 3. This will be coded as 1. Now, it is visible.

So, find that over here. My chain will become 0, 6, 4, 2 and over here, the chain code becomes 7, 5, 3, 1. So following 4-connectivity, though I could say that one is just a scaling version of the other, the moment I go for 8-connectivity because this gives me more precision. The chain codes are totally different. So, when I go for such chain code representation of boundary, you find that if I simply translate this figure move from one position to other position, the chain code will remain the same.

But if I rotate the figure, the chain code becomes totally different. So, this sort of representation, boundary representation is translation invariant because it does not vary with translation. But it is not rotation invariant. It depends upon rotation. Right? Also the problem of selection of starting point. So, over here, as I change the starting point, shape, the starting point, the chain code again becomes different. So, how to take care of these problems? If I want to represent a boundary or a shape by chain code, so there is a concept of differential chain code. So to go for this differential chain code, let us start with the same figure. Let me take another page.

(Refer Slide Time: 48:04)



So, we are talking about differential chain code. So, let us start with the same figure. Now, over here what I do is, when I move from the first point to the second point, I do not give any code to this. Right? When I move from the second point to the third point, so here the code was supposed to 0, right? Here, the code was supposed to be 6 following 8-connectivity. So, instead of giving this direct code, what I give is that how many rotations either in the clockwise direction or in anti-clockwise direction of the rotations in my coding scheme has to be made so that I move from 0 to 6.

So, we come to this coding scheme. Say, this is 0 and this is 6. So, if I follow anti clockwise rotation, I have to move 1, 2, 3, 4, 5, 6, 6 steps. So, I code this as 6. However, to this, I have not given any code. Now, when I move from this to this, this was originally 6 and this is originally 4, right?

So, how many such rotational steps I have to perform? So, this is 6. This is 4. So, number of steps will be 1, 2, 3, 4, 5, and 6. 6 again, so this also becomes 6, right? From here to here, I have initially 4 and now, it has to be 2, right? Initially, it was 4. Now, it has to be 2. So, again how many steps I have to move 1, 2, 3, 4, 5, 6. So, again 6 steps, right?

And this was 2, and this has to be 0. So, 2 to 0, how many 1, 2, 3, 4, 5, 6, again 6 steps. So, this 6, 6, 6, and 6 that becomes one differential chain code. What happens in this particular case? Here, my initial direction was 7, but I do not code it. The direction of movement was 7 same as this.

What is the next direction? Next direction is 5. You come over here. Next direction is 5. So, from 7 to 5, how many rotations I have to perform? 1, 2, 3, 4, 5, 6, so I have to perform 6 rotations. I did that, right? Now, this was 5 and this is 3, 5, and 3. How many rotations I have to perform? 1, 2, 3, 4, 5, 6 again. So, this also becomes 6. So, likewise, if you continue, you will find that all these directions in the differential code will be 6, 6 only.

So, this becomes 6, 6, 6, 6. This also becomes 6, 6, 6, 6. The codes are identical. Now, it has become very simple only 6, 6, 6, 6 simply because my figure is simple. If the figure is a complicated figure or an arbitrary figure, then my code will not be as simple as that. So, I will get the differential chain code, which is rotation invariant. But even then the differential

chain code does not remain starting point invariant depending upon the starting point, I will have to find chain codes for an arbitrary figure.

So, to take care of that, what you can do is, I can consider this code as a cycle. Instead of taking this as a chain, I can take it as a cycle. Once I have a cycle, I can decide where to cut that cycle, so that I can have a numerical representation, which is either maximum or minimum. That becomes my chain code because now instead of chain, initially I considered that to be a cycle. So, it will be a starting point invariant because cycle does not have any starting point or any end point. Then I enforce a starting point or end point depending upon a position.

So, if I cut the cycle at that point, the numerical representation of the code that I will get would be either maximum or minimum because it is maximum or minimum. Given a cycle, I can find out only 1 maximum or 1 minimum; not multiple unlike I have a situation like this. I mean wherever I cut this cycle, I get the same number. So, it is having all maxima or all minima. That is only because this is symmetric and simple. In case of arbitrary figure, I will not have that kind of unfortunate situations.

So, symmetry, though we like it in our daily life most likely, but it makes our life fail whenever for such auto machine job. This example there is no problem because this is a very simple one. But, when you work with the real life problems, you do not get such simple situations. You will have arbitrary figures. So, we will stop here today. Next day, we will continue with other features and feature description techniques.

Thank you.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 20
Perceptron Criterion

Good morning, so we are discussing about pattern recognition problem, and we are talking about the different discriminant functions, which are helpful for classification of unknown patterns. So, earlier what we have seen is starting from the Bayes decision theory that if you know the probability density function of the parametric form of the probability density function. We can design the different discrimination functions for pattern classification or classification of pattern vectors or feature vectors. We have discussed about specific case that, if the probability density function is either Gaussian or Normal distribution function.

And we know the mean and co-variance matrices of the different classes, then we can have different types of discriminant functions or different types of decision boundaries among different classes. In particular, if the co-variance matrices of all the classes are same then we have seen that discriminant function that we get is a linear discriminant function. Whereas, if different classes of the samples belonging to different classes they have different co-variance matrices, arbitrary co-variance matrices, in that case, in general the kind of discriminant function that we get is a quadratic discriminant function.

So, in multi dimension it is called hyper quadrics. And accordingly the decision between two different classes in one case is hyper plane. And if the co-variance matrices are arbitrary or different classes have different co-variance matrices, then the decision surface between two classes becomes a hyper quadric surface. However, following the base decision theory if we want to have the discriminant function, we have to know that, what is the parametric form of the probability density function of the samples belonging to that particular class.

And given the number of samples if we know the parametric form of the probability density function, we may not know the parameter values exactly. If we know the parametric form of the probability density function, then we have seen in previous classes that we can have the maximum likely hood estimate of those parameter values. So this maximum likely hood estimation says that these are the best representations or best supported by the samples, which are available from that particular class.

And then also we have discussed about the dimensionality problem because not every feature has equal discriminating power. So, following the principle component analysis or following the multiple discriminant analysis, we can reduce the dimensionality of the feature vectors ensuring that the features in the reduced dimensional feature vector will have better discriminating power than the features which are avoided. Now, today we will discuss about this same linear discriminant function, but unlike in case of what we have obtained from Bayes decision theory that we have to know the parametric form of the probability density function.

In case the parametric form of probability density function is not known. So, then what you have to do is we have to design the discriminant function or we have to find out the decision boundary between two different classes by using the samples which are available from different classes. So, here we do not assume any probability density functions. So, we know, so initially we will discuss about the linearly separable classes. So, we know that the classes are linearly separable and because we are discussing about the supervised learning process.

So, we know a set of samples belonging to one class ω_1 , we know set of samples belonging to another class ω_2 . So, using this information and without using any assumption of the nature of the probability density function. We have to find out the linear discriminant function which discriminates between these two different classes. So, we will assume that the classes are linearly separable. And as the classes are linearly separable, so we should be able to formulate we should be able to design linear discriminant function. Where no probability density function form is assumed so because the discriminant functions will be linear.

(Refer Slide Time: 05:24)

Perceptron Criterion

$g(x) = \mathbf{w}^T \mathbf{x} + w_0$ $x \rightarrow d\text{-dimensional}$

\downarrow Weight vector. \downarrow bias / threshold weight.

$\begin{cases} g(x) > 0 \rightarrow x \in \omega_1 \\ < 0 \rightarrow x \in \omega_2 \\ = 0 \rightarrow x \text{ on the decision boundary.} \end{cases}$

Nature of \mathbf{w} ?

x_1, x_2 on decision boundary.

$$\mathbf{w}^T \mathbf{x}_1 + w_0 = \mathbf{w}^T \mathbf{x}_2 + w_0$$

$$\Rightarrow \mathbf{w}^T (\mathbf{x}_1 - \mathbf{x}_2) = 0$$

So, you know that a linear equation can always be written as $g(X) = W^t X + W_o$, where X is the feature vector. And we assume that the feature vector X is of dimension d . So, it is d dimensional feature vector. And W is called weight vector and W_o is called the bias or threshold weight. So, we see that this particular expression is a linear expression, where X is a d dimensional vector, W is also a d dimensional vector. And this term $W^t X$ is nothing but inner product of vector W with a vector X . And our decision criteria will be therefore, an unknown feature X , if $g(X) > 0$. Then we classify X to belong to class ω_1 , if it is less than 0 then X will be classified to class ω_2 .

And if $g(X) = 0$, then we say X is on the decision boundary. So, if $g(X) > 0$ we decide X belongs to class ω_1 , if it is less than 0 we decide X belongs to class ω_2 , if $g(X) = 0$ then X is on the decision boundary. So, the decision boundary between two classes X , ω_1 and ω_2 is given by the equation $g(X) = 0$ or $W^t X + W_o = 0$. Now, let us see that what is the nature of this weight vector W ?

So, to find out the nature of the weight vector W let us take two points on the decision boundary. So, what we are interested in the nature of weight vector W . So, we take two points X_1 and X_2 on decision boundary. Now, because X_1 and X_2 are on the decision boundary so $W^t X_1 + W_o = W^t X_2 + W_o$. And both of them equal to 0 because X_1 and X_2 are lying on the decision boundary.

So, we can simply write that $W^t X_1 + W_o = W^t X_2 + W_o$. So, from this you find that $W^t(X_1 - X_2) = 0$. Now, what is this $(X_1 - X_2)$, $(X_1 - X_2)$ is nothing but, a vector lying on the decision surface, because X_1 is on the decision surface, that is a vector X_2 is also a vector that is lying on the decision surface $(X_1 - X_2)$ is a vector which is lying on the decision surface.

And $W^t(X_1 - X_2)$ is the inner product of W with the vector $(X_1 - X_2)$, and because this inner product is equal to 0. So, that clearly indicates that this vector W is orthogonal to any vector lying on the decision surface. Because we have taken X_1 and X_2 arbitrarily so $W^t(X_1 - X_2) = 0$. That indicates that vector W is orthogonal to any vector lying on the decision surface that means the vector W is orthogonal to the decision surface. And this surface being a planar surface or a linear surface in d dimensional space we call it a hyper plane.

(Refer Slide Time: 10:51)

Perception Criterion

$g(x) = \mathbf{w}^t \mathbf{x} + w_0$ $\mathbf{x} \rightarrow d\text{-dimensional}$
 Weight vector. bias / threshold weight.

$\begin{cases} g(\mathbf{x}) > 0 \rightarrow \mathbf{x} \in \omega_1 \\ < 0 \rightarrow \mathbf{x} \in \omega_2 \\ = 0 \rightarrow \mathbf{x} \text{ on the decision boundary.} \end{cases}$

Nature of \mathbf{w} ?

x_1, x_2 on decision boundary.
 $\mathbf{w}^t \mathbf{x}_1 + w_0 = \mathbf{w}^t \mathbf{x}_2 + w_0$
 $\Rightarrow \mathbf{w}^t (\mathbf{x}_1 - \mathbf{x}_2) = 0$ $\langle H \rightarrow \text{hyperplane} \rangle$

And let us represent that hyper plane by H is the hyper plane. So, this vector \mathbf{W} is orthogonal to hyper plane H . Now, what does this $g(X)$ actually represent?

(Refer Slide Time: 11:21)

$g(x) \rightarrow$ What does it represent?
 ↓ algebraic measure of \mathbf{x} from H .

How?

$$r = \frac{g(x)}{\|w\|}$$

$$\begin{aligned} x &= x_p + r \cdot \frac{w}{\|w\|} \\ g(x) &= w^t x + w_0 \\ &= w^t x_p + w_0 \\ &\quad + r \cdot \frac{w^t w}{\|w\|} \\ &= r \cdot \|w\| \end{aligned}$$

You find that $g(X)$ gives you some measure of the algebraic distance of vector X from the decision surface or from the hyperplane H . So, this $g(X)$ this gives an algebraic measure of X from H . Let us see how? How do you say that $g(X)$ represents an algebraic measure? It is not the exact distance value, but it is measure of the distance value and is something proportional to the distance algebraic distance of the vector X from the decision surface,

from the hyper plane H. How does it give us that measure? Let us consider a case in 3 dimensions.

Say I have a 3 dimensional space with the vector components represented by X_1, X_2 and X_3 . Suppose, I have a decision surface say something like this, in three dimensional planes. So, this is my plane H and let me take a point X somewhere over here. And when I take this point X over here, let me drop a perpendicular from this point X onto the hyper plane. And let me assume that X_p represents the foot of the perpendicular on this hyper plane. And suppose the distance between X_p and X, the length of this vector joining X_p and X is given by say r.

Now, if it so in that case I can write $X = X_p + r \cdot \frac{W}{\|W\|}$, because we have seen earlier that the vector W is orthogonal to the hyper plane, orthogonal to the decision surface. So, the direction of the W is in the same direction of X_p to X, because X_p to X this vector is orthogonal to this hyper plane. W is also orthogonal to the hyper plane. So, this is W. So, I can write this X is equal to $X = X_p + r \cdot \frac{W}{\|W\|}$ because $\frac{W}{\|W\|}$ is the unit vector in the direction perpendicular to the hyper plane.

So, I can write it in this form. And when I write if like this then what is $g(X)$, $g(X) = W^t X + W_o$ which in this case will be $g(X) = W^t X_p + W_o + r \cdot \frac{W^t W}{\|W\|}$. I am simply replacing this X by this $X_p + r \cdot \frac{W}{\|W\|}$. Now, out of this, the point X_p which is of the perpendicular from X onto the hyper plane that lies on the decision surface. So, $W^t X_p + W_o$ because X_p is on the hyper surface $W^t X_p + W_o = 0$ because it is on the decision surface.

So, what I get is this will be simply $g(X) = r \cdot \frac{W^t W}{\|W\|}$ and this $W^t W$ is nothing but $\|W\|^2$. So what is simply get is $g(X) = r \cdot \|W\|$. So, you find that this discriminant function $g(X)$, the value that you get is nothing but r, we said is the distance, perpendicular distance or orthogonal distance of point X from the decision surface, because it is the distance between X_p and X. So this r is the orthogonal distance of point X from the decision surface.

So, this $g(X)$ is the orthogonal distance scaled up by $\|W\|$. And if I want to find out what is

the actual orthogonal distance that is given by $r = \frac{g(X)}{\|W\|}$. So, that is why we said that this

discriminate function $g(X)$ actually gives you a measure, algebraic measure of the distance of the point orthogonal distance of vector X from the decision surface. Now, why are we calling it algebraic distance?

You come to our original expression $g(X) = W^t X + W_o$. This is your discriminate function. What we said is if $g(X) > 0$ then X belongs to ω_1 . If $g(X) < 0$ then X belongs to ω_2 . If it is equal to 0 then X is on the decision boundary. That is $g(X) = 0$ not $W^t X = 0$.

And $g(X) = W^t X + W_o$. So, because X_p is lying on the decision surface so $g(X_p) = 0$.

And $g(X_p) = W^t X_p + W_o$, is that okay? So, here why we are calling it as algebraic distance is that if r is positive then X lies on the positive side of the hyper plane.

If r is negative, then X is on the negative side of the hyper plane. And this expression that I have said that $r = \frac{g(X)}{\|W\|}$. This is an expression which is nothing new, I mean this is what you have already done in your school level mathematics.

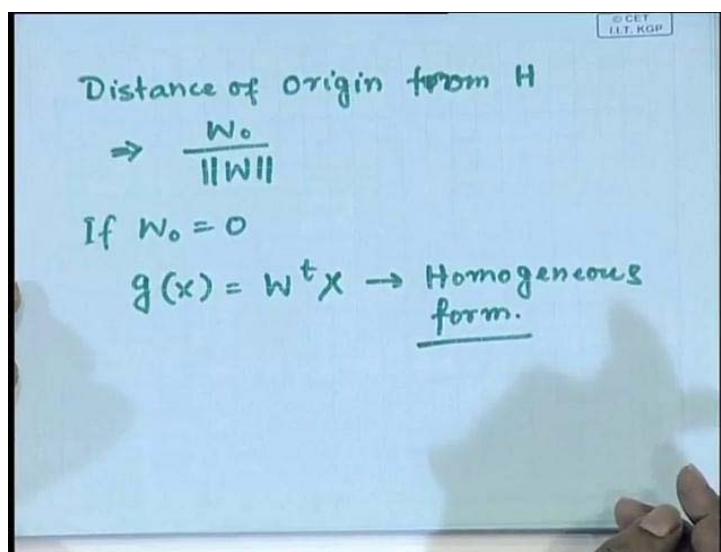
(Refer Slide Time: 20:31)

A photograph of a handwritten derivation on a light blue background. At the top, the equation $ax + by + c = 0$ is written, followed by the point (x_1, y_1) . Below these, the formula $\frac{ax_1 + by_1 + c}{\sqrt{a^2 + b^2}}$ is derived, showing the steps of substituting the point into the equation and dividing by the norm of the vector $\begin{pmatrix} a \\ b \end{pmatrix}$.

Because we know that in school level mathematics what you have done is, if I have the equation of a straight line given say $ax + by + c = 0$. Come to your co-ordinate geometry. And given a point (x_1, y_1) , what is the distance of point (x_1, y_1) from this straight line which is nothing but $\frac{ax_1 + by_1 + c}{\sqrt{a^2+b^2}}$. And if you compare this expression with this expression they are exactly identical. This expression is in the 2 dimensional, this expression is in d dimensional, where the dimensionality has increased.

So, what you have derived over here is nothing new, this is something that you have already done in your school level mathematics. So, that we are saying is that if r is positive then the vector X lies on the positive side of the hyper plane. If r is negative, then X lies on the negative side of the hyper plane. And what is the distance of origin from the hyper plane?

(Refer Slide Time: 22:03)



So, distance of origin from the hyper plane that is simply given by $\frac{W_o}{\|W\|}$. So here if W_o is positive then origin lies on the positive side of the hyper plane, if W_o is negative then origin lies on the negative side of the hyper plane. And if $W_o = 0$ then the hyper plane passes through the origin. So, if $W_o = 0$ then the hyper plane passes through the origin.

And not only that your discriminant functions take a particular form that $g(X) = W^t X$, where I do not have any bias or threshold weight. And this is the form which is called homogeneous form. And in mathematics it is always convenient that if I can represent an

expression in a homogeneous form that avoids many of the problems that we may face while we try to analyze different given problems. So, after we discuss about the nature of the decision surface if it is hyper plane then what are the conditions or what are the different positions of the hyper plane.

Then we can have then the hyper plane has two sides. Actually, hyper place divides your feature space d dimensional space into two half spaces. One of the half spaces is positive the other half space is negative. If a feature vector falls on the positive half space, then it will be classified into one task. If the feature vector falls on the negative half surface it will be classified into another class. If it is on the decision surface then we cannot classify it actually, because it is equally likely that it may belong to class ω_1 or it may belong to class ω_2 .

So, if the sample falls on the decision surface or falls on the hyper plane, we cannot really take any decision, it is an ambiguous case. Now, after discussing this let us see that how we can design this vector W , ultimately because it is a linear class. I have to find out the discriminate function $g(X)$ and if I can find out the weight vector W and the bias of the threshold weight W_0 then my linear classifier is designed.

So, I have to design or I have to decide about W and W_0 . Based on the samples that are available because it is supervised learning so we are given samples from both the classes ω_1 and ω_2 . So, I have to decide W and W_0 based on the information available from those known labeled samples.

(Refer Slide Time: 25:46)

© CET
IIT, KGP

Design of Weight Vector W

→ Two category linearly separable case.

$$g(x) = W^t x + W_0$$

$$\approx a^t y$$

$$y = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix} \quad \left| \quad \begin{bmatrix} W_1 & W_2 & \dots & W_d & W_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix} \right. \\ = \sum_{i=1}^d w_i x_i + w_0 \\ = W^t x + W_0$$

So, let us talk about design of the weight vector W . And here again for simplicity initially, we will assume that we have two categories linearly separable case. Later on we will generalize this to multiple category problems where we have c number classes, but to start with let us start with on the two category case that is we have classes ω_1 and ω_2 . So, what we have so far is, we have, we know that we have to have a discriminate function of the form $g(X) = W^t X + W_o$.

And as we said that this expression of the discriminate function this is not in the homogeneous form. However, if I can convert it into homogeneous form then my analysis will be easier or my design will be easier. So, how we can convert this into homogeneous form I can very easily convert this into homogeneous form. If I write this expression in the form $g(X) \approx a^t y$ where, a is my modified weight vector. And this modified weight vector will include all the components of the weight vector W and also the bias weight or the threshold weight W_o .

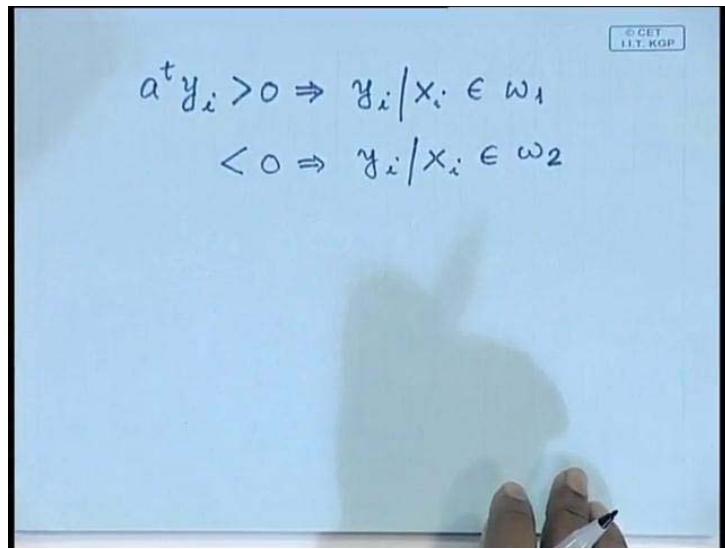
So, if this weight vector, a has to include all the components of W as well as W_o then I have to do some augmentation in the feature vector X . So, what I will do is, I will write this y is actually augmented feature vector X . So, what will be y will be nothing but, I have to take all the components of X . So, I have to take the first component x_1 , second component x_2 , all the d components of x that is up to x_d . And I make last component is equal to 1. So, if I do that for every X if I augment one to give me an augmented feature vector y .

Then I can write this expression in a homogeneous form as $a^t y$. Simply, because what is this

$$a^t y ? a^t y \text{ is nothing but } a^t y = [w_1 \ w_2 \ \dots \ w_d \ w_o] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix}. \text{ So, if you do this}$$

multiplication, this multiplication is nothing but, $\sum_{i=1}^d w_i x_i + w_o$. So, I get exactly the same expression as this. Only thing I have to do is I have to augment 1 to the feature vector X giving me the augmented feature vector y . And after having this homogeneous expression my decision rule remains the same.

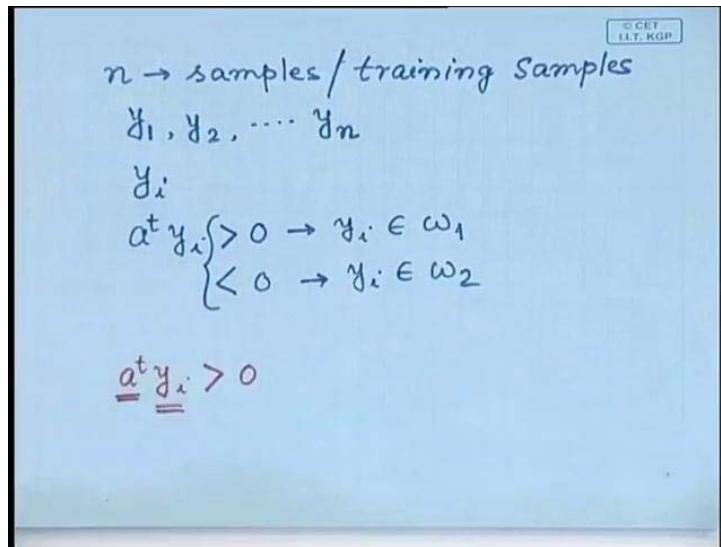
(Refer Slide Time: 31:04)



That if $a^T y_i$, for the i^{th} augmented feature vector X_i is greater than 0. Then I decide that y_i of the corresponding vector X_i that belongs to class ω_1 . And if it is less than 0 then I decide that y_i of the corresponding feature vector X_i , do not confuse between this subscript and this subscript here I have meant this subscript means i^{th} feature vector. So, if it is less than 0 then my decision will be this belongs to class ω_2 , is that okay?

So, my decision rule remains the same. Only thing is I have a modified formulation of the same problem. Now, let us see that how we can really design this weight vector W or in this case the weight vector a because a consists of all the components W and also the bias vector or the threshold vector W_0 . So, our aim is to design the weight vector a by using the knowledge of the labelled samples from class ω_1 as well as class ω_2 . So, we assume that we have n number of samples and when I say n number of samples that means we have n number of augmented samples where I have appended 1 to each of the feature vector.

(Refer Slide Time: 33:07)



So, I have n number of samples called training samples, because these are the samples which are used to train the classifier. So, this n number of samples y_1, y_2 up to y_n . So, each of this y_i 's are actually augmented feature vectors, by appending one to the original vector X . Some of these samples, some of these feature vectors are labeled as class ω_1 . Some of these feature vectors are labeled as class ω_2 . Now, for a particular feature vector augmented feature vector y_i as I say the earlier my decision rule will be $a^T y_i > 0$ indicates y_i is in class ω_1 , and less than 0 indicates y_i is in class ω_2 .

If it is equal to 0, I cannot take any decision, right? So if I, what I have to do is, I have to find out the value of this weight vector a . I have some samples which are labeled as class ω_1 . I have some samples which are labeled to belong to class ω_2 . Now, given a weight vector a , if I take all the samples which are labeled as ω_1 , If I find that for each of those samples $a^T y_i > 0$, that means that given weight vector a is correctly classifying all the samples, all the feature vectors which are labeled as class ω_1 .

If I also find that for the same weight vector a , all the samples belonging to class ω_2 , which are labeled as belonging to class belonging to class ω_2 . For all of them $a^T y_i < 0$ then the weight vector a is also classifying the samples which are labeled as belonging to class ω_2 . So, that particular a which we have obtained, that is the correct weight vector because it is classifying correctly all the samples which are in class ω_1 . It is also classifying correctly all the samples which are labeled as ω_2 .

So, that is a, weight vector that we want, but how to obtain such a weight vector. So, to obtain such a weight vector another modification that can be done is that now, you see that I have two decision criteria. In one case I am checking that it has to be a transpose $a^T y_i > 0$. In another case I am checking that a transpose $a^T y_i < 0$. So, instead of having these two different criteria cannot we have single criteria, that if some condition is true, a single condition is true I say that the sample is classified correctly.

If the condition that is not true, single condition if that is not true I decide that the sample is not classified category. So, cannot I make something like this that, if a transpose $a^T y_i > 0$, I will say sample is correctly classified irrespective of whether this feature vector y_i is labeled as belonging to class ω_2 or it is labeled as belonging to class ω_2 . So, I do not look at the label of feature vector y , whatever be its label if I can somehow convert this decision in the form that $a^T y_i$ irrespective of label of y_i . If $a^T y_i > 0$ I will say that y_i is correctly classified otherwise I will say that it is miss classified.

So, otherwise includes less than 0 as well as equal to 0 because equal to 0 does not give me any information. I cannot decide about the class belongingness of the sample. So, whether it is less than 0 or equal to 0, I will say it is miss classified. If it is greater than 0, I will say it is correctly classified. So, how we can do that? From here you said that if $a^T y_i > 0$ then y_i belongs to class ω_1 . If $a^T y_i < 0$ then y_i belongs to class ω_2 .

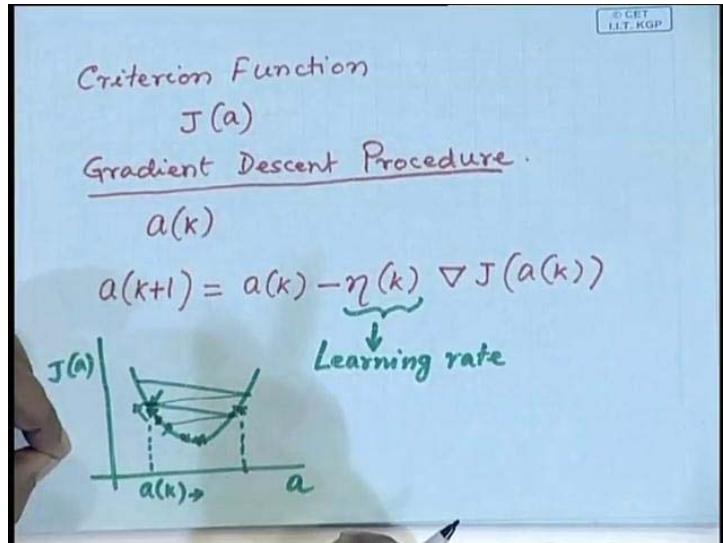
So why do not I do one thing I take all the samples which are labeled as ω_2 and then negate it. So instead of considering y_i , I consider $-y_i$, because I know all the training samples which are given from class ω_2 . So, I simply negate all those training samples. So, when I try to find out this weight vector a for the samples belonging to class ω_1 , I will take them as it is, as it is means after augmentation after appending one.

And for the samples belonging to class ω_2 , I will augment them by appending one and then take negative of it. So, if I take the negative then this $a^T y_i$ which is supposed to be less than 0 now, it will be greater than 0. So, I get uniform decision criteria after negating all the samples belonging to class ω_2 that in all the cases I should have $a^T y_i > 0$.

So, I will say that $a^T y_i > 0$ my sample is correctly classified irrespective of whether the sample is taken from class ω_1 or the sample is taken from class ω_2 is that okay? So, what I will do is I will take all the samples from class ω_1 as it is. The samples from class ω_2 , I will

take negative of it. And then use this for designing or to find out what will be the weight vector a .

(Refer Slide Time: 40:53)



Then again it is better if we can define some sort of criteria function. So, we take some sort of criterion function say $J(a)$ and the $J(a)$ has to be minimized, if a is the correct weight vector that means for that weight vector a which classifies all the training samples correctly, $J(a)$ will be minimum. So, I will define some sort of criterion function we will see what kind of criterion function we can have. And then my job will be, by varying a in an iterative way I have to minimize this $J(a)$.

And this $J(a)$ has to take a minimum value when I reach a weight vector which properly classifies all the training samples. And for minimization of this $J(a)$, the criteria function $J(a)$ we can make use of an approach which is called gradient descent approach or gradient descent procedure. What is the gradient descent procedure? Suppose at the k^{th} iteration I know what is $a(k)$, $a(k)$ is the value of the vector a , weight vector a in the k^{th} iteration. I have to update this weight vector a . So, from k^{th} iteration I have to go to $k + 1$ iteration.

So, from here when I want to find out the value of $a(k+1)$, which will be $a(k+1) = a(k) - \eta(k) \nabla J(a(k))$. This is what gradient descent procedure. And what is this $\eta(k)$? $\eta(k)$ actually specifies the learning rate. So, what does this procedure mean? Suppose, I have a criteria function something like this. This is the point where it is minimum. Suppose, in the k^{th} instant whatever the value of $a(k)$, I have the criteria function is somewhere over here.

So, I put that this criteria function this is $J(a)$ with respect to a . So, at the k^{th} iteration suppose, this is my $a(k)$ and the criteria function value of the criteria function is $J(a(k))$. If I take the gradient of this, gradient will increase in this particular direction. So, what I am doing is, I am moving $a(k)$ in a direction which is negative to the gradient. That means I am moving $a(k)$ in this particular direction.

So, in $k+1$ iteration depending upon the value of this $\eta(k)$ which is learning rate I may move either over here or over here or over here or even somewhere over here. So, if the value of $\eta(k)$ which is called learning rate, if the $\eta(k)$ is very large then when I modify this $a(k)$, $a(k+1)$ may be found somewhere over here. That means I am over shooting the minimum value, where if $\eta(k)$ is very small I may land somewhere over here. So, which says that you are landing rate is very slow.

If $\eta(k)$ is very large I may over shoot the solution and this may lead to oscillation. So, what I will have is from over here I will move to this point then to this point then to this point then to this point and so on. So, I will have oscillation around the solution vector, is that okay? So, this is what this gradient procedure does. That you start with an initial arbitrary value of the weight vector, for that weight vector you find out what is the criteria value of the criteria function. Then try to minimize the criteria function following the gradient descent procedure or this is also called procedure of steepest descent. That means you follow the negative of the maximum gradient value, path of the negative of the maximum gradient value. So, this is also called steepest descent procedure. Is that okay?

(Refer Slide Time: 46:55)

© CET
I.I.T. KGP

Perceptron Criterion

$$J_p(a) = \sum_{\forall y \text{ misclassified}} (-a^T y)$$

$$\nabla_a J_p(a) = \sum_{\forall y \text{ misclassified}} (-y)$$

$a(0) \rightarrow \text{arbitrary}$

$$a(k+1) = a(k) + \eta(k) \sum_{\forall y \text{ misclassified}} y$$

Now, one such criteria function as I said that, we will talk about the Perceptron criterion function, one such criterion function is called Perceptron criterion. Now what is this Perceptron criterion? Let us assume that at the k^{th} instant, somehow we have been able to obtain the weight vector $a(k)$. I am not saying how we have obtained it? Somehow in the k^{th} iteration I am about to find out the weight vector $a(k)$, because it is in the k^{th} iteration I say it is $a(k)$.

Now, this $a(k)$, I will try whether it can correctly classify all the training vectors or not, all the training samples or not. And when I consider all this training samples, as I said that the training samples from class ω_2 , they will be negated, after augmentation they will be negated, training samples belonging to class ω_1 they will simply be augmented, they will not be negated so that if all the samples are correctly classified for all the samples I must have $a_k^t y > 0$.

If there is any sample for which $a_k^t y < 0$ or it may be equal to 0, I say that particular training sample is not correctly classified. So, our aim will be to find out a weight vector a which will classify all the training samples correctly. I do not want to have even a single training sample which is not correctly classified or which is miss classified, right? So, I can try to design the criteria function which will make use of the training samples which are not correctly classified.

Because if the training sample is correctly classified, I do not bother about it because my job is done. If the sample is not correctly classified, then I have to think to modify my weight vector. So, the criteria function that I want to design; I will make use of the training samples which are not correctly classified by that particular weight vector $a(k)$. So, accordingly I can define the criterion function, I call it $J_p(a)$ because we are saying it Perceptron criterion, which is equal to $\sum_{y \text{ missclassified}} (-a^t y)$, is that okay?

So, you find that if y is misclassified then $a^t y_i$ it is negative. So, $-a^t y_i$ has to be positive, right? So, as a result this Perceptron criterion function $J_p(a)$, it can never have a negative value. It will always have a positive value, and the minimum value can be 0. So, I have a global minimum for this criterion, Perceptron criterion function. And this global minimum I can find out by gradient descent procedure, is that okay? So, if I want to apply that gradient descent procedure then I have to take the gradient of $J_p(a)$ with respect to this weight vector a . Is that okay?

So, I will have $\nabla_a J_p(a)$ which is nothing but $\sum_{\text{y missclassified}} (-y)$. Is that okay? And my update

rule or the algorithm to find out this weight vector a will simply be, I take an initial weight vector $a(0)$ arbitrarily. So, $a(0)$ will be arbitrary then $a(k+1)$ that is the weight vector in the $k+1$ iteration will be simply $a(k) + \eta(k)$, which is the learning rate into $\sum_{\text{y missclassified}} (-y)$.

So, this is the algorithm for design of my weight vector. So, once I have chosen $a(0)$ arbitrarily. From $a(0)$ I can find out $a(1)$ which is nothing but $a(0) + \eta(0) \sum_{\text{y missclassified}} (-y)$. So,

I have $a(1)$, from $a(1)$, I can find out $a(2)$, $a(2)$ is nothing but $a(1) + \eta(1) \sum_{\text{y missclassified}} (-y)$.

So, by doing this iteratively I can finally, find out a weight vector which will finally, classify properly all the training samples. May be the number of iterations will be more depending upon your initial choice and the number of feature vectors that you have the distribution of feature vectors, but if the feature vectors are linearly separable will get an weight vector which will properly classify all the training samples. So, we will stop her today next day will continue with this lecture.

Pattern recognition and applications
Prof. P.K. Biswas
Department of Electronic and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 21
Perceptron Criterion (Contd.)

Good morning. So, in the last class we started discussion on design of the linear discriminant function in cases when we do not know what is the probability density function of the nature of the probability function of the state of training samples that are given. That is in the cases when I do not know the nature of the data distribution then I cannot have any parametric form of the probability density functions. So, even in such cases how we can design the linear classifiers.

So, we have started discussion based on that because it is a linear classifier. So, in d dimensional space the classifier will be actually hyper plane. And we can have a wide vector, but the weight vector is actually orthogonal to hyper plane. So, if we can find out what is the weight vector then basically our classifier is defined particularly when I have two class problems that is I have to differentiate between two classes say ω_1 and ω_2 . So, the problem that we have tried to address was something like this.

(Refer Slide Time: 01:31)

Perceptron Criterion
(Contd.)

$$\{X\} \rightarrow \{Y\}$$
$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$
$$y = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix} \Rightarrow d+1 \approx d$$

That given a set of the vector say X , in that case actually what we have is, if I transform this feature vector X into another vector Y , the set of vectors Y , where this Y is obtained by appending one more dimension which is constant equal to 1 to the components of vector X . So, if X is of the form $[x_1 \ x_2 \ \dots \ x_d]^T$ where X is a d dimensional vector.

Then the vector Y will be of the form $[x_1 \ x_2 \ \dots \ x_d \ 1]^T$. So, effectively this vector Y actually becomes $d + 1$ dimensional vector. And let me put it as so we are considering vectors of dimension \hat{d} , which is nothing but $d+1$, where d is the dimensionality of the original set of vectors. So, given this type of situation we have seen that our classifier.

(Refer Slide Time: 02:58)

The image shows handwritten mathematical notes on a whiteboard:

- $a^T y > 0 \rightarrow y \in \omega_1$
- $< 0 \rightarrow y \in \omega_2$
- $J(a)$
- $\begin{cases} a(0) \text{ arbitrary} \\ a(k+1) = a(k) - \eta \nabla_a J(a) \end{cases}$
- $J_p(a) = \sum_{y \text{ misclassified}} -a^T y$

Now, is of this form that if $a^T y > 0$, we decide that y is from class ω_1 . If it is less than 0, then we decide y belongs to class ω_2 . And for design purpose what we have done is, for all the training samples which are given from class ω_2 we have negated them. So, that I have a uniform criterion then that if the weight vector a properly classifies a sample I always have the condition that $a^T y > 0$.

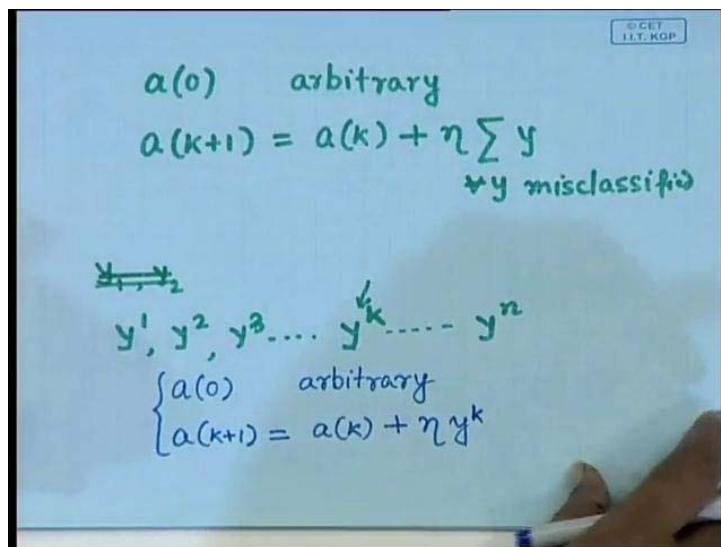
The moment $a^T y < 0$, I know irrespective of the class form whilst that y has been taken the moment I get $a^T y < 0$ I know that y has been misclassified by the weight vector a . So, I have uniform criteria after this negation operation that whenever a transpose $a^T y < 0$ means y is properly classified for any of the vectors y .

If $a^T y < 0$ then y is misclassified. And to find out such a weight vector a we have adopted criteria function. So, we have said that if, a is a solution vector in that case the criteria function will be minimum, if it is not a solution vector in that case the criteria function will be non-minimum. So, in general if we have a criteria function say J which of course is a criteria function of a .

We have to put it in function of a because we are trying to find out the proper value of a . And then we have adopted a gradient descent procedure to converge to proper value of it. So, the algorithm that we have adopted is initially we assume an iterative algorithm at $a(0)$ which is arbitrary and $a(k+1)$ was obtained from $a(k)$, $a(k+1) = a(k) - \eta(k) \nabla J(a(k))$, where this gradient will be taken with respect to vector a . So, this is the general gradient descent algorithm. And in perceptron criteria we have defined the criteria function $J_p(a)$

$$\sum_{\text{y missclassified}} (-a^T y), \text{ take the summation over all } y \text{ which are misclassified by } a.$$

(Refer Slide Time: 06:26)



So, following this we have come to an algorithm following the criteria that, again as in general case we start with an arbitrary weight vector a . And then every iteration a is defined following the similar algorithm that is $a(k)$. Now, because this term is negative, $-a^T y$. So, over here this will become positive so instead of minus it will be plus. So, it becomes $a(k) + \eta \sum_{\text{y missclassified}} (-y)$, summation of y for all y which are misclassified.

So, what we have said is that this criteria function $J_p(a)$ will attain a value equal to 0. When all the samples are properly classified by the weight vector a otherwise it will be non 0. So, as a result you will find that $J_p(a)$ can have a minimum value which is equal to 0 that means it has a global minimum and because it has a global minimum. So, following this iterative procedure, iterative algorithm we can attain the global minimum which will be equal to 0 whenever a is my solution vector.

Now, you find that there is a problem over here, the problem is in terms of memory required for an execution of this algorithm because in all real life situations we will have thousands of training samples for supervised learning of the classifiers. So, when I have thousands of training samples obviously I will have thousands of samples which will be misclassified initially. And this says I have to have summation of all the samples which are to be added to this vector weight vector a , so I need large amount of memory.

So, I can have that instead of taking all the samples together instead of considering all the samples together, I can consider sample by sample. That means I can have a sequential version of this particular step. So, what I have is I have samples. So I have samples y^1, y^2, y^3 it continues at the k^{th} instance I will consider that sample y^k that is the k^{th} sample vector. And we have n number of samples y^n .

So, what I will do is I will start with a particular weight vector that is the initial weight vector test this y^1 whether $a^t y^1 > 0$ or not. If I find that $a^t y^1 > 0$, I do not modify the weight vector a , I simply pass on to the next sample y^2 . So, if I find out that $a^t y^2$ is also greater than 0 that means the same weight vector. It properly classifies y^1 it also properly classifies y^2 . And when I consider this all the samples which we will actually belong to ω_2 they are already negated.

So, I assume that all the samples y^2 they are already negated. So, in all the cases I will have single criteria decision rule that $a^t y^k > 0$, if y^k is properly classified by the weight vector a . Now, suppose I find that $a^t y^k < 0$ before that up to y^{k-1} all the samples, for all of them $a^t y > 0$. So, I need not have to modify my weight vector a . So, when I find that a transpose $a^t y^k < 0$, that means it is the k^{th} sample, it is the k^{th} sample that has been misclassified by the weight vector a . So, as I determine that k^{th} sample has been misclassified by the weight vector a . So, immediately I have to update a

and for updation, I will follow the procedure same as this, but here instead summation of y I will have only y^k .

And with this modified vector I continue with other samples. Now, if all of the samples are properly classified by this modified weight vector I need not modify weight vector anymore up to this y^n , but I am not sure what happened to y^1 to y^{k-1} . Thus y^1 to y^{k-1} was properly classified by the previous weight vector. And I have modified the weight vector when I encountered k^{th} sample.

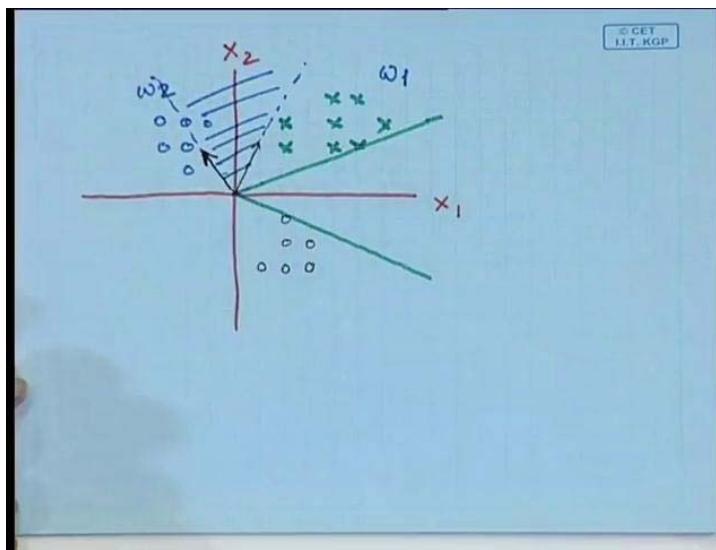
So, this modified weight vector I am not sure whether this will also properly classify all this previous samples or not so I have to try it again. And finally, the algorithms communicate when in a single pass all the samples are properly classified by that particular weight vector. So, whenever the weight vector is modified on that modified weight vector I have to try put all the samples.

If all the samples are properly classified by a weight vector, then only I will say that my classifier has been designed or the weight vector that has I have got that is my solution vector. So, following this now the new algorithm will be again as before $a(0)$ is arbitrary and $a(k+1) = a(k) + \eta y^k$. Now, you find that summation term is no more there.

So, here because we are considering sample by sample not all the misclassified samples together so the memory requirement for the execution of this algorithm will be much less compared to the previous version. Now, let us see that whether this ensures that the algorithm will really converge. So, that we will tell to demonstrate with the help of this sequential algorithm. I will get the same result whether I consider all the samples together or I samples I consider sample by sample. So, let us to demonstrate that let us consider a case in two dimensions because that is the one I can plot on the paper.

(Refer Slide Time:

14:01)



So, let me consider case in two dimensions. So I will have two dimensional samples suppose the components are x_1 and x_2 . And suppose I have a number of training samples say these are the samples which belong to class say ω_1 say arbitrary set of training samples. And suppose these are the samples which belong to class ω_2 .

So, these are from class ω_1 and these are from ω_2 . So, as I said that to have and uniform decision rule I negate all the samples which are given from class ω_2 .

So, all these samples have to be negating so the negated samples will be like this I will have over here then these are the negated samples. So, if this is y this is $-y$. Now, when I try to find out the weight vector we said that the weight vector is orthogonal to the decision surface orthogonal to the hyper plane. Now, as we are considering two dimensional cases so that hyper plane is equivalent to a straight line.

Now, what are the straight lines that or the set of straight lines which actually demarcates between these two different classes ω_1 and ω_2 . So, you find that I have some limiting cases, what are those limiting cases? One of the limiting case is say this one, this is one of the limiting case because the moment this particular straight line is rotated anti clock wise further this sample is going to be misclassified. If I rotate it clock wise all these samples will be properly classified, I have some zone over which even if the straight line is rotated all the samples are properly classified until and unless I reach this particular position.

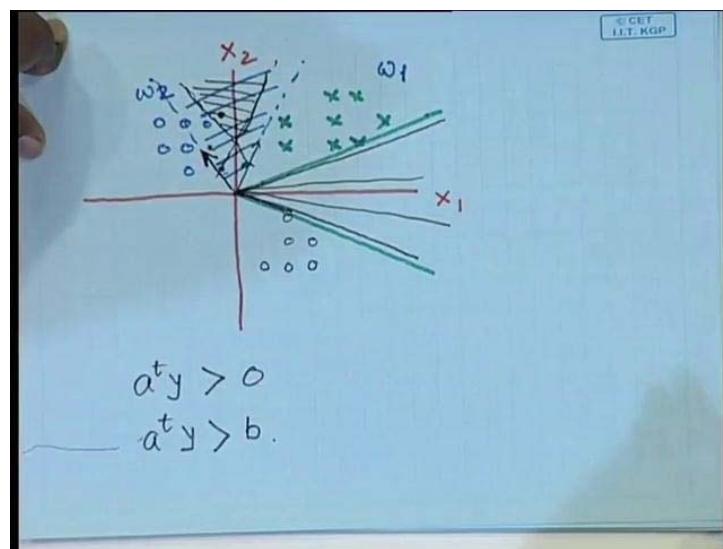
So, the moment I reach this position if the straight line is rotated further in the clock wise direction then this samples going to misclassified. Now, within this region whatever with the position of this straight line whatever be orientation of this straight line, the straight line is going to classify this set of samples from ω_1 also this set of samples from ω_2 . Both of these sets will be properly classified for any position of straight line within this region? Now, what is the nature of the corresponding vectors?

We said that weight vectors are orthogonal to the height of the planes that is over here the weight vectors will be normal to the straight lines. So, one limiting case of the weight vector is over here which is orthogonal to this straight line, perpendicular to this straight line. And the other limiting case will be this and this weight vector is perpendicular to this straight line. So, you find that I have a conical region, so I have a conical region like this where if the

weight vector is within this conical region, I mean any, many weight vector within this conical region is going to solve my purpose correct, right?

So, once I know so this is what is my solution region I call it the solution region. So, any weight vector within this solution region it will classify this ω_1 all the samples in ω_1 and all the samples ω_2 properly, So, my aim should be that when the algorithm converges, the algorithm should give a weight vector within this solution region, that is within this conical space. Let us how we really get it.

(Refer Slide Time: 19:24)



So I will start with some smaller number of samples for algorithm illustration. So, these are say samples from class ω_1 . And say these are samples from class ω_2 .

So, given this my limiting decision boundary, decision surfaces are one is this, another one is this. And the corresponding, I assume that these samples two ω_2 from class ω_2 are already negated. So, these are the negated samples. And the solution region in this situation will be this, bounded by this conical space.

This is the solution region, so this is my dimension x_1 . And this is the dimension x_2 . Now, because initially we said that $a(0)$ is arbitrary. So, I cannot ensure the initial choice $a(0)$ will be within this solution region. That is the weight vector in this only region. Let us assume that my initial weight vector because it is arbitrarily chosen, is chosen somewhere over here. So, this is my $a(0)$ right? When this is $a(0)$ the initial weight vector, the decision surface which is actually defined by $a(0)$ is orthogonal to this one, which is orthogonal to this.

Now, find that when this is my decision surface this decision surface properly classifies this sample as well as this sample, it also properly classifies all the samples which are in class ω_2 , but this decision surface misclassifies these three samples. These three samples actually belong to class ω_1 , but this weight vector classifies them to class ω_2 . So, these are three samples which are misclassified. Now, find suppose I take the samples in a sequence this is my y^1 .

So, I initially consider this sample for testing, I found that this sample is properly classified. So, I do not modify my weight factor. Then I have considered this sample this is also properly classified, so I do not modify my weight factor. The third sample is possibly this one and for this sample I found that a transverse y has become less than 0 because this sample is misclassified. And what is the algorithm, my algorithm says $a(k) = a(k-1) + \eta \sum_{y \text{ missclassified}} (-y)$

right?

So, what it means that this vector y will be scaled by a factor η and that will be added to $a(k-1)$. So, this is the vector y which has been misclassified a scaled version of this will be added to this weight factor w . That means the weight factor W will be moved in the direction of this vector y by certain fraction. So, this was my initial W , now say W is moved to somewhere over or let us put it into the decision region somewhere over here.

If this η is such that the scale factor that has been imposed on y is such that the vector has been pushed into the decision region or suppose the vector has been pushed somewhere over here. So, it is still it is outside the decision region. So, this was my $a(0)$, now this has become the position of $a(1)$. Now, this $a(1)$ will properly classify this sample, this sample, this sample, but it may not properly classify this sample.

If it does not classify this sample in that case determine, to determine $a(2)$ I have to add to $a(1)$ a scaled version of this sample. Now, this is in this direction so this vector will be moved in the same direction of $a(2)$ by certain factor. And while doing this you find that now it may be possible that this $a(2)$ has come within the decision region. So, once this $a(2)$ is pushed into the decision region all the samples over a single pass will be properly classified by this weight factor a .

So, effectively what I am doing is in every pass I am pushing the weight factor in the direction of the misclassified sample by certain distance. And while doing so I am ensuring that this weight factor will be pushed into the solution region. Now, it may so happen that the

value of eta is so high that the weight of convergence is so high that when I am trying to come to this $a(1)$. For this $a(1)$ it has totally moved out of the solution region and $a(1)$ has come somewhere over here. That is also possible if the value of eta is very high and if $a(1)$ is pushed over here.

Then you find that all this samples will be misclassified whether I mean all the samples in ω_1 from class 1 they will be properly classified, but all the samples from ω_2 they will be misclassified. And when the samples from ω_2 are misclassified actually the samples from ω_2 are on this side not on this side. This is actually negated version I am adding y that is the unnegated version.

So, because these samples are on this side I will have to add the corresponding samples to this location to this weight vector. And weight vector will start moving in the reverse direction. So finally, I will get a situation that when the weight vector will actually converge, when the algorithm converges the weight vectors will be within the solution region.

So, that means this perceptron criterion ensures that when the algorithm is terminated or converges I actually have an weight vector within the solution region, is that okay? Now, you find that there is still some problem, what is the problem. If at convergence, I get an weight vector somewhere over here just within the solution region or somewhere over here which is also just within the solution region.

So, still my algorithm will converge and for these training samples I get a weight vector which properly classifies these training samples, but you remember that these training samples are just representatives they do not cover the entire set of samples. Now, what happens if the training sample is somewhere if the weight vector converges here somewhere over here then actually I am getting a decision surface which is just outside this sample, right? So, if the weight vector converges here I get the decision surface which is just outside this sample.

So, it just barely classifies these two sets of samples, but because these are representative there is no guarantee that I will get other samples from this class which will be on this side or I will get samples from this class which will be on this side. So, to minimize the risk what I would like to have is I would like to have a decision surface in a narrow region somewhere over here, so that the risk is minimized. If I want to do that that means I want to restrict this solution region somewhere over here. So, I do not want to be satisfied simply by $a^T y > 0$.

If I want to constraint my solution region within a sub-region of the solution region which is well within the solution region, that means I will be satisfied only when I get weight vector over here I am not satisfied if I get a weight vector over here. So, effectively what it means is that this solution vectors or these decision regions they are pushed inside, right? And how I can ensure that I can ensure it not by my decision rule that $a^T y > 0$, this will be ensured by if I put some margin that $a^T y$ is greater than some margin b , where b is a positive constant.

So, in this criteria to decide whether this y^k is classified or misclassified, I will not use $a^T y^k > 0$ to decide if it is misclassified rather I will say if $a^T y^k > b$ then it is properly classified or safely classified. If it is less than b , but greater than 0 it is properly classified, but still not safe. So, instead of having my decision rule to be $a^T y^k > 0$ for proper classification, I will put the decision rule as $a^T y$ greater than the margin b for proper classification.

And by doing that if I take the misclassified samples then effectively I am ensuring that I land off with the weight vector within this restricted region. And then the classifiers that I get those are reasonably safe. I can never say that it is absolutely safe, but that reasonably safe given these two sets of training samples from class ω_1 and class ω_2 , is that okay? But this perceptron criterion is not the only criteria that can be used for this linear this classified design. Then there can be other many other variations of this perception criteria function and accordingly, we can have different algorithms. One of the variations of the criteria functions is defined like this.

(Refer Slide Time: 32:43)

The notes are handwritten on a whiteboard:

- $J_r(a) = \frac{1}{2} \sum_{y \text{ misclassified}} \frac{(a^T y - b)^2}{\|y\|^2}$
- Relaxation Criterion
- $\nabla J_r(a) = \sum_{y \text{ misclassified}} \frac{a^T y - b}{\|y\|^2} \cdot y$
- $a(0) \text{ arbitrary}$
- $a(k+1) = a(k) + \eta \sum_{y \text{ misclassified}} \frac{b - a^T y}{\|y\|^2} y$

$$J_r(a) = \frac{1}{2} \sum_{y \text{ missclassified}} \frac{(a^t y - b)^2}{\|y\|^2}, \text{ for all } y \text{ which are misclassified. This is another criteria}$$

function that can be used for getting a proper weight vector a . And this has to be minimized of course, and this is a criteria function which is called relaxation criteria. And accordingly the algorithm that we get for minimization of this criteria function to obtain the weight vector a is called relaxation procedure, but that is also done following the same gradient descent procedure that is if I take the gradient of this $J_r(a)$ of course this gradient is with respect to

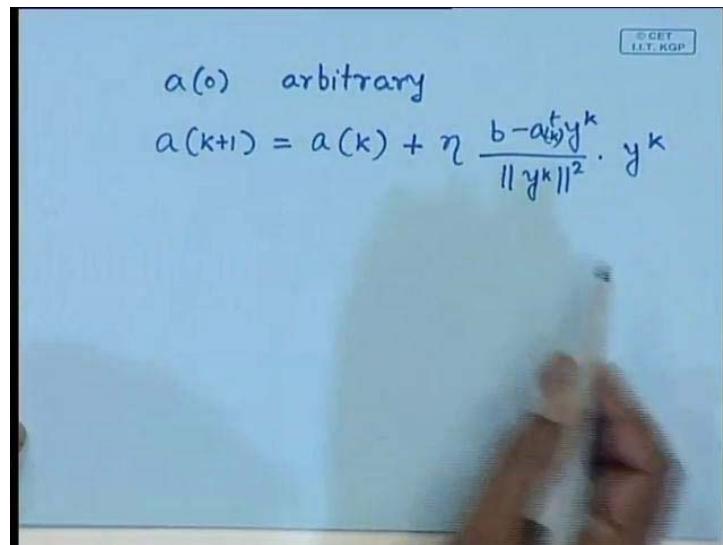
$$\text{the weight vector } a. \text{ This will be simply } \nabla J_r(a) = \sum_{y \text{ missclassified}} \frac{b - a^t y}{\|y\|^2} \cdot y, \text{ over all } y \text{ which are}$$

misclassified.

So, when I do this again my algorithm remains the same that is I choose $a(0)$ arbitrarily. And to decide about $a(k+1)$, I simply take $a(k+1) = a(k) + \eta \sum_{y \text{ missclassified}} \frac{b - a^t y}{\|y\|^2} \cdot y$, again for all y which are misclassified.

So this is the relaxation algorithm or relaxation procedure. So, again you find that over here I have summation of this term for all the samples which are misclassified. So, again the algorithm is memory intensive.

(Refer Slide Time: 35:53)



So, I can still have a sequential version of this and the sequential version as before will be you choose $a(0)$ arbitrarily. And $a(k+1)$ can be obtained using the iterative procedure

$$a(k+1) = a(k) + \eta \frac{b - a^t(k)y}{\|y\|^2} \cdot y^k. \text{ So, again over here we consider the samples sequentially one after another.}$$

The moment I encounter a sample which is misclassified I update the weight vector $a(k)$ following this updation rule. So, this is sequential version of the same relaxation procedure, right?

And there may be other criteria functions as well for designing of the classifier, but one thing you must have noticed that whether, I use the perceptron criteria or I use the relaxation procedure. In both these cases the convergence is guaranteed if the classes are linearly separable. Otherwise the algorithm will never converge there will be at least one sample at some point of time which will be misclassified is that okay? So, the convergence is not guaranteed. So, I can make use of these algorithms only when I know for sure that the classes are linearly separable otherwise I will not get any convergence.

However, if I am not sure if the classes are linearly separable or not still I can try to design a linear classifier, but there the, my aim will be that whichever linear classifier I design that will give me the minimum error. So, accordingly I can have a classifier based on minimum squared error.

(Refer Slide Time: 38:24)

A photograph of a whiteboard with handwritten mathematical notes. At the top, the text "Minimum Squared Error" is written above a horizontal line. Below this line, the inequality $a^t y \geq b$ is written. To the right of this, a small rectangular box contains the equation $a^t y = b$. Below the box, the equation $a^t y^i = b^i$ is written. A person's hands are visible at the bottom of the board, holding a piece of chalk.

So, how should I go about this, you find that so far my decision rule was that if $a^T y > b$, I assume y is properly classified, right? So, what is my decision surface? Decision surface is $a^T y = b$. So, I have to get a solution to this equation $a^T y = b$. And this, the solution of this equation can be obtained by this minimum squared error procedure. So, over here you will find if I consider the i^{th} sample my condition will be $a^T y^i = b$ or to be more general let me assume that this will be $a^T y^i = b^i$ if for every sample I can have different components or different margins for generalization.

So, for every i^{th} sample I have such an equation I have n number of such samples. So, I get n number of such equations then what I have to try to do is, what that I have this n number of simultaneous equations. And I have to solve this number of simultaneous equations, is that okay? In matrix form it will be something like this say

$$\begin{bmatrix} y_{10} & y_{11} & y_{12} & \dots & y_{1d} \\ y_{20} & y_{21} & y_{22} & \dots & y_{2d} \\ y_{30} & y_{31} & y_{32} & \dots & y_{3d} \\ \dots & \dots & \dots & \dots & \dots \\ y_{n0} & y_{n1} & y_{n2} & \dots & y_{nd} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \text{ So, that } n \text{ number of simultaneous linear equations}$$

can be put in a matrix form like this.

(Refer Slide Time: 40:26)

$$\begin{bmatrix} y_{10} & y_{11} & y_{12} & \cdots & y_{1d} \\ y_{20} & y_{21} & y_{22} & \cdots & y_{2d} \\ y_{30} & y_{31} & y_{32} & \cdots & y_{3d} \\ \vdots & & & & \\ y_{n0} & y_{n1} & y_{n2} & \cdots & y_{nd} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\Rightarrow ya = b \Rightarrow a = y^{-1}b.$$

And in the compact form I can write it as $Ya = b$. So, you can say that what is great in it I can as well if $Ya = b$ I can as well get $a = Y^{-1}b$, but the problem is this Y is not a square matrix it is a rectangular matrix. Number of rows which is equal to the number of samples

that is n , and the number of columns is $d + 1$ or \hat{d} . So, the number of rows is much less than the number of columns. So, in this case the vector a is actually over determined. And following this simple approach I cannot get an exact solution for this vector a . So, that is the major problem, so how to go about it, how to get the solution for this vector a ? So, to get the solution for this vector a we can define an error vector.

(Refer Slide Time: 42:51)

The image shows a handwritten derivation on a whiteboard. At the top, the error vector $e = Ya - b$ is written. Below it, the Sum-of-Squared Error Criterion is defined as $J_s(a) = \|Ya - b\|^2$. This is then expanded into the sum of squared differences between each sample's prediction and its target value: $= \sum_{i=1}^n (a^T y_i - b_i)^2$. Finally, the gradient of $J_s(a)$ is derived as $\nabla J_s(a) = \sum_{i=1}^n 2(a^T y_i - b_i) \cdot y_i = 2y^T(Ya - b)$.

So, that error vector e is actually $Ya - b$. And our aim will be to get a solution for a that minimizes this error, is that okay? So, instead of simply trying for $Y^{-1}b$ we will try to get a solution a , because Y is known this is nothing but the set of training samples b . Let us assume that this is also known that is the margin vector. So, what I have to do is I have to try to get a solution for a , which will minimize this error vector e .

So, for that let us define a criteria function which is sum of squared error. So, let me write it as $J_s(a)$ which is $J_s(a) = \|Ya - b\|^2$. And which in the expanded form is nothing but $\sum_{i=1}^n (a^T y_i - b_i)^2$. So, we compute the error for each and every sample vector take the square of it sum over all the samples that gives you sum of squared error. And what I will try to do is I will try to minimize this squared error, while trying to get a solution for this vector a one of the approaches is obviously the gradient descent approach.

You start with the initial weight vector and go then go on updating it. So, the gradient of $J_s(a)$ with respect to a from here will be nothing but $\nabla J_s(a) = \sum_{i=1}^n 2(a^T y_i - b_i) \cdot y_i$, which can be

simply be written as in the matrix form $2Y^t(Ya - b)$. So, using this gradient I can make use of the gradient descent procedure to obtain this solution for a . Otherwise I know that when I obtain a solution $2Y^t(Ya - b)$ has to be equal to 0 this term has to nullify accordingly.

(Refer Slide Time: 46:05)

The image shows a handwritten derivation on a blue background. At the top, it says "Closed form Solution." Below that, it shows the equation $\nabla J_s(a) = 0$. Then, it derives $2Y^t(Ya - b) = 0$, which simplifies to $Y^t Y a = Y^t b$. This is then solved for a as $a = (Y^t Y)^{-1} Y^t b$. A red note below this indicates that $a = Y^+ b$, where Y^+ is labeled as the "Pseudo Inverse of Y". There is a small watermark in the top right corner that reads "© CET I.I.T. KGP".

So accordingly, I can obtain a closed form solution which is given by, just by equating gradient of $\nabla J_s(a) = 0$. And that simply gives me because $\nabla J_s(a)$ is nothing but $2Y^t(Ya - b)$. That has to be equated to 0, that is this being a matrix equation it has to be equated to null matrix. And from here I will get $Y^t Ya = Y^t b$. And from here I get a solution for a which is equal to $(Y^t Y)^{-1} Y^t b$.

Now, over here you find that though Y is a rectangular matrix of dimension $n \times d$, but $Y^t Y$ that will be a square matrix of dimension $d \times d$. So, Y^t is d by n , Y is n by d , so d by n multiplied by n by d matrix it becomes a square matrix. And quite often this matrix is nonsingular. So, this being a square matrix and if it is non-singular I can find out $(Y^t Y)^{-1}$.

And this term $(Y^t Y)^{-1} Y^t$ this is what is known as pseudo inverse.

So, I get $a = Y^+ b$ where Y^+ is the pseudo inverse Y . And pseudo inverse is defined as $(Y^t Y)^{-1} Y^t$. So, this Y^+ this is actually pseudo inverse of Y . And you find that if Y is square and non-singular in that case the pseudo inverse will be same as regular inverse of Y that you

can verify. So I get a solution like this, a closed form solution for the weight vector a , but we said that in most of the cases this pseudo inverse exist, in most of the cases $Y^t Y$ is nonsingular, it is square. And in most of the cases it is non-singular, but there is no guarantee that it will always be non-singular, but still we can find out a solution to this weight vector a following again some iterative algorithm starting from here. So, that is an algorithm which is called Widro-Hoff procedure or LMS procedure, least mean square procedure.

(Refer Slide Time: 49:47)

© C.E.T.
I.I.T. KGP

Widro-Hoff / LMS Procedure

$$a(0) \text{ arbitrarily.}$$

$$a(k+1) = a(k) + \eta Y^t (b - Ya(k))$$

a(0) arbitrary

$$a(k+1) = a(k) + \eta [b^k - a^T(k) y^k] y^k$$

So, what is that we have Widro-Hoff or LMS procedure. So, what is this LMS procedure again we will assume that $a(0)$, will be chosen arbitrarily. And $a(k+1) = a(k) + \eta Y^t (b - Ya(k))$. So, as I said you find that this is the gradient of or mean square error criteria. So, if I follow the gradient descent procedure to obtain the $k+1$ weight vector $a(k+1)$, it is $a(k) + y^k$.

So, before this if we want we can put this convergent rate η so it will be $a(k) + \eta y^k$ initially it was negative. So, this $Ya - b$ will now be $b - Ya$ because now I have made it positive. So, this is $\eta Y^t (b - Ya(k))$. Now, find that over here all the samples, all the training samples are considered together because it is a matrix equation. So, the same problem of memory requirement, it will require large amount of memory for execution of this algorithm.

So, to solve that again I can have a sequential version of the same algorithm. And the sequential version is similar to what we have done before that is we assume this $a(0)$ to be

arbitrary. And $a(k+1) = a(k) + \eta [b^k - a^t(k)y^k]y^k$. So, this is the sequential version of the same least mean square procedure which makes use of the iterative technique to find out the weight vector a , is that okay? So, over here you will find that since I do not have to find out the inverse of the matrices.

So, this guarantees that I will always get a solution and my aim is to have a solution obviously I cannot have a perfect solution because I have not assumed the linear separability, but what I wanted to do is I wanted to get a vector which will give me a linear classifier. And that linear classifier will ensure that your least mean square error will be minimum. So, I will stop here today next day we will continue with other topics.

Thank you.

Pattern Recognition and Applications
Prof. P.K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 22
Linear Machine

Good afternoon, so we are continuing with the different criteria functions which are used for designing for the linear discriminator, so far we have discussed about three types of criteria functions. The first one was the perceptron criteria then we have talked about the relaxation criterion and the last one we have said was the minimum squared error criteria, so what we have seen in case of perceptron criteria is that.

(Refer Slide Time: 00:51)

J_p = \sum_{\forall y \text{ misclassified}} -a^t y \rightarrow \text{Perceptron Criterion}
$$a(0) \leftarrow \text{arbitrary}$$

$$a(k+1) = a(k) + \eta \sum_{\forall y \text{ misclassified}} y \rightarrow \text{Sequential}$$

$$a(k+1) = a(k) + \eta y^k$$
 A pencil is visible at the bottom right of the note."/>

You define a criteria function which is given by J_p which is nothing but $\sum_{\forall y \text{ misclassified}} (-a^t y)$,

for all y which are misclassified. So, this was the perceptron criterion function and our aim was that we wanted to find out the weight vector a , which will actually classify all the training samples properly. So, when all the samples are the training samples are classified properly in that case the criterion function, the perceptron criterion function J_p has to be equal to 0 which is the minimum value. If any sample is misclassified by the weight vector a , then the criterion function J_p will be greater than 0 and following this what we have done is we have taken the gradient of this criterion function J_p with respect to a .

Then we have followed the gradient descent procedure to come to a particular solution, to come to the solution of the weight vector a . Accordingly, we had the weight of updation rule or the perceptron algorithm which was given like this that $a(0)$ was initially selected arbitrarily and then in every iteration $a(K+1)$ was obtained from the previous value $a(k)$ following the gradient descent procedure. It was simply $a(k+1) = a(k) + \eta \sum_{y \text{ misclassified}} (-y)$,

for all y which are misclassified and there we have seen the problem with this algorithm is that because we have to take the summation of all the misclassified samples.

So, the amount of memory which is required for execution of this particular algorithm is quiet high, so we had a sequential version of this algorithm. Where, the first step that is selecting $a(0)$ arbitrarily remains the same and what you are doing is you are taking the samples one by one and whenever a sample is misclassified instead of trying to aggregate all the misclassified samples together. Whenever a sample is misclassified, immediately you go for the updation of weight vector a , so accordingly our weight updation rule was sample by sample.

So, if the k^{th} sample is misclassified then immediately you update the weight vector following this updation rule, that is $a(k+1) = a(k) + \eta y^k$, where this y^k is the sample which is misclassified by the weight vector $a(k)$ and η , we said that this is the convergence factor and similarly we have also talked about the relaxation procedure,

(Refer Slide Time: 04:19)

Relaxation Criterion

$$J_r(a) = \frac{1}{2} \sum \frac{(a^t y - b)^2}{\|y\|^2}$$

forall y misclassified.

$$a(0) \leftarrow \text{arbitrary}$$

$$a(k+1) = a(k) + \eta \sum \frac{b - a^t(k)y}{\|y\|^2} \cdot y$$

forall y misclassified.

→ Sequential

$$a(k+1) = a(k) + \eta \frac{b - a^t(k)y^k}{\|y^k\|^2} \cdot y^k$$

or the relaxation criterion where the problem faced with the perceptron criterion that is we can have a solution vector which is just on the boundary. So, if you get a solution vector which is just on the boundary then it is quite likely that if I use that vector for classification of unknown samples. Then the probability of misclassification of the probability error will be quiet high, so what you want is that the weight vector should be well within the solution region.

So, that problem is solved by this relaxation criterion and this relaxation criterion the criterion function was modified as $J_r(a) = \frac{1}{2} \sum_{y \text{ missclassified}} \frac{(a^t y - b)^2}{\|y\|^2}$. Here, this vector y is misclassified, so it has to be done for all y misclassified, but our design rule remains the same that is you take the gradient of this with respect to a . Then follow the gradient descent procedure starting from some initial weight vector which is chosen arbitrarily and, so we have this algorithm relaxation algorithm which is given as $a(0)$ is chosen arbitrarily.

And then we have this iterative approach that is $a(k+1) = a(k) + \eta \sum_{y \text{ missclassified}} \frac{b - a^t y}{\|y\|^2} \cdot y$, for all y misclassified. And then we have seen that the corresponding sequential version where the updation rule is modified as $a(k+1) = a(k) + \eta \frac{b - a_{(k)}^t y^k}{\|y^k\|^2} \cdot y^k$. Where y^k is the sample which is misclassified by the weight vector $a(k)$ and then we have said that both these criterion functions whether it is perceptron criterion or the relaxation criterion, this is useful when the samples are linearly separable or the classes are linearly separable in the sense that given the set of training samples from two classes.

Say ω_1 and ω_2 , I should be able to draw a boundary between these two classes where this boundary a linear boundary. So, in case of 2 dimensions it has to be a straight line and in case of 3 dimensions it has to be a plane or multi dimensions it has to be hyper plane.

So, it is actually a linear boundary between the two classes ω_1 and ω_2 , but if the classes are not linearly separable then neither the perceptron criterion based algorithm nor the relaxation based relaxation criterion based algorithm, they will give you unique solution and the algorithm will never converge, so in such cases we have this minimum squared error algorithm, or

(Refer Slide Time: 08:20)

MSE Criterion

$$e = Ya - b$$

$$J_s(a) = \|Ya - b\|^2$$

$$= \sum_{i=1}^n (a^t y_i - b_i)^2$$

$$a = Y^+ b \quad Y^+ \rightarrow \text{Pseudo inverse of } Y.$$

$$a(0) \leftarrow \text{arbitrary}$$

$$\begin{aligned} a(k+1) &= a(k) - \eta Y^t [Ya(k) - b] \\ a(k+1) &= a(k) + \eta [b(k) - a^t(k) y^k] y^k \end{aligned}$$

MSE criterion function, where the criterion function is based on an error measure where error is nothing but $Ya - b$ where Y is the collection of all the training samples, a is the weight vector and b is called a margin vector. So, for these MSE criteria the aim is to reduce this square of this error or minimize the square of this error. So, accordingly you define the squared error criteria which is given by $J_s(a)$, which is nothing but $\|Ya - b\|^2$ and I can expand it as $\sum_{i=1}^n (a^t y_i - b_i)^2$ take the summation over all the samples y is equal to 1, to n if I have n number of training samples.

And then again if I take the gradient decent procedure or going for that pseudo inverse concept I get a solution which is given by $a = Y^+ b$ where this Y^+ is actually pseudo inverse of Y . And then we have said that we can still have an iterative algorithm following the gradient descent procedure and in which case the iterative algorithm will be similar to the algorithms that we have said earlier.

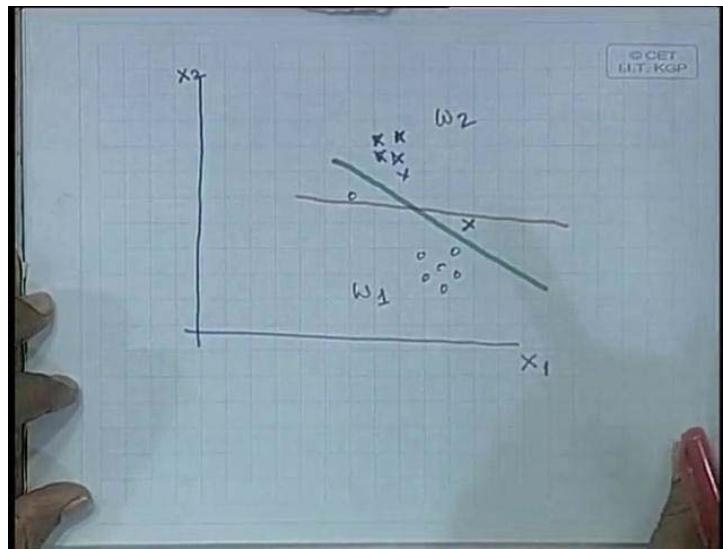
That is perceptron criterion relaxation criterion, here again or initial weight vector is 0 this will be chosen arbitrarily. In every iteration step, a will be modified as $a(k+1) = a(k) + \eta Y^t (b - Ya(k))$, but still you find that, here we are considering all the samples together to find out the solution vector. So again the problem of the memory and working with large matrices, so we can also have in the same manner a sequential version of this iterative algorithm, and in the sequential version of the iterative algorithm.

The iteration step will be modified as $a(k+1) = a(k) + \eta [b^k - a^t(k)y^k]y^k$. So, find that we can go for a sequential algorithm from this updation state, from this updation step which actually considers all the samples together. Now, again this iterative algorithm or the mean square error criteria function based iterative algorithm to find out a solution vector is a problematic in the sense that we have said the perceptron criteria and the relaxation criterion are useful when the classes are linearly separable and if the classes are not linearly separable in that case we can go for this minimum based error criterion function.

But, if I have a problem whether the classes are actually linearly separable, but because we do not know beforehand whether the classes are linearly separable or not, so a safest approach will be that you go for minimum squared error criterion based approach to get your solution vector a , but the problem here is even if the classes are linearly separable it is not guaranteed that linear that this minimum squared error based criterion function based algorithm will give you a weight vector which will linearly separate the two classes or it is not guaranteed that I will have a linear boundary between the two classes. Where the samples from two

different classes will be put into two different regions, so I can have a situation something like this that.

(Refer Slide Time: 13:22)



Even if I have the samples belonging to 2 classes which are given like this suppose this is a set of samples which actually belongs to say class ω_1 and I can have a set of samples over

here. Let me use different symbol something like this where these samples actually belong to class ω_2 , now here you find that I can easily, so these are my two features x_1 and x_2 .

So, I have a two dimensional feature vector, so now here you find that it is quiet easy to find out that I can a line separating these two classes, so because this are linearly separable, so I should get an weight vector which will actually define this line. But, when I go for this mean square error based criterion function, I may land up with a boundary something like this. Because in case of mean square error based criterion function, it tries to minimize the sum of squared distances of the given feature vectors from the line So, as a result the boundary that you get that may not separate the samples belonging to two different classes, so how do you solve this particular problem.

So, though this MSE or minimum squared error based criteria function may lead to this kind of problem, however, it can be shown that if the classes are really linearly separable then it is possible to have some a,

(Refer Slide Time: 15:23)

$$y_a = b$$

$$b > 0$$

$$J_a(a, b) = \|y_a - b\|^2$$

$$\nabla_a J_b = 2 y^t (y_a - b)$$

$$\nabla_b J_b = -2 (y_a - b) \Leftarrow$$

$$\text{for } a' b \Rightarrow a = y^+ b$$

weight vector a , and some bias vector b where $y_a - b$ by a minus b or $y_a = b$ where this $b > 0$, we did not have any such constant when we talked about minimum squared error criteria function. So, if the classes are actually linearly separable in that case it is possible to have an weight vector a and a margin vector b where this equation will be satisfied.

That is $Ya = b$, where $b > 0$, so when I say $b > 0$ that means every component of b is greater than 0 because b is a margin vector, so you say that every component and this will have n number of components if I have n number of samples. So, when I say this margin vector b is greater than 0, then every component of this margin vector is greater than 0 and assuming this we have to define a criteria function earlier.

We have defined the criteria function as a function of only the weight vector a , because our aim was to select the weight vector. But, now what we have to do is we have to select both a and b because we do not know which particular b will satisfy this equation for a particular a . So, we have to try to update or iterate on both a and b , so again we have to start with an initial value of a and initial value of b and then we have to update on b . For every updated value of a , and for every iterated value b we can find out what is the corresponding a , so while doing that where we update b .

We have to keep in mind that in none of the cases any component of b can be less than 0, we always want to maintain that every component of b will be greater than 0. And that can be satisfied if we start with b , with every component positive and while updation, while modification of b , we will allow only a component of b to be incremented not to be reduced. It is because if we allow a component of b to be reduced then there is a possibility that a component of b may come down below 0 which will not satisfy this condition.

So, while updation we have to keep in mind that we will always allow increment of b , but we will not allow reduction of b , because that is a matter of scale factor I have $Ya = b$. So, what I am saying is I will always increment the value of b , I will not reduce the value of b because reduction of value of b may lead to a condition that a component of b say b_i may be less than 0. So, this is what I will not permit I will always increment the value of b and the scale factor that is incorporated, that will be reflected in the corresponding value of a in the weight vector is that okay?

So, when I do that, I will redefine my criteria function that is J_s , earlier it was defined as a function of only a , now I have to define this as a function of both a and b where a is the weight vector and b is the margin vector. However, the definition will remain the same that $J_s(a,b) = \|Ya - b\|^2$, fixed that was fixed in MSE algorithm. Our assumption was the margin vector was fixed and for a given margin vector, we are trying to find out a value of weight vector. So, the problem that we have faced that whenever we fix the margin vector because

we never ensure that this condition will be satisfied as accordingly, we could have got a boundary something like this, it is still a linear boundary.

So, this boundary ensures that your sum of square error will be minimum, but this does not ensure that the classes are actually linearly separated. So, if the classes are linearly separable, then we have said that we can have a solution of this form $Ya = b$, where b is greater than 0. So, we will take the different margin vector which are greater than 0, and then find out the corresponding value a , so while doing. So, we start with a merging vector and then go on updating the margin vector with a to complete proper solution.

So, this is what we have, this is our criteria function, squared error criteria function which is given by $J_s(a,b) = \|Ya - b\|^2$. Then this error function we have to take the gradient, once with respect to a , and then with respect to b because we want to update both, so I take the gradient of this J_s with respect to a . The solution, the gradient with respect to a is what we have seen earlier $\nabla_a J_s = 2Y^t(Ya - b)$, this is part we have seen earlier and I will take the gradient with respect to b .

Then the gradient with respect to b is $\nabla_b J_s = -2(Ya - b)$. Now, once we obtain any value of b then the corresponding value of a can be obtained as. So, for any b the corresponding weight vector a can be obtained as $a = Y^+b$ or we can go in for the same iteration which we have done when we discussed about the mean square error algorithm. If we are not sure that this pseudo inverse is nonsingular because if the pseudo inverse is singular we cannot have, if $(Y^tY)^{-1}$ is singular then we cannot compute the pseudo inverse.

So, for the computation of a , either we can go for this, if you assure that $(Y^tY)^{-1}$ is singular or we can imply the same iterated algorithm that we have discussed earlier. So, here our main aim is that we have to find out a proper value of b by allowing the gradient descent procedure with the help of the gradient of J_s with respect to b . And while doing so, as we said that we have to ensure that none of the components of b can all below 0 and that what we said that we can only allow increment of the components of b , we cannot allow reduction of the components of b , so if I want to do that in that case.

(Refer Slide Time: 23:21)

$$b(k+1) = b(k) - \eta \frac{1}{2} (\underbrace{\nabla_b J_s}_{\text{CET I.I.T. KGP}} - |\nabla_b J_s|)$$

$b(0) > 0$ otherwise arbitrary.

$$b(k+1) = b(k) - \eta \frac{1}{2} (\nabla_b J_s - |\nabla_b J_s|)$$

At every iteration I can have $b(k+1)$ which is same as the same gradient descent procedure

$$b(k+1) = b(k) - \eta \frac{1}{2} (\nabla_b J_s - |\nabla_b J_s|).$$
 So, here you find that because to obtain $b(k+1)$, we are

subtracting from $b(k)$ this quantity $\eta \frac{1}{2} (\nabla_b J_s - |\nabla_b J_s|)$, so if gradient of $\nabla_b J_s$ is positive.

Then what we are able to ensuring that this total quantity will be equal to 0 where as if gradient of $\nabla_b J_s$ is negative.

So, here it is minus, here it is also minus, so I get a negative quantity which is twice of $\nabla_b J_s$.

So, to nullify this factor 2, you incorporate this half and, here it is negative here also it is negative, so overall you will get a positive term. So, effectively what I am doing is to obtain $b(k+1)$ from $b(k)$, I am adding a quantity I am not subtracting any quantity from $b(k)$ to get $b(k+1)$. So, that insures that you are $b(k+1)$ can be utmost greater than $b(k)$, it can never be less than $b(k)$.

So, by this step we are ensuring that none of the elements of the margin vector b will be reduced at a step of iteration and, since we are ensuring that it will not reduce in any steps of iteration. So, the possibility of having a component of margin vector with b which is less than 0 will not arise. So, our iteration algorithm on b will became as before that we will have a $b(0) > 0$, but otherwise arbitrary.

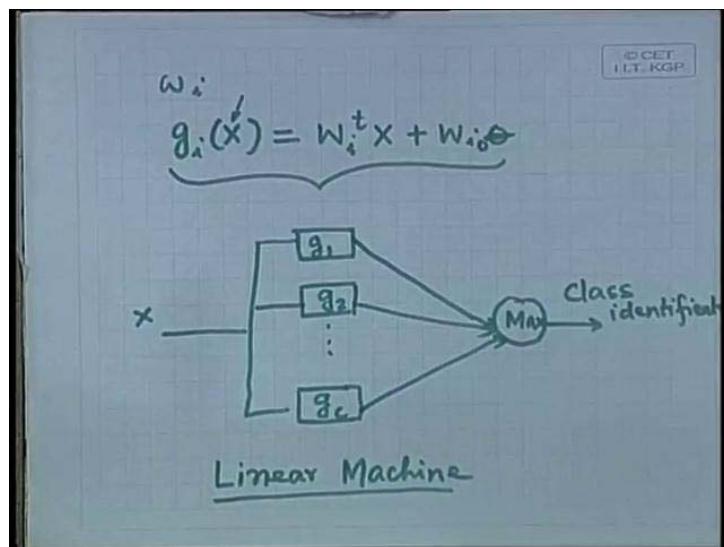
It is magnitude of every component of the gradient vector, what I am saying is every component it is not modulus. So it is a vector.

It is a vector, yes it is a vector only the positive components.

So what the iterative algorithm will be something like this that $b(0)$ will be greater than 0 which is initially chosen. But, otherwise it is arbitrary and at every step this $b(k+1) = b(k) - \eta \frac{1}{2} (\nabla_b J_s - |\nabla_b J_s|)$. So, once I have a particular value of b , I can use that b to compute the value of a , either making use of pseudo inverse or following. The another iterative algorithm of obtaining a by using the gradient descent procedure with gradient of J_s with respect to a that is similar to what we have discussed in our last class.

And by doing this because we are ensuring that b will always be greater, will always remain greater than 0 it is quite likely that it will even for linearly separable cases. We will get a solution vector which will actually linearly separate between the two sets of samples belonging to two different classes. Now, all this different criteria functions that we have discussed so far whether it is perceptron criteria, or relaxation criteria or this minimum squared error criterion. In all these cases, we have assumed that we have just two different classes ω_1 and ω_2 and our criteria was that if $a^t y > 0$, the sample belongs to one class, if it is less than 0 it belongs to another class or if $a^t y > b$ it belongs to one class, if $a^t y < b$ it belongs to another class, right? Now, the question is how to generalize this in a multi category case because in all practical situations will have multiples number of classes, not that we will have only two classes, and that is what the essence of linear machine that we have said earlier.

(Refer Slide Time: 29:12)



That is for every class ω_i we have a discriminant function defined as $g_i(X)$ where this $g_i(X)$ is nothing but $W_i^T X + W_{i0}$, right? So, for given any unknown sample x , I have to compute this discriminant function, for all values of i , i varying from 1 to c , if c is the number of classes. Then for whichever i , $g_i(X)$ is maximum we have to assign or we have to classify X to that particular class, so we have something like this that we have this g_1, g_2 like this.

If there are c number of classes we have this discriminant function g_c then given a vector X , the vector X will be fed to the discriminant function of all the classes then I have a maximum selector all these are outputs come to this max selector. And then max selected depending upon whichever discriminant function gives you the maximum value this gives you the class identification, right? This is what we have said it is a linear machine and we have seen earlier that following the Bayes decision theory if I know what is the parametric form of the probability functions, of the samples belonging to different classes.

Then following Bayes minimum error classification or minimum risk classification we can obtain the classifiers or we can obtain the discriminant functions for different classes. And then we have discussed about different situation when this discriminant function can be a linear discriminant function or when the discriminant function will be a quadratic discriminant function, right?

But, now we are talking about the non-parametric cases, that is we do not know what is the probability of the density function of the samples belonging to different classes. So, whether it is possible to have an algorithm like this gradient descent procedure to have this sort of linear machine which can classify a sample to one of the c different classes. So, here we find that our condition is something like this.

(Refer Slide Time: 32:33)

That if I get that $g_i(X)$ where X is the sample vector, $g_i(X)$ is the discriminant function corresponding to the class ω_i . So, if $g_i(X) > g_j(X)$ for all $j \neq i$, this indicates that X belongs to class ω_i because $g_i(X)$ is maximum among all the discriminant functions. So, this indicates that the sample X belongs to class ω_i , right? or in other words if I expand this it simply becomes $a_i^t y$, where this y is the augmented vector X by adding 1 to it.

So $a_i^t y > a_j^t y$, again for all $j \neq i$, so that is the condition that this sample X or sample y augmented sample y , that belongs to class ω_i . And since, we are having c number of classes that is i and j that can vary from 1 to c , so you find that for a particular value of i when I have this condition that $j \neq i$ for all values of j . So, this j will assume every value from 1 to c except i that means j can have $c - 1$ number of values, so that one which is left out that is equal to i .

So, this inequality that $a_i^t y > a_j^t y$ for all $j \neq i$ this is actually equivalent to $c - 1$ number of inequalities is that okay? So in all those inequalities I have to have this particular condition true, that suppose $i = 1$. So, $a_1^t y > a_2^t y$, $a_1^t y > a_3^t y$, $a_1^t y > a_4^t y$ and so on up to $a_1^t y > a_c^t y$.

Then only I can say that this sample y or the corresponding sample X that belongs to class ω_1 , so this equation actually corresponds to for all values of j for $c - 1$ number of inequalities. So, whether it is possible to have a unified approach to take care of this $c - 1$ number of inequalities simultaneously and give me c number of weight vectors a_1, a_2, a_3 up to a_c , so over here you find that.

(Refer Slide Time: 36:31)

$$i=1$$

$$a_i^t y > a_j^t y ; j=2,3,\dots,c$$

$$c\text{-dimensional wt. vector } \alpha$$

$$\alpha = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_c \end{bmatrix}$$

Let us take a specific case that $i = 1$, so I want that $a_1^t y > a_j^t y$, for j varying from

2, 3 up to $c - 1$ later on we will generalize. So, I take a particular value of $i = 1$, so you find that all these $c - 1$ number of inequalities can be put in a unified approach if I take a $c \times \hat{d}$ dimensional weight vector.

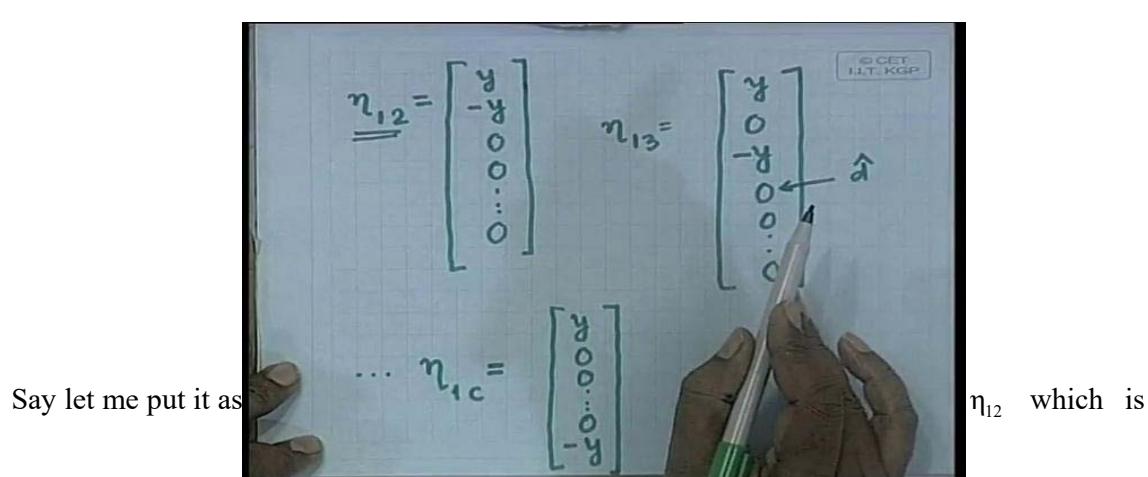
The total number of equalities is $c - 1$.

So all these inequalities can be put in a unified form in a single expression. If I consider a $c \times \hat{d}$ dimensional weight vector I am taking the same terminology which we used earlier that my weight vector is actually d dimensional. So, after appending 1 it becomes $d + 1$ dimensional, so that $d + 1$, we are talking we are taking as \hat{d} . So, I take a $c \times \hat{d}$ dimensional weight vector let us say this weight vector is α , now what is this α ? This α will be nothing but

$$\alpha = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \dots \\ a_c \end{bmatrix}, \text{ this is my weight vector } \alpha.$$

Each of these weight vectors say a_1, a_2, a_3 up to a_c each of them are actually \hat{d} , I have c number of such weight vectors. So, that overall dimension of this vector α is $c \times \hat{d}$, So this overall vector that becomes $c \times \hat{d}$ dimensional. So, we can think that these particular weight vectors if I have this condition that $a_i^T y > a_j^T y$ for all values of j varying from 2 to c . So, I can equivalently say that effectively this $c \times \hat{d}$ dimensional weight vector α will properly classify all the $c - 1$ number of feature vectors, where the feature vectors will be of the form.

(Refer Slide Time: 40:08)



$$\text{nothing but } \eta_{12} = \begin{bmatrix} y \\ -y \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}, \quad \eta_{1c} = \begin{bmatrix} y \\ 0 \\ 0 \\ 0 \\ \dots \\ -y \end{bmatrix}. \text{ Where this 0 actually means that this is a column vector}$$

of dimension \hat{d} , y is obviously a column vector of dimension, \hat{d} . So, as there are c number of such vectors, every vector having a dimensionality of \hat{d} , so the overall dimension of these vectors η_{12} or η_{13} up to η_{1c} each of them are of dimension $c \times \hat{d}$, is that okay?

(Refer Slide Time: 42:08)

$$i=1$$

$$a_1^T y > a_j^T y$$

$$c \hat{d} - \text{dimens}$$

$$\alpha = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_c \end{bmatrix}$$

$$\eta_{12} = \begin{bmatrix} y \\ -y \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \eta_{13} = \begin{bmatrix} y \\ 0 \\ 0 \\ 0 \\ \vdots \\ -y \end{bmatrix}$$

So, this inequality that $a_1^T y > a_j^T y$ for j varying from 2 to c , it is equivalent to say $\alpha^T \eta_{1j}$, is it not? if I put these two side by side. So, let us take these two vectors side by side α and η_{12} ,

$$\eta_{12} = \begin{bmatrix} y \\ -y \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}. \text{ So, if I take } \alpha^T \eta_{12} \text{ that computation will be equivalent to } a_1^T y - a_2^T y \text{ rest of the}$$

components in η_{12} are 0.

So, that corresponding vector multiplication will have null effect will not have any effect on the result is that okay? So if I take $\alpha^t \eta_{12}$ that is simply $a_1^t y - a_2^t y$, right? Similarly, $\alpha^t \eta_{13}$ that will be simply $a_1^t y - a_3^t y$, is that okay? Similarly, $\alpha^t \eta_{1c}$ it will be a simply $a_1^t y - a_c^t y$. So, this condition that $(a_i^t y - a_j^t y) > 0$ for all the values of j for y to be classified to class ω_1 . That is equivalent to saying that this weight factor α properly classifies all these $c - 1$ number of feature vectors.

Where these feature vectors are actually generated from the original feature vector y , so you find that the overall data that we are generating over here that increases dramatically. And that is quite obvious because from a 2 dimensional, from a 2 category case we are moving to c category case, so your computation obviously will be c fold. So, for every feature vector y , I am generating $c - 1$ number of feature vectors, right? Where the dimensionality of the feature vectors each of this feature vectors is also being incremented by a factor c initially vector y is of dimensionality \hat{d} .

Now, each of them are of dimension $c \times \hat{d}$, so the dimensionality is increasing by the number of classes that we have and the number of vectors for every original vector is also being increased by a factor $c(c - 1)$. So, the amount of data that we are generating that becomes quite high, so this is what we have considered for a particular case by assuming $i = 1$.

(Refer Slide Time: 45:49)

$$\eta_{ij} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ y \\ 0 \\ \vdots \\ 0 \\ -y \\ \vdots \\ 0 \end{bmatrix}$$

$$\alpha^t \eta_{ij} > 0 \Rightarrow \alpha_i^t y - \alpha_j^t y > 0$$

In general, a feature vector η_{ij} will be something like this that I will have this vectors having component i^{th} component will be equal to y , j^{th} component will be $-y$. Then for this 0 again,

where this is my i^{th} component and this is my j^{th} component is that okay? And this $\alpha^t \eta_{ij} > 0$ that is equivalent to if you follow from here it will simply be $(a_i^t y - a_j^t y) > 0$. So, if this condition is satisfied for all values of $j \neq i$ then we say that the sample y actually belongs to class ω_i , it does not belong to class ω_j .

So, when I have this sort of formulation that $\alpha^t \eta_{ij} > 0$ for every such inequality I actually have the weight vectors corresponding to discriminate functions of two different classes ω_i and ω_j involved in it and for proper classification of any sample, a_i must have that it $\alpha^t \eta_{ij} > 0$ that means $(a_i^t y - a_j^t y) > 0$. So, if I find that this condition is not satisfied for any value of y , for any sample y then I have to modify actually two weight vectors not only a_i but, I will also have to modify a_j because both of them are responsible in earlier case because the 2 class problem, so I had a single weight vector, so I have to decide whether that to modify two weight vector. If it is to modify in which way it has to be modified, now I have 2 weight vectors involved one for the discriminate function of class ω_i and the other one for the discriminate function of class ω_j . So, any time any sample is misclassified by α , I have to what modify both these weight vectors a_i and a_j involved that particular η_{ij} , so the weight updation or weight vector updation rule in this case can be simply like this initially I assumed

(Refer Slide Time: 48:56)

$$a_i(0) \leftarrow \text{arbitrary for all } i$$

$$a_i(k+1) = a_i(k) + y^k$$

$$a_j(k+1) = a_j(k) - y^k$$

$$(a_i(k+1) = a_i(k)) \quad l \neq i \& l \neq j$$

$$\alpha^t \eta_{ij} > 0$$

Kesler's Construction

that $a_i(0)$ is arbitrary for all i , that means for every class I assume an arbitrary weight vector initially, right? And then for any vector which is misclassified. So, with the help of this initial

weight vector I have to form initial α which is just concatenation of all these different weight vectors. Using that I have to go for classification of the samples where the samples to be modified in this form whenever I know that our sample belongs to class ω_i .

So, for every such i I have to generate η_{ij} , but the i^{th} component will be equal to y and the j^{th} component will be $-y$. So, I have to generate η_{i1} , I have to generate η_{i2} , I have to generate η_{i3} , I have to generate up to η_{ic} . So, for each of these samples I have to generate $c - 1$ number of sample vectors and I have to make, I have to ensure that each of those modified sample vectors are properly classified by the corresponding α .

If any of those samples is not properly classified by α then I have to modify both a_i as well as a_j , so that weight updation rule will be $a_i(k+1) = a_i(k) + y^k$. Where this is the sample which is misclassified and this is what we have obtained using our perceptron criteria, this expression is same. So, we are incrementing a_i in particular direction, a_j has to be incremented in the negative direction because we want to penalize a_j .

Whereas we want to reward the a_i and $a_j(k+1)$ for this the weight updation rule will be this will be simply $a_j(k+1) = a_j(k) - y^k$, so you find that only two weight vectors. We are updating one is a_i and a_j for every η_{ij} which is misclassified and all other weight vectors will say that $a_l(k+1)$ will remain as $a_l(k)$ where $l \neq i$ and $l \neq j$. So, only i^{th} weight vector and the j^{th} weight vector these 2, we are modifying modification of i^{th} vector follows the same updation rule that we have obtained in the perceptron criteria modification of the j^{th} weight vector a_j similar updation rule.

But, it is instead of incrementing we are decrementing it, so a_i is actually rewarded a_j is penalized all other weight vectors other than a_i and a_j , they remain unchanged because these weight vectors really do not take part to tell us, whether $\alpha^T \eta_{ij} > 0$ or not these weight vectors do not tell us anything about this, only the weight vectors say a_i and a_j decide whether this condition will be satisfied or not is that okay? So, we are keeping $a_l(k+1) = a_l(k)$ for all values of l , which is not equal to i and not equal to j and this kind of construction because we find that every time from a_i we are constructing this weight vector α .

And for every sample y we are constructing η_{ij} this kind of construction is what is known as Keslers construction, yes one is sufficient. But, there is nothing wrong if you do both of them together because our aim is that we want to have $(a_i^T y - a_j^T y) > 0$, so increment one reduce the other one. So, that this condition is reached quite fast, is that okay? If you do not do this then the number of iterations that will be taken will be more nothing else, so let us stop here today. Thank you.

Pattern Recognition and Image Understanding

Prof. P.K. Biswas

Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology, Kharagpur

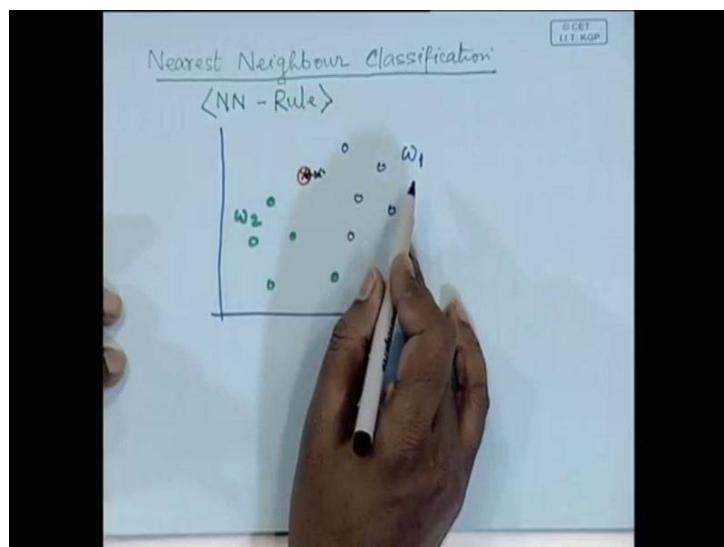
Lecture no. # 17

Neural Network for Pattern Recognition

Good morning, so today we are going to start our discussion on use of neural network for pattern recognition, so far what we have discussed. Initially we started discussing about the Bayes decision theory or you have seen that the classifiers are to be designed using the probability density function of that any samples. Later on we have gone to non-parametric classification techniques where we have seen that different types of criteria functions like perceptron criteria, relaxation criteria or

Minimum squad error criteria which are to be designed, which are to be minimized for designing of linear classifier, so before I actually go to this use of neural network for pattern recognition. Let me detail about two more techniques of non-parametric pattern classification, pattern classification, so one of them non parametric technique for the pattern classification is called nearest neighbor rule.

(Refer Slide Time: 01:02:27)

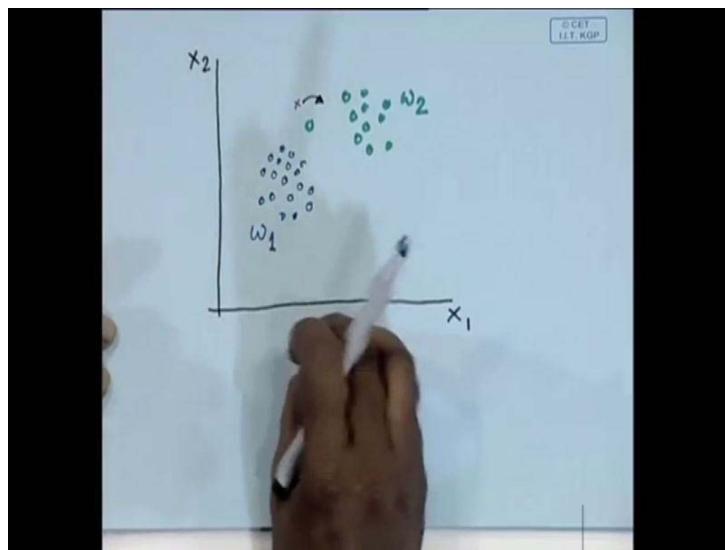


Nearest neighbor classification or it is also called NN rule, so when you use this nearest neighbor classification technique what you do is instead of trying to find, design any classifier or any discriminant function. For any of the classes what I have is, I have a set of training samples the training samples coming from different classes, so if it is 2 class problem I have training samples coming from two different classes. For a c class problem, I have training samples coming from c different classes. In nearest neighbor classification, what we do is whenever an unknown sample is given we have to classify this unknown sample to one of the classes for which I have the training samples, so what I will simply do is I simply find out the nearest training sample which is nearest the unknown sample.

That means you try to compute the distances from the unknown sample to all other training samples feature provided and you find out the minimum of those distances. So, whichever sample is nearest to the unknown sample, or the distance is minimum I know what is the class belongingness of that particular sample of that training sample which is nearest with the unknown sample. So, accordingly you classify this unknown sample to that class of the training sample which is nearest. So, I have situation like this see if I have a set of training samples say these are the training samples which belong to one class and suppose these are the training samples that belong to another class.

Suppose I have an unknown sample which is somewhere over here, so if these samples actually belong to class ω_1 and these are the samples which belong to class ω_2 , I just find out that which sample is nearest to this unknown sample. So, you find that the distance between this sample and this sample is minimum, so the nearest sample is this one and this sample belongs to class ω_1 . So, I simply classify this unknown sample to class ω_1 , but this particular approach has a problem. The problem in the sense that is a sample nearest does not mean that it is most probable class, so I can have a situation something like this.

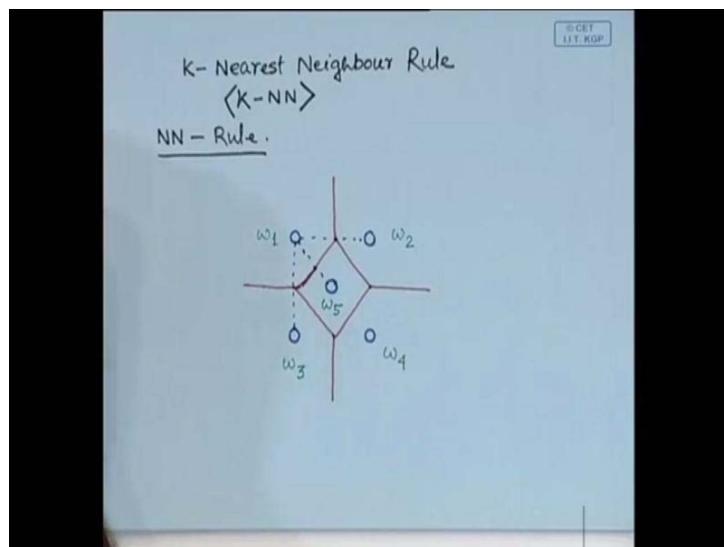
(Refer Slide Time: 01:05:54)



So, I put again I take 2 dimensional feature vectors having the feature component x_1 and x_2 and the samples are for one class it is something like this, and for another class the samples may be something like this. So, I said that these are the samples, training samples taken from class ω_1 and these are the training samples taken from class ω_2 and in over here if the unknown sample is somewhere at this position. So, you find that as before this unknown sample will have a minimum distance from this training sample which belongs to class ω_2 . So, I will classify this sample to belong to class ω_2 , however you find that here I have a single point, a single sample from class ω_2 .

Whereas, I have more number of samples from class ω_1 which may actually or which would actually influence the classification of this unknown sample. But, without looking at the density of the training samples, we are blindly classifying this unknown sample to a class which is nearest to one of the training samples. So, this sort of nearest neighbor rule does not take into consideration the maximum probability of a class. So, a solution to this is that instead of taking a single point which is nearest to the unknown sample, you take a pre specified number of points which is more than 1. Then try to classify this unknown sample to one of the classes based on a voting mechanism which is given by those training samples where the number of training samples is pre specified.

(Refer Slide Time: 01:08:38)



So, accordingly we have a rule which is K nearest neighbor rule or K-NN classification rule, so over here what you do is instead of taking single point which is nearest to the unknown sample. You take K number of points, K number of training samples which is nearest to the

unknown sample and these samples, these K number of samples may come from single class may come from multiple number of classes and out of this K nearest neighbors or K nearest training samples that we get. You try to find out that which class is mostly occurring, is maximally occurring in that K number of samples or K number of training samples. So, whichever class is maximally represented in that K number of samples you classify these unknown samples to that particular class so considering that.

Sir, how will that is K number of samples from ω_1 class or ω_2 class.

It may be from any class; I just want that I have to have K number of samples which are nearest to the unknown sample I do not bother about whether it belong to class ω_1 or it belongs to class ω_2 . Once I have that K number of training samples from whichever class it is without any class consideration you take K number of training samples which are nearest to unknown samples. and within that K number of nearest samples, now you find out that from which of the class I have got maximum samples.

So, which class is maximally represented in that K number of samples, so whichever class is maximally represented in that K number of samples, I classify the unknown sample to that particular class. So coming to this same example. Say, this is the position of the unknown sample and these are the samples which are from class ω_1 , these are the samples from class ω_2 , right? So, what I do is I try to incorporate K number of samples which are nearest to this, so for doing that what I have to do is I have to find out the distance of the training samples. Whether it is coming from class ω_1 or it is coming from class ω_2 irrespective of that I have to find out K number of samples which are nearest to this unknown sample.

So, suppose I decide say value of K = 5, let us take any value, so I will take 5 samples which are nearest to this unknown sample, obviously this is nearest to this training sample. So, this will be included in that 5 samples among the remaining suppose this, this, this, these 3 of the samples or may be this one, say these 4 of the samples which are also nearest to this unknown sample. So, I have selected 5 number of samples, out of 5 you find that there are 4 samples which are coming from class ω_1 and I have only one sample which is coming from class ω_2 .

So, it is class ω_1 which is maximally represented in those 5 training samples, so I have to classify this unknown sample x, to class ω_1 rather than class ω_2 . So, find the difference that if I go for simple nearest rule that is I find out one training sample which is nearest to the unknown sample. In that case, this unknown sample is classified to class ω_2 whereas as if I take 5 nearest samples and out of those 5, I find out which class is maximally represented.

Here, it is class ω_1 which has 4 samples out of the 5 whereas class ω_2 has got only 1 sample out of 5, so accordingly this unknown sample is classified as class ω_1 rather than class ω_2 . Now, naturally if I go for the nearest neighbor classification where value of $K = 1$, the error rate or the errors bound will obviously more than error bound that you get by using Bayes decision or Bayes minimum error classifier because in case of Bayes minimum error classifier we classify an unknown sample to a class for which the probability of a particular class is maximum.

And in case of a single, 1-NN or nearest neighbor classification rule, I am just taking a sample which is nearest to the unknown sample. That particular sample training sample may be just I may not have other training samples nearer to this belonging to the same class, so accordingly the probability of that particular in that region is very less. So, naturally the error that you get in case of nearest neighbor rule is more than the error that you get using Bayes minimum error classifier where as if I go for this classification where I am telling to find out that which class is maximally represented in those K number of training samples and it can be shown that if value of K as well as value of N by K .

Where N is the total number of samples is very large in that case the error bound of K-NN classification approach is the error bound of Bayes minimum classification. The reason is very simple that if K is very large and also N by K is very large, then the number of samples within a small region that gives you good approximation of the probability density function of the samples belonging to different classes. So, in the limit when K becomes very large as well as N by K becomes very large the error bound given by K-NN classifier approach is that of the Bayes minimum error classifier.

So now following this if I go for simply nearest neighbor rule, I should be able to find out what is the decision boundary between different classes. So, let us take a very simple example something like this say I have a number of samples say this is one sample, this is one sample, this is one, and this is one and I may have another samples somewhere over here. Let us assume that this sample belongs to class say ω_1 , this sample belongs to class say ω_2 , this sample belongs to class ω_3 , this one belongs to class ω_4 and say this one belongs to class ω_5 . Now, following nearest neighbor rule of I want to find out what is the decision boundary which separates all this different classes.

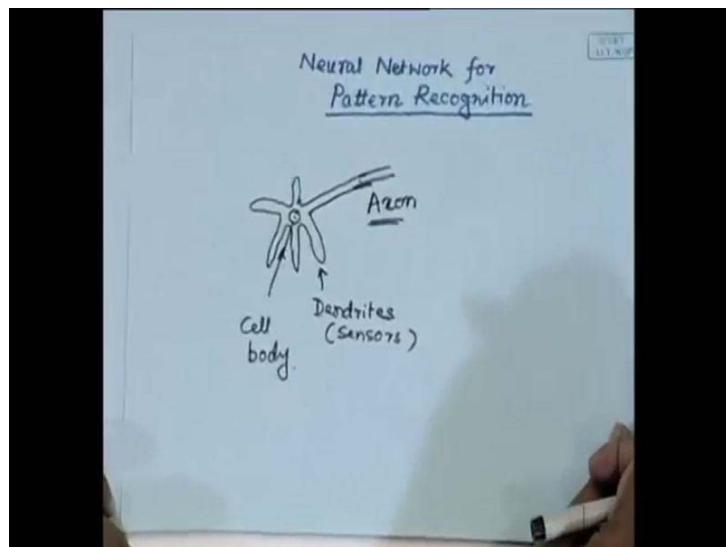
So, you find that whenever I go for nearest neighbor rule the nearest neighbor classifier is nothing but a minimum distance classifier because the class to which I am classifying this

unknown sample I have one sample from that class whose distance from the unknown sample is minimum, so naturally this is nothing but a minimum distance classifier. If I have a minimum distance classifier then the decision boundary between the two classes where I have representatives of 2 classes, then the decision boundary is nothing but a perpendicular bisector or orthogonal bisector of the line joining those 2 examples samples or those 2 representatives.

So, following that you find that the decision boundary between class ω_1 and ω_2 will be an orthogonal bisector of the line joining these two samples. So, the decision boundary that I have is simply this and it will be extended on this side as regards ω_1 , ω_2 and it will be extended on this side. Now, coming to the decision boundary between class ω_1 and ω_5 , that will also be an orthogonal bisector of the line joining ω_1 and ω_5 , so this decision boundary will be simply this. Similarly, the decision boundary between classes ω_1 and ω_3 , that will be orthogonal bisector of this line.

So, this particular decision boundary, sorry it will be something like this will be, this one in the same manner the decision boundary between ω_3 and ω_5 will be this between ω_5 and ω_4 . It will be this between ω_2 and ω_4 the decision boundary will be this, between ω_3 and ω_4 the decision boundary will be this. Between ω_2 and ω_5 the decision boundary will be like this, is it okay? So if I follow the nearest neighbor rule, I can very easily find out the decision boundary among different classes for which I have the training samples, is it okay? So, these are the different non parametric techniques for classifier design or for classification of unknown samples to one of the known classes.

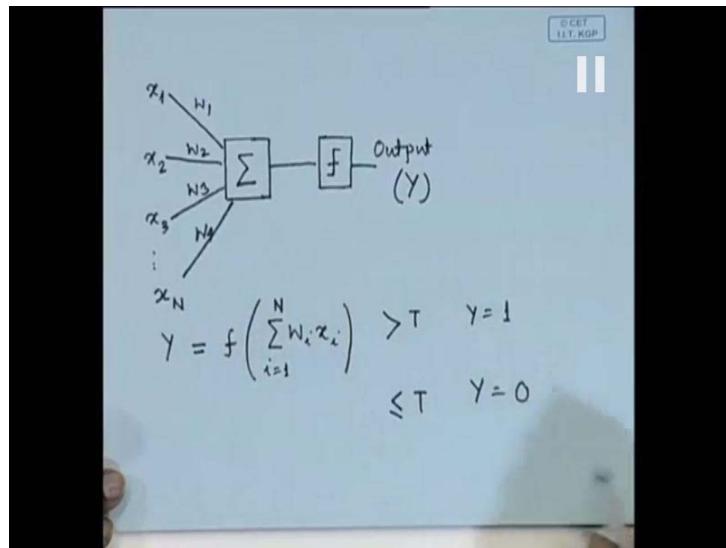
(Refer Slide Time: 01:20:54)



So, now let us start our discussion on use of neural network for pattern recognition, so when we talk about a neural network, neural network is nothing but a computational model which tries to imitate the human brain. So, when we talk about human brain or human nervous system this actually consists of a number of numerous nerve cells and in the nerve cell. The nerve cell is something like this if you draw any particular nerve cell it goes like this where this whole thing is a cell body, these are what are called dendrites or sensors inside I have the cell body and these fibers, these are actually called axons.

So, these dendrites of the sensors they actually sense different stimulus and that signal is carried to the brain by these axons. Now, depending upon what kind of stimulus or what kind of sensing this cell body has done, the brain takes a particular axon decides an axon and sends that information to or different muscles through a set of such neurons. And then we take certain axon by activating corresponding muscles, so when we talk about the neural network the neural network also tries to imitate a functionality something like this. So, our neuron model for machine intelligence or for artificial intelligence will be something like this.

(Refer Slide Time: 01:23:29)



Say I have to have something like a cell body and then I have to have a set of inputs which act as dendrites and then I have to have an output which will act as the axon, right? So, what I have is I give the feature vector as input to this neuron and our feature vectors are having different components x_1, x_2, x_3 up to say x_N . If I have N dimensional feature vector, then this cell body performs a weighted summation of these different feature components and weighted sum is feed as an input of a nonlinear function. In most of the

cases this nonlinear functions are some sort of threshold function or it may be account inverse nonlinear function as well as different types of neural networks used, different types of non-linearity.

This is our output let us call it as Y and because this will be this particular cell body will compute the weighted summation of this different feature components. So, I have to have the corresponding weights, so those weights are say w_1, w_2, w_3 and w_4 , so what I have at the

output of this is just like $\sum_{i=1}^N w_i x_i$, where i varies from 1 to N . So, on this I have a nonlinear

function F and there is one output Y if I use this non linearity as a simple threshold function then what I will have is if this nonlinear output. If it greater than threshold, I set Y to 1 and if this is less than or equal to threshold I set Y to 0, so having this kind of functionality you find that it has a very close similarity with one of the linear discriminators that we have designed earlier.

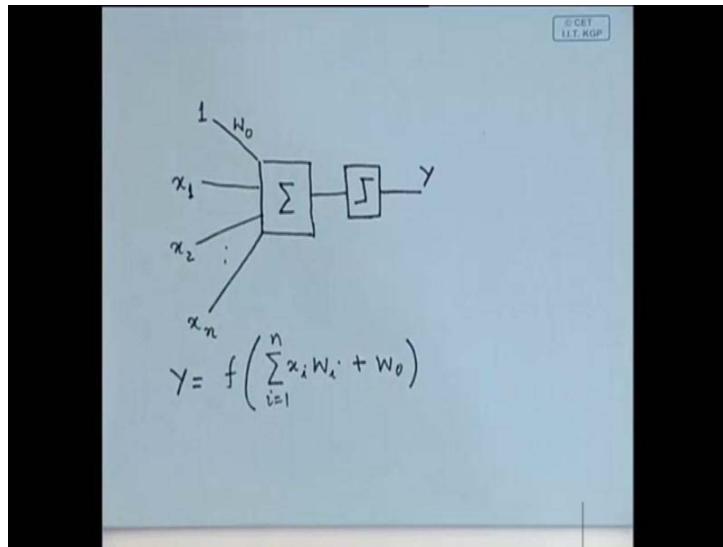
That is, we have said that $g_i(X) > 0$, it belongs to some class if it is less than 0 it belongs to some other class and if I have a linear discriminant function that $g_i(X)$ is nothing but this

$\sum_{i=1}^N w_i x_i$. Where my feature vector is N dimensional and if this function is greater than

threshold I said $Y = 1$ indicating that this X belongs to one class. If this is less than or equal to threshold, I set $Y = 0$ indicating that this X belongs to the other class, earlier we have said that if it is equal to 0. Then X actually belongs to the boundary, I can also put that condition over here if this function, weighted sum becomes equal to 0, I will not take any action, whereas if it is less than 0, then only I will set $Y = 0$.

So, accordingly this function will be different or threshold function will be equal to will be different where if this weighted sum becomes equal to 0, then I will not take any action. However, in case of linear discriminant function we have seen that before you go for designing the classifier you have to append one to the feature vector, so we get a modified feature vector and this purpose of this modified feature vector is that I can incorporate a bias weight which is different corresponding to different classes, that is W_{i0} , I can also incorporate the same bias within a neural model what I do is I can simply.

(Refer Slide Time: 01:28:52)



So, let me draw the same neural model, where I have this addition and I have different inputs, one of the inputs I can put equal to 1 constantly and the weight given to this corresponding input can be w_0 . Whereas, to other inputs I have the different components of the feature vectors that is x_1, x_2 up to x_N the other functionality remains the same I have this non linearity and then I have get this output Y , is it okay? So, where here you find that output of this is

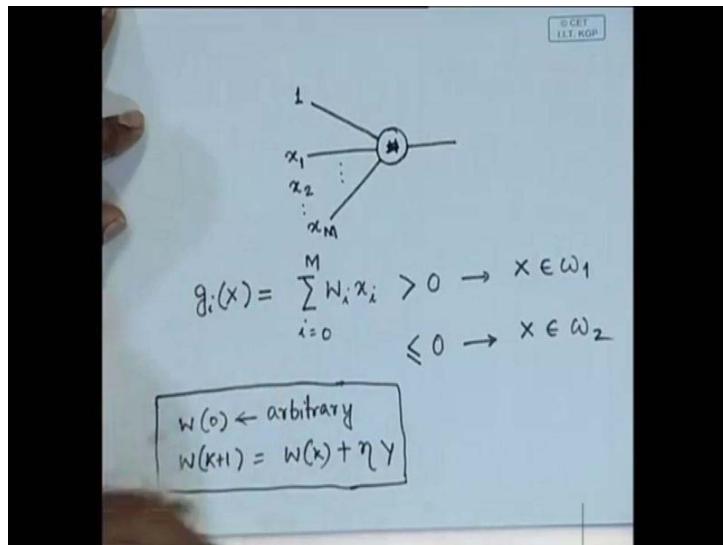
$$\sum_{i=1}^N w_i x_i + w_0 \text{ and on top of that I put this non linearity which gives me the output } Y.$$

Again I can have similar kind of decision that Y will be set equal to 1, if this term is greater than threshold it will be set to 0, if this term is less than or equal to threshold and accordingly I can go for classification. So, find that in both the cases the model that we have is closely similar to what we have followed in case of linear discriminant function. We have seen that they have different approaches of designing linear discriminant function like perceptron criteria, relaxation criteria, mean square error criteria, minimum square error criteria and so on.

So, what I have to do whenever I go for designing such a kind of neuron is that I have to select the weights W , or w_i properly which was the purpose of having different types of criteria functions. We wanted to have different weight vectors, a weight vector which gives you proper classification between 2 or more classes. So, here also the major task is to decide about this connection weights and that is what is done during training or learning of neural network and as it is very close to what we have got in case of perceptron criteria.

A neural network formed out of such neuron models they are also called perceptrons, so in the simplest case I can have a single output single layer perceptron which is nothing but a single neuron like this, so in case of a single output single layer perceptron, the neural network is simply this.

(Refer Slide Time: 01:32:29)



So, now all this function we put in the form of a node these 2 taken together can be represented by a node, so when I say a node the node includes both this adder which performs the weighted summation as well as this non-linear unit. So a node gives me this particular output for a given input, so a single output single layer perceptron is a single node something like this.

So, which has number of inputs and one output I can say that, so this is my neural node, let me not use M because in many cases we are using N to represent the dimension if I have a bias 1. One of the inputs will be equal to 1 and the other inputs will be the different component of the feature vector up to say x_M , let us say. And here, my $g_i(X)$ what this node compute is simply $\sum_{i=0}^M w_i x_i$, i varying from 0 to M and we say that if this is greater than 0 then we say that the vector x belongs to class ω_1 .

If this is less than or let us include equal also if it is less than or equal to 0, we say that the sample X belongs to class ω_2 . Now, in case of perceptron criteria we have said that this weight updation rule was that initially we decide $W(0)$ to be arbitrary when we had the perceptron criteria initially decided that $W(0)$ the weight, initial weight is chosen arbitrary then at every iteration stage the weight was chosen as $W(k+1) = W(k) + \eta Y$, this is what we had in case of perceptron criteria.

Now, similarly in this case also we have to have some weight updation rule which will update the weights of the input links and this weight updation has to be done using the training samples that which is provided. So, we have to make use of all the training samples feature that are provided then keep on updating the weights until and unless we get the correct output. Since, the training samples we know what are the class labels of the training samples then for a training sample, I know what should be the corresponding output.

Say if the, if a training sample x belongs to class ω_1 , I know that output should be equal to 1, if the training sample belongs to class ω_2 , I know that output of the neuron has to be equal to 0. So, what I can do is, I can find out what is the difference between the target outputs if the actual output, I call it as target output. That is what is expected and because of improper weights, improper connection weights I may get an output which is actually the target output.

So, the difference between the target output and the actual output is the error, right? And all the weight updation rules connection which updates the connection weights in a neural network it tries to minimize that output error because I know that what is the exact output is, but should be my target output and what output I actually get. So, I can compute what is the error then try to minimize that error by modifying the connection weights, so over here you find that for a given input X , for a given training sample X .

(Refer Slide Time: 01:37:44)

$$D_p = \sum_{i=0}^N w_i x'_i$$

$$e = D - d$$

$$E = \frac{1}{2} \sum_{p=1}^N (D_p - d_p)^2$$

$$\frac{\partial E}{\partial w_i} = (D - d) x_i$$

$$w(k+1) = w(k) - \eta (D - d) x_i$$

If I know that my output the target output is say d whereas the actual output that is get is nothing but $\sum_{i=0}^N w_i x_i$, where i varies from 0 to N . So, this is my target output, this is the actual output that I am getting, so the error e is nothing but $D - d$, I can have sum of squared error from this. So, I can say that sum of squared error, let me write it like $E = \frac{1}{2} \sum_{p=1}^N (D_p - d_p)^2$, let me say that this is for a p^{th} sample, where p varies from 1 to N , where N is the total number of training samples, so every sample I have an error take the summation of that error, summation of square the error which gives you the sum of squared error out of this if you find that the sum of squared error is actually a function of W , where W is the connection weight, right?

So, the simplest approach that can be taken is you try to minimize this E by differentiating this with respect to W and as W has got $M+1$ number of components. So, by differentiating this with respect to W and equating that to 0, I will get $M+1$ number of simultaneous linear equations. You solve that simultaneous, that set of simultaneous linear equations which will give you the values of different components of W .

Your set of different values of was your connection weights and your neural networks should give you the classification result whenever an unknown sample is fed to the input of the neural network. However, as we have done in our earlier case that instead of trying to solve this number of simultaneous equations we can have sequential algorithms or iterative algorithm which will iteratively update the weight vector or the connection weights. And for that what we have to do is we have to go for gradient decent procedure. X is the input vector, yes x_i is different, you mean to say that I should put it as like this does it really matter if it is p^{th} sample.

Yes I can put it like this also, this is the i^{th} component of the p^{th} sample, you can put it like this, so we can again follow the gradient decent procedure for that we have to take the derivative of this square error with respect to or weight vector W . So, if I take the gradient of this say $\frac{\partial E}{\partial w_i} = (D - d) \cdot x_i$, w_i which is the i^{th} component of the weight vector where this i varies from 0 to capital $M + 1$ number of components including that one that we have added as bias, so this $\frac{\partial E}{\partial w_i}$ if you compute this it will simply become $(D - d) \cdot x_i$. If you differentiate this with respect to W_i then what I get is this $D - d$, where D is the actual output that I get,

lower case d is the target output, so $D - d$ is the error that I can easily compute and then multiply with x_i .

(Refer Slide Time: 01:42:49)

$$d_p = \sum_{i=0}^N w_i x_i$$

$$e = D - d$$

$$E = \frac{1}{2} \sum_{p=1}^N (D_p - d_p)^2$$

$$\frac{\partial E}{\partial w_i} = (D - d) x_i$$

$$w_i(k+1) = w_i(k) - \eta (D - d) x_i$$

So, accordingly our weight updation rule will be $w_i(k+1) = w_i(k) - \eta(D - d)x_i$, so you find the difference between what we had in case of perceptron criteria. In case of perceptron criteria our weight updation rule was something like this that $W(k+1) = W(k) + \eta Y$ where Y is the vector. So, the entire weight vector was modified simultaneously and in this case it is that, no that differentiation you do it will come this because we are differentiating with respect to W_i that is the i^{th} component of W .

So, j^{th} component, that is $W_j x_j$ which is the independent of W_i that will become 0, so I will be left with only x_i the contribution of this x_i into this error term was $W_i x_i$.

I am not for the time being i am not considering p^{th} sample given a particular sample what is the error, so given a sample the error is $D - d$.

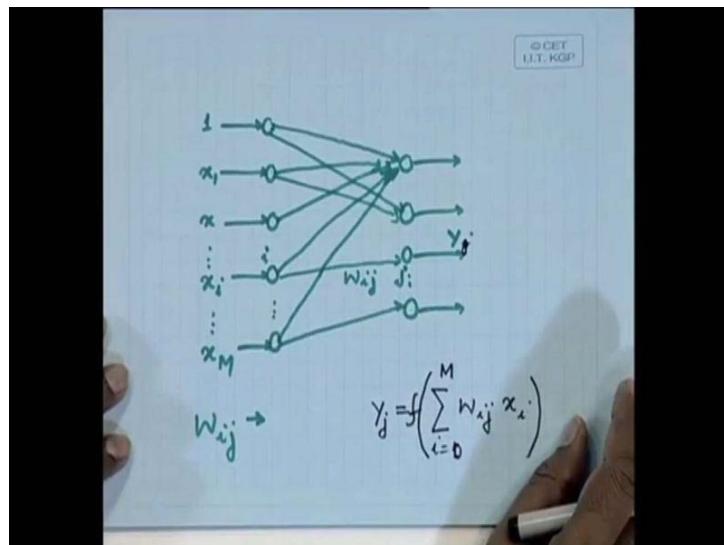
So, in that above equation there is the possibility that p^{th} sample this is for individual sample I am trying to compute what is the error for individual sample and whenever I face an error I am trying to modify the weight vector. So, you can say that this is the p^{th} sample variable, so this is what I have for an individual sample, so our training sample which is fed to the input I know what is the target output. That is what is d , I know what is the output that I am actually getting that is D , difference between these two my error, so the weight updation rule in this

case. Unlike this case, perceptron criteria where the weight updation rule was this over here the weight updation rule is $W(k+1) = W(k) + \eta Y$.

But, sorry this will be about i^{th} component because I am talking about i^{th} component only, $W_i(k+1) = W_i(k) - \eta(D - d)x_i$ where x_i is the i^{th} component is that okay? So, this is what I have in case of a single output single layer perceptron, where w_i actually represents the connection weight from i^{th} component x_i . So, this is my w_i , so for every individual component I am doing it, right? And this single output neuron, perceptron actually takes care of only 2 class problem.

If the output becomes 1, I will say that the sample belongs to 1 class, if the output belongs to becomes 0 I will say the sample belongs to another class. But, if I have multiple class problems then this single output neuron model is not sufficient I have to have a neuron, I have to have a neural network having multiple numbers of outputs.

(Refer Slide Time: 01:47:31)



So, I have one input layer where I fed in the feature vectors and I have an output layer, but the neurons in the output layer are same as the number of classes that I have the inputs are the feature components are fed to input layer which is simply passed to the output of the neural of the input layer neurons, so this is my say this is 1 then x_1, x_2 say this is x_i . Then I have this x_M , I have the connections from outputs of each of the input the neuron to the input of every neuron of the output layer.

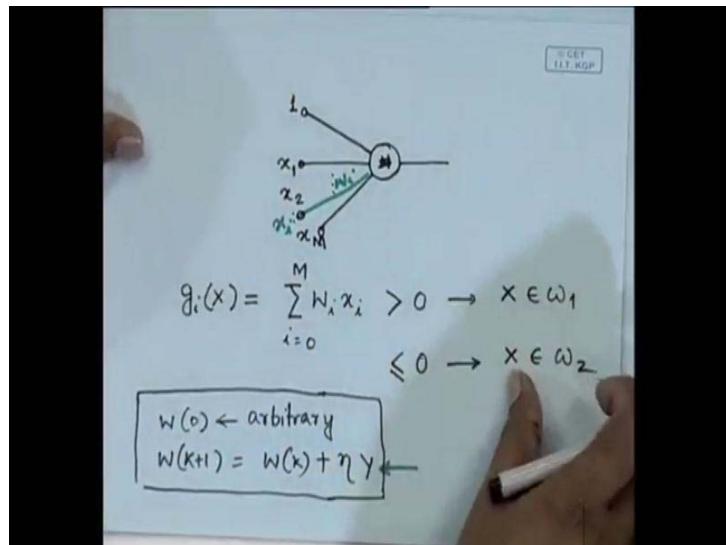
So, I have connections like this so it continues like this and these are my outputs, so I represent the connection weights as w_{ij} . So, this w_{ij} connection weight means that this is the weight of the connection from i^{th} node of the input layer to the j^{th} node of the output layer. So, this W_{ij} this represents say this may be my i^{th} node of the input layer, and this is the j^{th} node in the output layer, so this connection weight from the i^{th} node of the input layer to the j^{th} of the output layer.

This is represented by W_{ij} , so accordingly you find that if this output I say Y_j , so this Y_j will be nothing but $Y_j = f\left(\sum_{i=0}^M w_{ij}x_i\right)$, i varying from 0 to capital M then the non-linearity which is applied at the output of the output layer nodes. So, for different values of i , the individual components are being weighted by the corresponding w_{ij} , so at the output layer what I get is $W_{ij} \cdot x_i$. This also inputs the bias because I have started $i = 0$ M, so $i = 0$ means this w_{0j} this connection component will actually correspond to the bias or bias weight?

So, I have this Y_j which is a non-linear function of $\sum_{i=0}^M w_{ij}x_i$, where i varies from 0 to M in some cases as we said earlier that this non linearity can be a threshold function, in other cases the non-linearity can also be a sigmoidal function, where the sigmoidal function is.

Over here input in node is only single, it may be multiple links. In the first column of the nodes why there is single input is having, in previous case we have multiple are there.

(Refer Slide Time: 01:52:00)



You are talking about this one, here this is my, say actually each of these are actually nodes because when I am feeding this inputs where do I feed these are actually nodes which are unit again nodes. That means whatever is fed to the input that is simply passed to output when I say that the number of layers this input layer is not really specified because this comes by default.

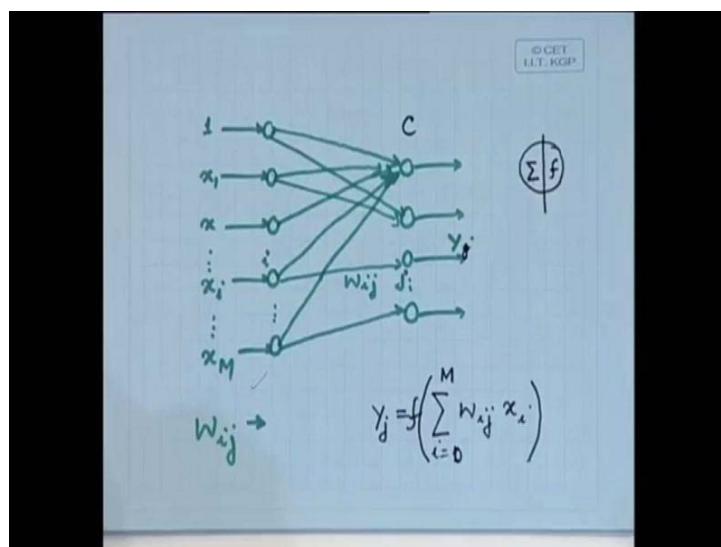
So, single input is input to these nodes

Input to this nodes, here I said that this single layer single output neural network when I say single layer apart from this input layer how many layers I have. So, that is only 1 here also it is single layer, here also it is single layer I have only the output layer input layer comes by default it is simply what I hook into the inputs.

So, whatever input you are feeding here the same input is simply crossed to the output and when these are connecting to the inputs of the output layer nodes. So, every such input or output of the input layer node that gets weighted by the corresponding connection weigh, right? So, accordingly the input to each of the output layer node that is input to the j^{th} output

layer node is this $\sum_{i=0}^M w_{ij}x_i$, I can consider that this is the input to the j^{th} output layer node.

(Refer Slide Time: 01:53:52)



This node I can break actually in 2 parts every output layer, node I can break into 2 parts one portion computes the summation and the other portion gives you the non-linearity, that is F. Coming to our initial nodes which we have defined I have an adder unit followed by a non-linearity, right? So, I can say that each of these nodes actually are broken into 2 parts, one

part computes the summation which is followed by non-linearity F that is not layering multiple layers I will come later on. So, because this question has come let me just clarify this say what we have done in case of this single layer single output neural layer network I get a linear equation of this form.

This linear equation is equation of single straight line is that okay? If the point when it is greater than 0 means a point which lies on the positive of the straight line which is given by this equation. Here, if it is less than 0 means the point which belongs to the negative side of the straight line given by this equation. So, this single layer single output neural network actually divides the feature space into 2 half spaces, one half is positive the other half is negative extending that to this single layer multiple output neural network. You find that each of these output neurons actually defines equation of a straight line is it not because this

$\sum_{i=0}^M w_{ij}x_i$, i varying from 0 to M for a particular j it gives me equation of one straight line, right?

For another value of j it gives me equation another straight line, right? So when I have, say on this output layer that if I have a c class problem I will have C number of output layer nodes every node corresponding to one class, right? So, when I have this C number of output layer nodes each of these nodes define equation of the straight line or each of them give you a linear equation which is the discriminant function for individual classes. For classification I will come later on, what I will do is I will find out that for a given input which of these output layer nodes gives you the maximum output, right?

And the sample will be classified to that particular class which gives the maximum output which is nothing but discriminant function. That we have said in other cases and coming to a linear separability because each of them are actually representing your linear equations. So, it assumes that the classes are linearly separable what you do if the classes are not linearly separable then this single layer will not be sufficient I have to go for multiple layers. So, when I go for multiple layers what I get is a non-linear boundary is actually broken into linear pieces. So, piece wise linear approximation is required, so I will come to that later on, let us stop here today.

Thank you.

Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 23
Liner Discriminator
(Tutorial)

Hello. So, in the last few classes, we have talked about different types of linear discriminators and different types of techniques for designing of linear discriminators. So, we have talked about the statistical model that is starting from the probability distribution of the samples belonging to different classes. How we can design a classifier using a Bayes decision theory? We have also talked about other type of designing techniques for linear discriminators when we have a set of samples, which are given for training of the classifier or design of the linear discriminator. And one of them we have discussed is use of perception criteria.

We have also talked about the relaxation criteria for designing of linear discriminators. We have also talked about the mean square error criteria for designing of linear discriminators. Then various such techniques we have discussed in last few classes. We have also talked about a linear machine. Where we have said that every class has a discriminant function and when you design the discriminant function, you make use of the samples for which the class belongingness is known so effectively what we are trying to do is we are trying to implement supervised learning technique.

That means training of the classifier is supervised in the sense the classifiers are trained using some samples for which the classes are known. So, you have also talked about a linear machine for every class has a discriminant function. So, for a given unknown sample, what we try to do is we compute the discriminant functional value for that unknown sample for every classes. So, whichever class gives maximum value, the unknown sample is classified to that particular class.

So, in today's class, what we will do is we will take few problems. We will take few problems and try to solve those problems. So that, your ideas of designing of such classifiers become clearer. So, let us take the first problem where we have the discriminant functions for every class. From those discriminant functions from that set of discriminant functions, we have to find out what is the class boundary between different classes, and also

what is the region in the feature space, which are allotted to different classes. So, we will take the first problem.

(Refer Slide Time: 03:43)

$$\begin{array}{c} \omega_1 \quad \omega_2 \quad \omega_3 \\ g_1(x) \quad g_2(x) \quad g_3(x) \\ g_1(x) = 10x_1 - x_2 - 10 \\ g_2(x) = x_1 + 2x_2 - 10 \\ g_3(x) = x_1 - 2x_2 - 10 \\ g_{12}(x) = g_1(x) - g_2(x) = 0 \\ g_{23}(x) = g_2(x) - g_3(x) = 0 \\ g_{13}(x) = g_1(x) - g_3(x) = 0 \end{array}$$

I assume that that I have got 3 classes say the classes are ω_1 , ω_2 and ω_3 . So, I have got these 3 different classes. For every class, I have a discriminant function that means for class ω_1 ; I have a discriminant function, which is given by $g_1(X)$. For ω_2 , I have a discriminant function, which is given by $g_2(X)$ and for ω_3 , I have a discriminant function, which is given by $g_3(X)$. So, these are the discriminant function for 3 different classes. For ω_1 , I have $g_1(X)$. For ω_2 , I have $g_2(X)$. For ω_3 , I have $g_3(X)$.

So, naturally our classification is if I find that $g_1(X) > g_2(X)$ and $g_1(X) > g_3(X)$, then X will be classified to class ω_1 . If $g_3(X) > g_1(X)$ and $g_3(X) > g_2(X)$, then X will be classified to class ω_3 . If $g_2(X) > g_1(X)$ and $g_2(X) > g_3(X)$, then X will be classified to class ω_2 . So now, let us see that what are the different functional forms each of these linear discriminating functions have. So, I will put say $g_1(X) = 10x_1 - x_2 - 10$, $g_2(X) = x_1 + 2x_2 - 10$. And Suppose that $g_3(X) = x_1 - 2x_2 - 10$, so these are the discriminating functions, linear discriminating functions for different classes.

So, in order to classify a sample to class ω_1 , what I have to do is I have to find out what is the value of $g_1(X)$. What is the value of $g_2(X)$ and I have to find out what is the value of $g_3(X)$.

If I find that $g_1(X) > g_2(X)$, then obviously between these 2 classes, X belongs to ω_1 . If also $g_1(X) > g_3(X)$, then between these 2 classes, X also belongs to class ω_1 .

So, undoubtedly, the unknown feature vector X will be classified to class ω_1 . Now, if it is so happening that $g_1(X) > g_2(X)$, but $g_3(X) > g_1(X)$, so those are the different such confusing cases. We will see what happens to all these different confusing cases. So, here what happen I will do is, this is the discriminating function for class ω_1 . This is the discriminating function for class ω_2 . This is the discriminating function for class ω_3 .

Now, what I want to do is, I want to find out what is the decision boundary between pair of classes say between ω_1 and ω_2 . What is the decision boundary between ω_2 and ω_3 ? What is the decision boundary and between ω_1 and ω_3 ? What is the decision boundary? So, for that, what I need to compute is $g_{12}(X)$. $g_{12}(X)$ will be given by $g_1(X) - g_2(X)$. If this $g_1(X) - g_2(X)$, if this is positive or $g_{12}(X)$, then obviously X belongs to class ω_1 , when I compare between classes ω_1 and ω_2 , so that decision boundary between the classes ω_1 and ω_2 will be given by $g_{12}(X) = 0$ or $g_1(X) - g_2(X) = 0$.

Similarly, I find out what is $g_{23}(X)$ that is the decision boundary between class ω_2 and class ω_3 . So, I will compute $g_{23}(X)$ as $g_2(X) - g_3(X)$. And I will equate this to 0 to get the decision boundary between class ω_2 and class ω_3 . Similarly, I will compute what is $g_{13}(X)$.

(Refer Slide Time: 09:43)

$g_1(x)$	$g_2(x)$	$g_3(x)$
$g_1(x) = 10x_1 - x_2 - 10$		
	$g_2(x) = x_1 + 2x_2 - 10$	
		$g_3(x) = x_1 - 2x_2 - 10$
		$g_{12}(x) = g_1(x) - g_2(x) = 0$
		$g_{23}(x) = g_2(x) - g_3(x) = 0$
		$g_{13}(x) = g_1(x) - g_3(x) = 0$



Which is nothing but $g_1(X) > g_3(X)$. And by equating this to 0, I get the decision boundary between class ω_1 and class ω_3 . Now, let us see that what each of these functions $g_{12}(X)$, $g_{23}(X)$ or $g_{13}(X)$ take using these discriminating functional values $g_1(X)$, $g_2(X)$ and $g_3(X)$.

(Refer Slide Time: 10:20)

$$g_{12}(x) = g_1(x) - g_2(x) = 0$$

$$\Rightarrow 9x_1 - 3x_2 = 0 \Rightarrow 3x_1 - x_2 = 0$$

$$g_{23}(x) = g_2(x) - g_3(x) = 0$$

$$\Rightarrow 4x_2 = 0 \Rightarrow x_2 = 0$$

$$g_{13}(x) = g_1(x) - g_3(x) = 0$$

$$\Rightarrow 9x_1 + x_2 = 0$$

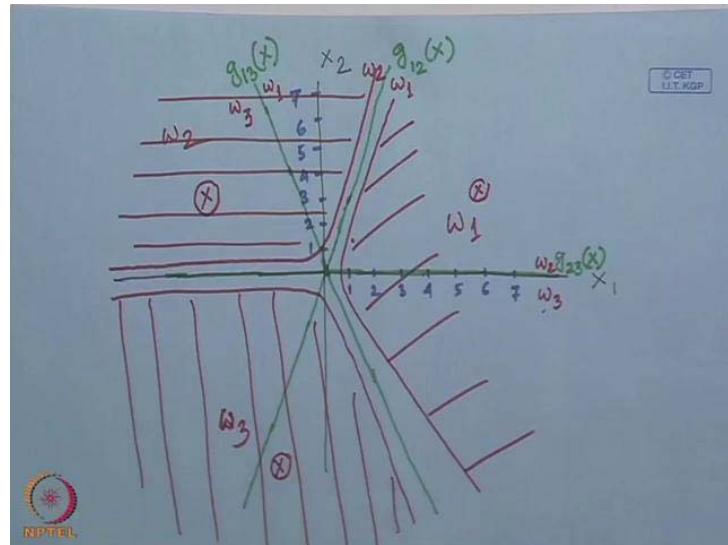
So, from here, you find that my $g_{12}(X)$ which is nothing but $g_1(X) - g_2(X)$. See if I compute this $g_1(X) - g_2(X)$. That will be $g_{12}(X) = 9x_1 - 3x_2$. So, this is equal to $g_1(X) - g_2(X)$, which is equal to 0. If I compute this, then as I said that it will be $9x_1 - 3x_2 = 0$. And which is nothing but $3x_1 - x_2 = 0$. Similarly, when I compute $g_{23}(X)$, it is nothing but $g_2(X) - g_3(X)$. So, $g_{23}(X) = 4x_2 = 0$.

So, that gives me $4x_2 = 0$ of which is nothing but $x_2 = 0$. $x_2 = 0$ is nothing but x_1 axis. Similarly, when I compute $g_{13}(X)$ is $g_1(X) - g_3(X)$, so when I do that $g_{13}(X) = 9x_1 + x_2 = 0$. So, you find the decision boundaries between the pair of classes.

The decision boundary between ω_2 and ω_1 is given by $3x_1 - x_2 = 0$, $g_{13}(X) = 9x_1 + x_2 = 0$ decision boundary between ω_1 and ω_3 . And $x_2 = 0$ decision boundary between ω_2 and ω_3 . So, these are the different decision boundaries and what we are talking about. We are talking about; we are considering 2 dimensional feature vectors having components x_1 and x_2 . So, it

is a 2 dimensional space. Now, I plot all these decision boundaries. Let us see what is the kind of situation that we have the first decision boundary.

(Refer Slide Time: 14:24)



So, I will put it this way. This is say my x_1 axis. This is my x_2 axis. Now, when you come to this $g_{12}(X)$ that is the decision boundary between class ω_1 and ω_2 , which is given by $3x_1 - x_2 = 0$. So, this decision boundary, I plot the decision boundary. The decision boundary will look like this.

So, I will put it this way say suppose here I have 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7 and so on. So, this $g_{12}(X)$, which is $3x_1 - x_2 = 0$ that will be somewhere here. This is $g_{12}(X)$. Coming to this $g_{23}(X)$, which is $x_2 = 0$ that is nothing but my x_1 axis. So, this is $g_{23}(X)$. That is the decision boundary between the classes, ω_2 and ω_3 and coming to the next one. Here, $g_{13}(X) = 9x_1 + x_2 = 0$.

So, if I plot this decision boundary on the same graph, you find that this plot will be something like this, where this is $g_{13}(X)$. Now, when it comes to $g_{12}(X)$, we said that if $g_{12}(X) > 0$, then X belongs to class ω_1 . If it is less than 0, then X belongs to class ω_2 . Similarly, $g_{23}(X)$, if $g_{23}(X) > 0$, then that X belongs to class ω_2 . If it is less than 0, then that X belongs to class ω_3 . Similarly, considering this $g_{13}(X)$, if $g_{13}(X) > 0$, then that X belongs to class ω_1 . If it is less than 0, then that X belongs to class ω_3 .

So, let us see that what the positive side of these different decision boundaries. If I take upon a point say $(1, 1)$ and put that into $g_{12}(X)$, so over here $g_{12}(X)$, if I put $x_1 = 1$ and $x_2 = 1$, then $g_{12}(X) = 2$. That means this point $(1, 1)$, which is somewhere over here is on the positive side of $g_{12}(X)$. So, this $g_{12}(X)$ as it divides this 2 dimensional space into 2 half space. This side of the space is positive and the other side is negative. This means that when I go or classification, this side of the decision boundary is ω_1 and this side of the decision boundary is ω_2 .

Similarly, when I consider this $g_{23}(X)$ as $g_{23}(X)$, it is nothing but x_2 . So, whenever x_2 is positive that side belongs to ω_2 . Whenever x_2 is negative, then that side belongs to ω_3 . So, considering this decision boundary, this side belongs to omega 2, this side belongs to ω_3 . Similarly, when you consider this $g_{13}(X)$, $g_{13}(X)$ is nothing but $9x_1 + x_2$.

So, if I set $x_1 = 1$ and $x_2 = 1$, this is nothing but 10, which is positive. So, coming to $g_{13}(X)$, this side of the decision boundary gives me class ω_1 . This side of the decision boundary gives me class ω_3 . Now, let us try to combine what we get from these 3 decision boundaries. So, coming to this one between ω_1 and ω_2 , this side belongs to ω_1 between ω_3 and ω_1 this side belongs to ω_1 . So, you find that so far, as this decision boundary is concerned between class ω_1 and ω_2 , this half belongs to class ω_1 . Considering ω_1 and ω_3 , this half belongs to ω_1 .

So, finally, the region which will be allocated to class ω_1 is the intersection of these 2 regions. And which is nothing but this region which is allocated to class ω_1 . Then when I consider ω_2 , you find that between ω_1 and ω_2 , this half belongs to class ω_2 . And between ω_3 and ω_2 , it is this half which belongs to class 2; sorry it is upper half ω_2 and ω_3 . It is the upper half which belongs to class ω_2 . So, $g_{12}(X)$ tells me that this half goes to ω_1 and $g_{23}(X)$ tells me that this half goes to ω_2 .

So, finally, the region which is allotted to ω_2 is nothing but this region. So, this is the region, which goes to class ω_2 and coming to ω_3 . This just $g_{23}(X)$ tells me that this lower half goes to ω_3 and $g_{13}(X)$ gives me tells me that it is this half which goes to ω_3 and intersection of this two is nothing but this region, so this is the region which is going to class ω_3 .

So, when I have these decision boundaries $g_{13}(X)$, $g_{12}(X)$ and $g_{23}(X)$, if I have an unknown feature vector, which is lying within this region. Obviously, this unknown feature

vector will be classified to class ω_1 . If I have an unknown feature vector which is in this region, this will be classified to class ω_2 . And I have an unknown feature vector, which is here. This unknown feature vector will be classified to class ω_3 .

So, from these different discriminating functions, when I get the decision boundaries, the decision boundaries divide the spaces into 3 different sub spaces. Where every sub space is allotted to one of the classes. So, this was very simple case. Now, let us think of a situation that if these discriminating functions themselves are taken as the decision boundaries. In other words, what I mean to say is,

(Refer Slide Time: 24:51)

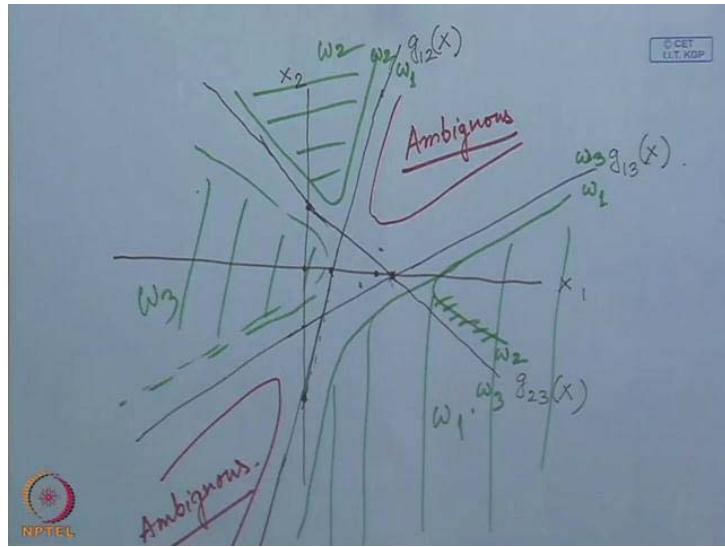
$$g_{12}(x) = 10x_1 - x_2 - 10 = 0$$

$$g_{23}(x) = x_1 + 2x_2 - 10 = 0$$

$$g_{13}(x) = x_1 - 2x_2 - 10 = 0$$

If I take say $g_{12}(X) = 10x_1 - x_2 - 10 = 0$, $g_{23}(X) = x_1 + 2x_2 - 10 = 0$ and say $g_{13}(X) = x_1 - 2x_2 - 10 = 0$. If I have a situation something like this, then let us see how the different regions will be allotted to different classes. So, what I have to do is, I have to again draw the straight lines represented by these 3 different functions. So, let us draw those straight lines.

(Refer Slide Time: 26:01)



So, I have again this axis. This is x_1 , this is x_2 . Here you find that this particular straight line $10x_1 - x_2 - 10 = 0$. See if I put $x_1 = 0$, $x_2 = 10$. So, that will be point somewhere, let us say over here. And if I put $x_2 = 0$, $x_1 = 1$. So, that will be point somewhere over here.

So, I have this $g_{12}(X)$ in this case, which is given by this straight line. This is $g_{12}(X)$. Coming to the next one, $g_{23}(X)$, if I put $x_1 = 0$, $x_2 = 5$. So, it is $(0, 5)$, which will be a point say somewhere over here. If I put $x_2 = 0$, $x_1 = 10$. So, that is the point somewhere over here. So, that will be a point somewhere over here.

So, this is a straight line, which represents $g_{23}(X)$. and coming to the third one, which is $g_{13}(X)$, which is nothing but $x_1 - 2x_2 - 10 = 0$. So, here you find that if I put $x_1 = 0$, $x_2 = -5$. So, that will be a point somewhere over here. And if I put $x_2 = 0$, $x_1 = 10$. So, that is this point itself.

So, I get a straight line, which is this. And this straight line represents $g_{13}(X)$. now, let us put the point $(0, 0)$ on each of this, to each of this discriminating functions. So, for $(0, 0)$, you find that $g_{12}(X)$ is negative, $g_{23}(X)$ is negative, $g_{13}(X)$ that will also be negative. So, that means this origin lies on the negative side of each of these planes.

So, as the origin lies on the negative side that indicates that considering this $g_{12}(X)$, this side of $g_{12}(X)$ is ω_1 . This is because on this side, I have $g_{12}(X)$ is equal to positive. On this side,

it is ω_2 . Coming to $g_{13}(X)$ as this is on the negative side, so $g_{13}(X)$ will be positive on the other side. So, this side will be your ω_1 . This side will represent ω_3 . Coming to this $g_{23}(X)$, this side will represent ω_2 . And this side will represent ω_3 . Coming to this $g_{23}(X)$, this side will represent ω_2 . This side will represent ω_3 .

So, with this now, let us try to distribute or allot different regions to different classes. So, here, you will find between class ω_1 and ω_2 , this side is ω_1 . Between ω_3 and ω_1 , this side is ω_1 . So, when I take the intersection of this and this, you find that this is the region, which will be allotted to class ω_1 ; sorry this is the region that will be allotted to class ω_1 . Similarly, coming to ω_3 , this side between ω_1 and ω_3 , this side is ω_3 and between ω_2 and ω_3 . So, coming over here, this side goes to ω_3 . And between W_2 and W_3 , this side goes to W_3 .

So, I will have this particular region, which will be allotted to W_3 . And coming to W_2 , you find that this is the region, which is allotted to W_2 . And between W_2 and W_3 , this is the region on this side that is allocated to W_2 . So, if I take the intersection, then this is the region which will be allotted to W_2 . So, I have this region allotted to W_1 . I have this region allotted to W_2 . And I have this region allotted to W_3 . And as I do that, you find that there is a region over here and there is a region over here. These 2 regions cannot be allotted to any of the classes.

So, I have some empty region or these are called ambiguous regions. So, when I have this linear discriminator considering the decision boundaries between different classes. And if I have a multiclass problem, it is quite possible that in the feature space, I can have few sub spaces. If a feature vector falls that feature vector cannot be classified to any of the classes because those sub spaces are actually ambiguous regions.

So, this is one of the problems that I wanted to discuss to see that how you can solve or how you can identify the regions belonging to different classes when you have linear discriminating functions. So, the second problem that I will try to discuss today is on classifier design using perception criteria. So, if you will remember what we said is in case of perception criteria, the criteria function for designing of the classifier was something like this.

(Refer Slide Time: 34:08)

$$J_p(W) = \sum_{y \text{ misclassified}} -W^t y$$

$$\begin{aligned} W(0) \\ W(k+1) &= W(k) - \eta \nabla J_p(W) \\ &= W(k) + \eta \sum_{y \text{ misclassified}} y \end{aligned}$$

$$x \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_d \end{bmatrix}$$

It was $J_p(W)$ which was given by $\sum_{y \text{ misclassified}} (-W^t y)$, where this summation is taken over all

Y , which are misclassified. And to get the value of W that minimizes this criterion function the perception criteria function, we have used the gradient descent procedure. The gradient descent procedure was something like this that initially you assume or you set a weight vector, which is set at random.

So, initially you assume a weight vector W naught. And then in every iterative step, you set say $W(k+1)$, which comes from $W(k)$. That is your weight vector at the previous iteration. We are following gradient descent approach. So, it will be $W(k+1) = W(k) - \eta \nabla J_p(W)$. And if I take gradient of $J_p(W)$, it will be nothing but $\sum_{y \text{ misclassified}} (-y)$; for all the samples Y which are misclassified.

So, effectively this algorithm comes to $W(k+1) = W(k) + \eta \sum_{y \text{ misclassified}} (y)$. And how do I get this Y if my original given feature vector is X , I append 1 to this X ? So, if this X is say x_1, x_2 up to x_d , I get Y by appending 1 to these vectors. So, my Y becomes 1, x_1, x_2 up to x_d . And then what we do is if this Y belongs to class say ω_1 , Y remains as it is.

If Y belongs to class ω_2 , then we negate Y , so that whenever a sample is correctly classified, this term $W^t Y$ is always positive if the sample is correctly classified, whereas if the sample is not correctly classified, then $W^t Y$ will be negative making this $-W^t Y$ to be positive. And

these are the iterative steps, which have to be performed unless the algorithm converges. That means unless I get a \mathbf{W} weight vector, which classifies all the training samples correctly, so this was our perception criteria for design of linear classifiers. And this is particularly the one, which is applicable for two class problem.

So, we are considering 2 class problem. Then we have extended this concept to a multi class problem by using a construction technique, which is called Keslers construction. So, I will consider auto class problem. I will take a problem from a two class case and try to see the different steps of the algorithm, how your vector \mathbf{W} evolves? Of course, I will not try to solve the entire problem. I will not try to get the solution. I will simply explain the steps. The reason is that it takes a large number of iterations. So, I get a situation something like this.

(Refer Slide Time: 38:40)

$$\left\{ \begin{pmatrix} 1 \\ 8 \end{pmatrix}, \begin{pmatrix} 6 \\ 7 \end{pmatrix}, \begin{pmatrix} 8 \\ 5 \end{pmatrix}, \begin{pmatrix} 11 \\ 4 \end{pmatrix} \right\} \in \omega_1$$

$$\left\{ \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \begin{pmatrix} 3 \\ 3 \end{pmatrix}, \begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 11 \\ 1 \end{pmatrix} \right\} \in \omega_2.$$

$$w_0 = \begin{bmatrix} -10 \\ 1 \\ 1 \end{bmatrix}$$

Suppose that you have been given a set of feature vectors say $\begin{pmatrix} 1 \\ 8 \end{pmatrix}, \begin{pmatrix} 6 \\ 7 \end{pmatrix}, \begin{pmatrix} 8 \\ 5 \end{pmatrix}, \begin{pmatrix} 11 \\ 4 \end{pmatrix}, \dots, \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \begin{pmatrix} 3 \\ 3 \end{pmatrix}, \begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 11 \\ 1 \end{pmatrix}$, 2 dimensional feature vectors. Then it is $\begin{pmatrix} 6 \\ 7 \end{pmatrix}$, then $\begin{pmatrix} 8 \\ 5 \end{pmatrix}$, then $\begin{pmatrix} 11 \\ 4 \end{pmatrix}$. So, this is the set of feature vectors, which are taken from class ω_1 . I have another set of feature vectors say $\begin{pmatrix} 2 \\ 4 \end{pmatrix}$, maybe say $\begin{pmatrix} 3 \\ 3 \end{pmatrix}$, maybe $\begin{pmatrix} 6 \\ 2 \end{pmatrix}$ and suppose $\begin{pmatrix} 11 \\ 1 \end{pmatrix}$. These are the feature vectors, which are taken from class ω_2 .

So, I have these 2 set of feature vectors, which are my training vectors to train the classifier. So, it is a supervised planning. And I also assume that initial weight vector, which we said

that the initial weight vector is taken at random. So, I assume that my initial weight vector

$$W(0) \text{ is something like } \begin{bmatrix} -10 \\ 1 \\ 1 \end{bmatrix}.$$

So, this is my initial weight vector. So, considering these vectors to be the vector sets, I have to compute Y out of this vector. So, as we said that we have to append 1 to each of these vectors, after appending 1 all the vectors, which are coming from class ω_2 they have to be negated.

(Refer Slide Time: 40:21)

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
1	1	1	1	-1	-1	-1	-1
1	6	8	11	-2	-3	-6	-11
8	7	5	4	-4	-3	-2	-1

$\eta = 0.2$

$W(0) = \begin{bmatrix} -10 \\ 1 \\ 1 \end{bmatrix}$

So, I will get Y_1 . I will put it like this. I will get Y_1 which is nothing but $\begin{bmatrix} 1 \\ 1 \\ 8 \end{bmatrix}$ because I have

to append 1 over here. So, it becomes $\begin{bmatrix} 1 \\ 1 \\ 8 \end{bmatrix}$. Y_2 is $\begin{bmatrix} 1 \\ 6 \\ 7 \end{bmatrix}$. So, I will get $\begin{bmatrix} 1 \\ 6 \\ 7 \end{bmatrix}$. Y_3 is $\begin{bmatrix} 1 \\ 8 \\ 5 \end{bmatrix}$. So, Y_3

will be $\begin{bmatrix} 1 \\ 8 \\ 5 \end{bmatrix}$. Y_4 is $\begin{bmatrix} 1 \\ 11 \\ 4 \end{bmatrix}$. So, these are the appended samples or modified samples from class

ω_1 . In the same manner, I have to get the appended sample or modified samples from class

ω_2 . So, I will take these samples from class ω_2 . And as I have said that after appending 1,

these vectors are to be negated. So, my Y_5 will be $\begin{bmatrix} -1 \\ -2 \\ -4 \end{bmatrix}$.

In the same manner Y_6, X_6 is $\begin{pmatrix} 3 \\ 3 \end{pmatrix}$. So, Y_6 will be $\begin{bmatrix} -1 \\ -3 \\ -3 \end{bmatrix}$. Y_7, X_6 is $\begin{pmatrix} 6 \\ 2 \end{pmatrix}$. So, Y_7 will be $\begin{bmatrix} -1 \\ -6 \\ -2 \end{bmatrix}$.

Y_8, X_8 was $\begin{pmatrix} 11 \\ 1 \end{pmatrix}$. So, Y_8 will be $\begin{bmatrix} -1 \\ -11 \\ -1 \end{bmatrix}$. So, these are all my modified samples. And I also

take where this if you look at this algorithm, this η which indicates the rate of convergence.

Let me take this state of convergence η is equal to say 0.2. As I said my initial weight vector

$W(0)$ is $\begin{bmatrix} -10 \\ 1 \\ 1 \end{bmatrix}$. So, how the algorithm has to work? I have to take $W^t Y$ for each of these Y s,

if $W^t Y$ is positive that sample is correctly classified by this weight vector. If $W^t Y$ is negative for any of the Y , then that sample Y is misclassified by this weight vector.

So, I have to identify all those samples, Y which are misclassified for which $W^t Y < 0$. So, all those are misclassified samples and those misclassified samples are to be used in my gradient descent procedure for modification of the weight vector W . So, let us see what values of $W^t Y$ that I get for different samples of Y . So, if I take, as you find that Y_{11} , the first component is 1, the second component is 2, and the third component is 8. So, clearly here it is $W(0)$ at the 0th iteration that is the initial value.

So, $W_{(0)}^t Y_1$ will be -1 that is negative that means Y_1 is misclassified by this $W(0)$, Coming to Y_2 , $W_{(0)}^t Y_2$ which is nothing but 3 that is greater than 0. So, Y_2 is correctly classified by this sample. So, I do not have to consider Y_2 for modification of W . I definitely have 2. Consider Y_1 for modification of W . So, let us see that what these different values that we get.

(Refer Slide Time: 45:18)

©CET
IIT-KGP

$$\begin{aligned}
 W^t(0)Y_1 &= -1 < 0 & Y_1 \\
 W^t(0)Y_2 &= +3 > 0 \\
 W^t(0)Y_3 &= +3 > 0 & \sum Y = Y_1 + Y_8 \\
 W^t(0)Y_4 &= +5 > 0 \\
 W^t(0)Y_5 &= +4 > 0 \\
 W^t(0)Y_6 &= +4 > 0 \\
 W^t(0)Y_7 &= +4 > 0 \\
 W^t(0)Y_8 &= -2 < 0 & Y_8
 \end{aligned}$$

$= \begin{bmatrix} 0 \\ -10 \\ 7 \end{bmatrix}$

NPTEL

I will get $W_{(0)}^t Y_1$, if I compute this $W_{(0)}^t Y_1$, this has values -1 which is less than 0. So, I have to consider Y_1 for modification of my weight vector. When I consider $W_{(0)}^t Y_2$, $W_{(0)}^t Y_2$ is 3 as we have just seen which is greater than 0. So, I do not consider Y_2 for modification of the weight vector. Similarly, $W_{(0)}^t Y_3$ which also 3 greater than 0. I do not have to consider Y_3 medication $W_{(0)}^t Y_4$. Which is, it will come out to be 5. That is again greater than 0. $W_{(0)}^t Y_5$ which will be 4. I compute this which will again be greater than 0.

$W_{(0)}^t Y_6$ which will again be 4 greater than 0. $W_{(0)}^t Y_7$, which if you compute, it will be 4 again, which is greater than 0. $W_{(0)}^t Y_8$, if you compute this; it will be -2, which is less than 0. That means I have to consider this Y_8 for modification of the weight vector W . now, looking at this algorithm, you find that weight updation algorithm is

$$W(k+1) = W(k) + \eta \sum_{y \text{ missclassified}} (y).$$

So, over here, it is Y_1 and Y_8 , which are misclassified.

So, sum of Y of the misclassified samples is nothing but $Y_1 + Y_8$. And you look at this. A

modified sample Y_1 is $\begin{bmatrix} 1 \\ 1 \\ 8 \end{bmatrix}$ and Y_8 is $\begin{bmatrix} -1 \\ -11 \\ -1 \end{bmatrix}$. So, if I add this to what I get $\begin{bmatrix} 0 \\ -10 \\ 7 \end{bmatrix}$. So, this

sum of Y is nothing but $\begin{bmatrix} 0 \\ -10 \\ 7 \end{bmatrix}$. This is sum of Y, which is misclassified. Now, what I have to

compute is by using this, I have to compute the next iterated value of W.

It is equal to W(1) because I had W(0), I had chosen W(0) at random.

(Refer Slide Time: 48:46)

$$\begin{aligned}
 W(1) &= W(0) + \eta \sum y \\
 &= \begin{bmatrix} -10 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ -2 \\ 1.4 \end{bmatrix} \\
 &= \begin{bmatrix} -10 \\ -1 \\ 2.4 \end{bmatrix}
 \end{aligned}$$

So, my $W(1) = W(0) + \eta \sum_{y \text{ missclassified}} (y)$ where eta is we have assumed η as 0.2. So, this $W(1)$

simply becomes, $\begin{bmatrix} -10 \\ 1 \\ 1 \end{bmatrix} + 0.2 \begin{bmatrix} 0 \\ -10 \\ 7 \end{bmatrix}$ will be $\begin{bmatrix} -10 \\ -1 \\ 2.4 \end{bmatrix}$, right? So, $W(1)$ simply becomes $\begin{bmatrix} -10 \\ -1 \\ 2.4 \end{bmatrix}$.

So, you see that how we have modified the weight vector W from W(0) to W(1) using W(0), I have identified all the feature vectors, all the samples Y which are misclassified. Those misclassified samples are used to modify W(0) to get W(1). Now, as I have this W(1) now, I have to try to classify all those samples Y₁ to Y₈ using this W(1). So, I have to compute $W_{(1)}^T Y_1$. I have to compute $W_{(1)}^T Y_2$, $W_{(1)}^T Y_3$ up to $W_{(1)}^T Y_8$. And I have to identify all those Ys from which $W_{(1)}^T Y$ is negative. And these are the samples, these are the Ys, which will be used to update W to get W(2).

(Refer Slide Time: 51:25)

$$\begin{aligned}
 W_{(1)}^t Y_1 &= 8.2 > 0 \\
 W_{(1)}^t Y_2 &= 0.8 > 0 \\
 W_{(1)}^t Y_3 &= -6 < 0 \\
 W_{(1)}^t Y_4 &= -11.4 < 0 \\
 W_{(1)}^t Y_5 &= 2.4 > 0 \\
 W_{(1)}^t Y_6 &= 5.8 > 0 \\
 W_{(1)}^t Y_7 &= 11.2 > 0 \\
 W_{(1)}^t Y_8 &= 18.6 > 0
 \end{aligned}$$

$$\sum Y = Y_3 + Y_4 = \begin{bmatrix} 2 \\ 19 \\ 9 \end{bmatrix}$$

So, if I compute this, you will find that $W_{(1)}^t Y_1$ that will be 8.2, which is positive $W_{(1)}^t Y_2$, which will be 0.8, which is also positive. $W_{(1)}^t Y_3$ that will be -6, which is negative that means this Y_3 has to be used or modification of W .

Similarly, $W_{(1)}^t Y_4$, if I compute it will be -11.4, which is again negative. So, Y_4 has to be considered for weight updation at the next iteration. $W_{(1)}^t Y_5$ which will be equal to 2.4. That is greater than 0. $W_{(1)}^t Y_6$ which will be 5.8. Again, it is greater than 0. $W_{(1)}^t Y_7$ will be 11.2 again greater than 0. $W_{(1)}^t Y_8$ which will be 18.6. Again, greater than 0. So, I have to consider this Y_3 and Y_4 to get my $W(2)$ that is the weight vector to be used at the next iteration.

if I add this Y_3 and Y_4 sum of Y is nothing but $Y_3 + Y_4$, and you find that Y_3 is this $\begin{bmatrix} 1 \\ 8 \\ 5 \end{bmatrix}$

Y_4 is $\begin{bmatrix} 1 \\ 11 \\ 4 \end{bmatrix}$, I add these two, I get $\begin{bmatrix} 2 \\ 19 \\ 9 \end{bmatrix}$. So, $W(2)$ that I have that is the weight vector to be

used at the next level of iteration is nothing but $W(2) = W(1) + \eta(Y_3 + Y_4)$. So, I had $W(1)$

that is this, that is $\begin{bmatrix} -10 \\ -1 \\ 2.4 \end{bmatrix}$.

(Refer Slide Time: 54:06)

W(2) = \begin{bmatrix} -10 \\ -1 \\ +2.4 \end{bmatrix} + \begin{bmatrix} 0.4 \\ 3.8 \\ 0.8 \end{bmatrix}
$$= \begin{bmatrix} -9.6 \\ -2.8 \\ 3.2 \end{bmatrix}$$
 The NPTEL logo is located at the bottom left of the whiteboard."/>

So, from here I get $W(2)$, which will be equal to $\begin{bmatrix} -10 \\ -1 \\ 2.4 \end{bmatrix} + 0.2 \begin{bmatrix} 2 \\ 19 \\ 9 \end{bmatrix}$. So, this will be simply

$\begin{bmatrix} -9.6 \\ -2.8 \\ 3.2 \end{bmatrix}$. So, I get my next weight vector $W(2)$, which is given by this. That is $\begin{bmatrix} -9.6 \\ -2.8 \\ 3.2 \end{bmatrix}$.

So, I get my next weight vector, which is given by this, that is $\begin{bmatrix} -9.6 \\ -2.8 \\ 3.2 \end{bmatrix}$. Again using this

$W(2)$, I have to try to see whether the feature vectors are correctly classified or not. If with this $W(2)$, all the feature vectors are correctly classified that means $W_{(2)}^t Y_1 > 0$ for all the Y .

That means this weight vector $W(2)$ is my correct classifier.

Again, if I get any Y , which is negative for which $W_{(2)}^t Y$ is negative, those Y s feature vectors are to be used at the next step to get the value of $W(3)$. This is how the perception criteria works. Then we have also talked about the sequential version of the perception criteria.

So, in the original form, what you are doing is in every pass, I am identifying all the feature vectors, which are misclassified by that weight vector. And all those feature vectors are added together. That is used for more modification of the weight vector. In the serial version of the perception criteria algorithm or the perception algorithm, I do not identify all the feature vectors.

(Refer Slide Time: 56:30)

© CET
I.I.T. KGP

$$\begin{aligned}
 W(1)Y_1 &= 8.2 > 0 \\
 W(1)Y_2 &= 0.8 > 0 \\
 W(1)Y_3 &= -6 < 0 \\
 \cancel{W(1)Y_4} &= -11.4 < 0 \\
 W(1)Y_5 &= 2.4 > 0 \\
 W(1)Y_6 &= 5.8 > 0 \\
 W(1)Y_7 &= 11.2 > 0 \\
 W(1)Y_8 &= 18.6 > 0
 \end{aligned}$$

$\sum Y = Y_3 + Y_4$
 $= \begin{bmatrix} 2 \\ 19 \\ 9 \end{bmatrix}$

$$\begin{aligned}
 W(2) &= W(1) + \eta Y_3 \\
 W(3) &= W(2) + \eta Y_7
 \end{aligned}$$

What I do is the moment I find that this $W(1)$ misclassifies Y_3 , I do not wait for Y_4 to be identified. Immediately, I use this Y_3 to get $W(2)$, which will be $W(1) + \eta Y_3$. So, immediately I get $W(2)$, and maybe this $W(2)$ will correctly classify Y_4 . So, Y_4 will not be misclassified by this, but it may so happen that for Y_7 , I find that this feature vector $W(2)$ misclassifies Y_7 . So, whenever it misclassifies Y_7 , immediately I will compute $W(3)$ to be $W(2) + \eta Y_7$.

So, whenever I get a misclassified sample; immediately using that misclassified sample, you modify the weight vector and in a complete pass, if I have W where for the same W , Y_1 to Y_8 , all of them are correctly classified, then I will take that W to be my solution vector. So, I hope with these problems, your concept of the discriminating functions, the decision boundaries, the class regions and the second one have been able to clarify how the perception algorithm works to get the weight vector while supervised learning of a classifier.

Thank you.

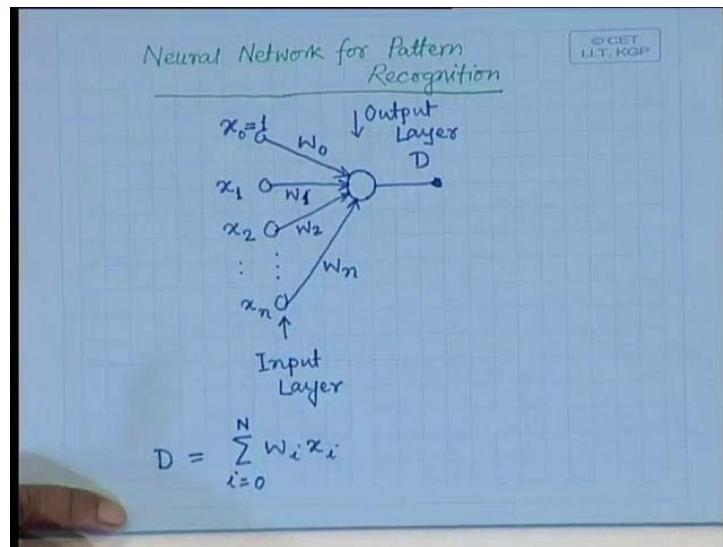
Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecturer - 24
Neural Networks for Pattern Recognition

Good morning, so we are going to continue our discussion on neural network for pattern recognition. So, firstly I will just briefly go through, what we have done in our last class and then we will proceed further. So, in the last class we have talked about a single layer single output neural network, where when I say it is single layer, basically there are two layers. One of the layers is the input layer and the nodes in the input layer are simply used to connect the feature vectors.

So, the purpose of this input layer node is whatever is inputted to the input layer nodes, those nodes simply forward that input to the layer above it. And those inputs get weighted by the corresponding connection weights connecting the input layers to the layer above it, and summed up at the next layer and on top of that you apply some of sort of non-linearity.

(Refer Slide Time: 01:23)



So, effectively the single layer, single output neural network, we have said was something like this, that it has a number of input layer nodes. So, this is the layer which is input layer. So, the number of nodes in the input layer is same as the dimensionality of the feature vectors, last one which is used to take care of the bias or bias weight. And the number of

nodes in the output layer in this case will be only one, which will indicate either one of the two classes.

So, if the output of the output layer node is greater than threshold in which is the output of the output layer node will set equal to one, which will indicate say class one and if the output is zero, it will indicate class 2. So, the connections that will have is something like this. So, here I have x_0 , which is usually set to 1 and the connection weight over here is the w_o , which takes care of the bias weight. Whereas, the other inputs will have a different feature components x_1, x_2 up to say x_n and the corresponding connection weights are w_1, w_2 up to w_n .

And output of the output layer node, which let say that this output is equal to D . So, this is my output layer node, where D will be equal to the weighted sum of the input feature vector components. So, it will be simply $\sum_{i=0}^N w_i x_i$, where i varies from 0 to N , where N is the dimensionality of the future vector and x_0 which is usually set to 1, that is to take care of the bias weight.

So, then what we said is while training the neural network, we have to decide this connection weights or what will be the weight vector that will classify the unknown feature vectors between one of the 2 classes, say class ω_1 and class ω_2 . Now, while training because all the feature vectors which are given, we know what is the class belongingness of those feature vectors. So, we effectively know that what expected output is.

So, if I take training sample, a training feature vector which we know that the training feature belongs to class ω_1 , we know that the corresponding output as to be equal 1 or if I know that the future vectors belong to class ω_2 , then it is known that the corresponding output has to be equal to 0. So, for that training sample or for each of the sample which is presented to the neural network, we know what the corresponding output.

And because initially, the weights are not set properly because we start with arbitrarily set weights of very low value. So, even if our training sample which belongs to the class ω_1 is given to the neural network, the output may be 0 or output may have any of the continuous value between 0 and 1. So, it may not be equal to 1, so what is the expected output and what is the actual output that you get the difference between that between these 2 is the error. So, if I say that my actual output that I obtain after giving a training sample, after inputting a

training sample is given by the expected output is say d and the actual output that I get D,
which is nothing but the $\sum_{i=0}^N w_i x_i$, i value is from 0 to n.

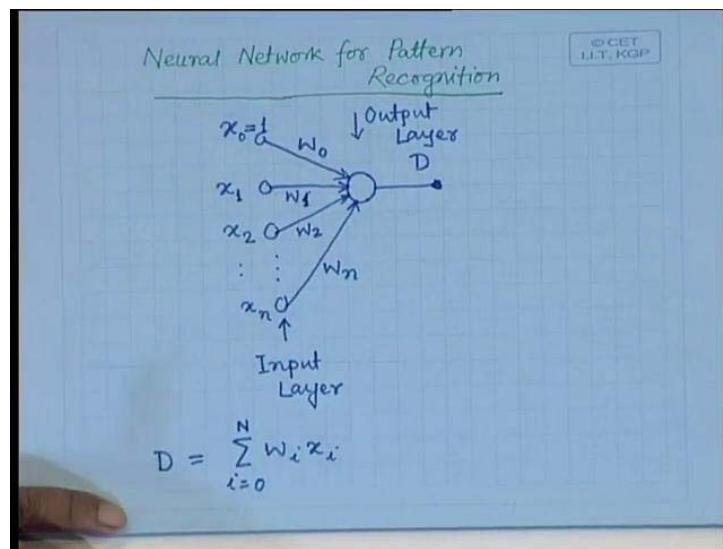
Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecturer - 25
Neural Networks for Pattern Recognition
(Contd.)

Good morning, so we are going to continue our discussion on neural network for pattern recognition. So, firstly I will just briefly go through, what we have done in our last class and then we will proceed further. So, in the last class we have talked about a single layer single output neural network, where when I say it is single layer, basically there are two layers. One of the layers is the input layer and the nodes in the input layer are simply used to connect the feature vectors.

So, the purpose of this input layer node is whatever is inputted to the input layer nodes, those nodes simply forward that input to the layer above it. And those inputs get weighted by the corresponding connection weights connecting the input layers to the layer above it, and summed up at the next layer and on top of that you apply some of sort of non-linearity.

(Refer Slide Time: 01:23)



So, effectively the single layer, single output neural network, we have said was something like this, that it has a number of input layer nodes. So, this is the layer which is input layer. So, the number of nodes in the input layer is same as the dimensionality of the feature vectors, last one which is used to take care of the bias or bias weight. And the number of

nodes in the output layer in this case will be only one, which will indicate either one of the two classes.

So, if the output of the output layer node is greater than threshold in which is the output of the output layer node will set equal to one, which will indicate say class one and if the output is zero, it will indicate class 2. So, the connections that will have is something like this. So, here I have x_0 , which is usually set to 1 and the connection weight over here is the w_o , which takes care of the bias weight. Whereas, the other inputs will have a different feature components x_1, x_2 up to say x_n and the corresponding connection weights are w_1, w_2 up to w_n .

And output of the output layer node, which let say that this output is equal to D . So, this is my output layer node, where D will be equal to the weighted sum of the input feature vector components. So, it will be simply $\sum_{i=0}^N w_i x_i$, where i varies from 0 to N , where N is the dimensionality of the future vector and x_0 which is usually set to 1, that is to take care of the bias weight.

So, then what we said is while training the neural network, we have to decide this connection weights or what will be the weight vector that will classify the unknown feature vectors between one of the 2 classes, say class ω_1 and class ω_2 . Now, while training because all the feature vectors which are given, we know what is the class belongingness of those feature vectors. So, we effectively know that what expected output is.

So, if I take training sample, a training feature vector which we know that the training feature belongs to class ω_1 , we know that the corresponding output as to be equal 1 or if I know that the future vectors belong to class ω_2 , then it is known that the corresponding output has to be equal to 0. So, for that training sample or for each of the sample which is presented to the neural network, we know what the corresponding output.

And because initially, the weights are not set properly because we start with arbitrarily set weights of very low value. So, even if our training sample which belongs to the class ω_1 is given to the neural network, the output may be 0 or output may have any of the continuous value between 0 and 1. So, it may not be equal to 1, so what is the expected output and what is the actual output that you get the difference between that between these 2 is the error. So, if I say that my actual output that I obtain after giving a training sample, after inputting a

training sample is given by the expected output is say d and the actual output that I get D , which is nothing but the $\sum_{i=0}^N w_i x_i$, i value is from 0 to n .

(Refer Slide Time: 01:23)

The image shows a handwritten derivation on a blue background. At the top, it says "©CET I.I.T. KGP". Below that, the equation $D = \sum_{i=0}^N w_i x_i$ is written. Then, the squared error function $E = \frac{1}{2} (D-d)^2$ is shown. The derivative of the error with respect to w_i is calculated as follows:

$$\frac{\partial E}{\partial w_i} = (D-d) \cdot \left(\frac{\partial (D-d)}{\partial w_i} \right)$$

$$= (D-d) \cdot x_i$$

A red bracket groups the term $\frac{\partial (D-d)}{\partial w_i}$ and the term $(D-d) \cdot x_i$. This bracketed group is then equated to $\frac{\partial D}{\partial w_i}$, which is further simplified to $\sum_{i=0}^N w_i x_i$ and finally to x_i .

And then using the difference between these two, I can define what the error. So, if I define the squared error, the squared error is given by $E = \frac{1}{2} (D_p - d_p)^2$, D which is the actual output and d is the expected output or which is also called as target output. So, while training the neural network, the aim is that we should be able to set the weight vectors or connection weights in such a way that, this sum of squared error will be minimized.

So, for that again we can make use of the gradient descent procedure. So, for making use of the gradient descent procedure, we have to find out what is the gradient of the error with respect to the weight vectors. So, what I have to find out is $\frac{\partial E}{\partial w_i}$ for a particular weight vector.

And if I compute this term, you find that this is nothing but if I take the derivative of this with respect to w_i , this will be simply $\frac{\partial E}{\partial w_i} = (D - d) \left(\frac{\partial (D-d)}{\partial w_i} \right)$.

So, $\frac{\partial (D-d)}{\partial w_i}$, if I just concentrate on this particular term, because there was question on how do we get this.

So here you find that this lower case d is this. So, what I am trying to find out now is, what is $\frac{\partial(D-d)}{\partial w_i}$. So, as I said this lower case d, this is our expected output or target output so that is fixed.

Given a training sample as I know what is the class belongingness of training sample is so my target output is fixed, this does not depend upon the w_i .

So, $\frac{\partial(D-d)}{\partial w_i}$, this is nothing but $\frac{\partial D}{\partial w_i}$. And this capital D is nothing but $\sum_{i=0}^N w_i x_i$, where i varies from 0 to capital N, is that okay? So, you find that if I take the derivative of the summation with respect to w_i , then only term that remains is nothing but x_i because all of the terms are independent of w_i , only $w_i x_i$ that is dependent only on w_i and if I take the derivative it simply becomes x_i .

So, that is why this $\frac{\partial E}{\partial w_i}$ that we have obtained as $\frac{\partial E}{\partial w_i} = (D-d) \cdot x_i$. So, only this simple derivative expression, this has given me this particular expression. So, while the training the neural network as our aim to set the weight vectors. So, we have to go for a weight adjustment algorithm, that is what is neural network training? So, in this weight adjustment problem, initially will have to start with arbitrary weight vectors. So, you select with some random weight vectors for different w_i , you select some random values and the values are quite small. So, you generate small random values which are taken as w_i or I generate a random weight vector.

(Refer Slide Time: 10:58)

The image shows handwritten mathematical notes on a grid background. At the top right, there is a small logo with the text "©CET I.I.T. KGP". The notes consist of two lines of equations:

$$w_i(0) \leftarrow \text{random}$$
$$w_i(k+1) = w_i(k) - \eta (D-d) x_i$$

So, I have that $w_i(0)$, that is the initial weight vector which is set randomly, random but very small value. And then in every iteration $w_i(k+1)$ will be some update on $w_i(k)$ that was the value of w_i in the previous iteration. So, this will be $w_i(k) - \eta(D - d)x_i$ and our η represents what is the rate of convergence into this expression $\frac{\partial E}{\partial w_i}$, which is nothing but

$$\frac{\partial E}{\partial w_i} = (D - d).x_i. \text{ So, this will simply be } (D - d).x_i. \text{ So, how trained the neural network, you}$$

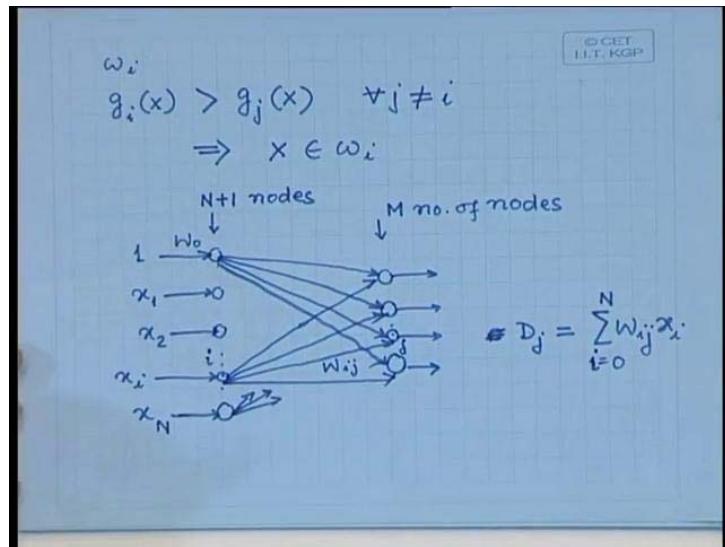
feed the training vectors one after another.

For every such training vector, you find out what is this component $(D - d).x_i$, which is the input vector component and this $w_i(k)$ was the previous vector. So, you modify the previous weight according to this update formula, so that you do this for every sample, with every sample for the weight will be modified. So, once the weight is modified for every sample, I do not know whether if you sample for which the weight was modified, because after modification with the previous unclassified sample, again the weight vectors are going getting modified treatment.

So, it may happen with the previous sample either the previous sample will be properly classified or it will be wrongly classified. So, if it is wrongly classified I have to modify the weight vectors again. And this iteration is continued, until and unless either all the training vectors are properly classified or I come to a situation where the error that is $(D - d)$, that error is quite small. It is within tolerance limit. So, the convergence criteria or the termination criteria of this iterative algorithm will be that either, in a complete pass all the training samples are properly classified or I get an error where is very small, the error that I can tolerate.

So, that will be my termination criteria or convergence criteria for this training algorithm. So, with the single layer single output neural network, we have considered only two class problem, that is if the output is greater than threshold or if the output is greater than zero, I say that sample belongs to one class. If the output is less than zero, I say that the sample belongs to another class. So, the kind of thresholding which we are applying is a hard threshold, but when you have multiclass problems, naturally the multiclass problems cannot be taken care of by a single output. So, I have to have multiple outputs. So, that gives you multiple output single layer neural networks and the action of this will be similar to all linear machine.

(Refer Slide Time: 14:35)



In case of linear machine, we have said that for every class ω_i , I have a discriminant function $g_i(X)$ and we said that if $g_i(X) > g_j(X)$ for all $j \neq i$. So, this indicates that the feature vector X belongs to class ω_i , that is for every class, I have a discriminant function for the given unknown feature vector X , you compute the discriminant function for every class. And whichever class gives you the maximum value of the discriminant function, you assign this unknown feature vector X to the corresponding class. Right?

So, if I want to have this similar kind of action using this neural network, then obviously I have to have a neural network with multiple numbers of outputs. So, the input layer as before will remain same that is the number of neurons in the input layer will be same as the dimensionality of the feature vectors of class 1, so that additional node takes care of the bias weight. And in the output layer, we have a number of nodes, which is equal to the number of classes that we have say if there are m numbers of classes, then in the output layer I have to have M number of nodes. And here I have $N+1$ number of nodes where N is the dimensionality of the feature vector.

So, you apply the feature vectors to this inputted nodes, as you have assumed the first component is equal to 1, that takes care of the bias weight w_0 other components are x_1, x_2 up to say here this x_i and here it is x_N . Now, the connection is like this that for every input layer node, now the output of every input layer node will be connected to the input of every node in the upper layer

So, the connection will be like this. Let us consider one node over here this is j node.

So, similarly it will be like this. Similarly, this will also be connected to every node in the output layer. And these are the outputs from every output of the output layer node. Now, the output of the j node suppose, I call it j^{th} node in the output layer. Output of the j^{th} node in the output layer will be nothing but if I represent this by say D_j , which will be equal to $\sum_{i=0}^N w_{ij}x_i$,

where i varies from 0 to M . And what is w_{ij} ? w_{ij} is the connection weight from the output of the i^{th} layer in the input node to the input of the j node in the output layer.

So, when this is my i^{th} node in the input layer and this is the j node in the output layer, this connection weight from the i^{th} node in the input layer to the j^{th} node in the output layer is w_{ij} , right? So, given this sort of connection and the sort of network topology, here training of the neural network means that I have to select all the connection weights w_{ij} for all values of i and all values of j . And unlikely in case of single layered network and single output neuron where the output was of scalar value, it was not a vector, it was a single value either 0 or 1.

In case of multiple outputs, the output will not be a scalar, but the output will be vector because I have capital M number of components. All the output layer nodes will give me some output, so that means output was also a vector not a scalar. However, for known feature vector that is during training as well all the feature vector for which the class belongingness is known. So if I know that a feature vectors belongs to class ω_3 , I know my target output should be only the third neuron in the output layer should give me a value 1. All other neurons and output layer should give me value 0.

If the feature vector belongs to class ω_1 , then only the first output layer neuron should give me an output 1, all other output layer neurons should give value 1. So, my target feature vectors will have only one of the components to be equal to one, the rest of the components should be equal to 0. That is my target. However, as initially the weight vectors are not properly set, I may get an output vector, which is not the same as target vector. So, accordingly the error that I get is a vector difference, the difference between 2 vectors, the target vector and the actual vector. Is it okay?

(Refer Slide Time: 21:25)

The image shows handwritten mathematical notes on a whiteboard:

$$E = \frac{1}{2} \sum_{j=1}^M (D_j - d_j)^2$$

$$\frac{\partial E}{\partial w_{ij}} = (D_j - d_j) \cdot x_i$$

$$w_{ij}(0) \leftarrow \text{random}$$

$$w_{ij}(k+1) = w_{ij}(k) - \eta (D_j - d_j) \cdot x_i$$

Training Algorithm.

So, accordingly we have to define the sum of squared error, which in this case, we can define

as $E = \frac{1}{2} \sum_{j=1}^M (D_j - d_j)^2$. So, this D_j means the actual output of the j^{th} neuron in the output layer

whereas this lower case d_j , this indicates that what should be the value of the output of the j^{th} neuron in the output layer in my target vector. And because I have M number of components in output vectors, take the summation for $j = 1$ to M as I have M number of components in the output vector.

Now, given this, here again you find that for the training algorithm or for learning of the neural network I have to take the gradient of this with respect to w_{ij} . Again following the

similar derivative to procedure I can find out that this $\frac{\partial E}{\partial w_{ij}}$ will be simply being $(D_j - d_j) \cdot x_i$.

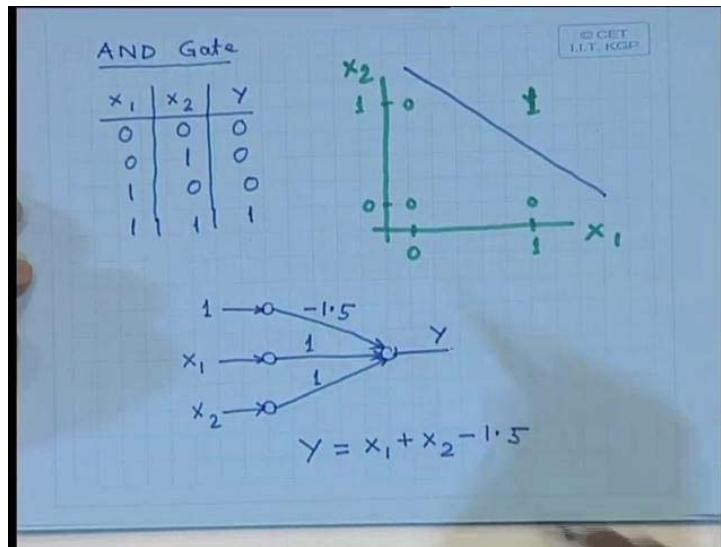
Where x_i is the i^{th} component of the input feature vector. I will get the same, with following the same procedure I get this. So, once I get this, now my training algorithm before will be $w_{ij}(0)$, that is the initial weight.

You have to set arbitrary or this will be random, but a small value and while iteration, this $w_{ij}(k+1) = w_{ij}(k) - \eta (D_j - d_j) \cdot x_i$, η is nothing but our rate of convergence. This decides rate of convergence. So, this is what the training algorithm. So, as before the training algorithm being an iterative procedure, I have to decide what is the convergence criteria. So, again here the convergence criteria can be either, all the training samples are correctly classified by a

particular set of weight vectors or I reached a condition where the error values will be small that is the error is tolerable.

Now, if you look at the discriminant function that this kind of neural network actually imitates, this is the linear discriminant function, because it is nothing but $\sum_{i=0}^N w_{ij}x_i$ for all values of i varies from 0 to M. So, it is a linear equation for fixed weight vectors. So, that means for every neuron in the output layer node, actually gives me linear equation of the straight line or an equation of a hyper plane. So, that simply says that if the classes are linearly separable, I can design the decision boundary using this sort of neural network, but if they are not linearly separable a single layer neural network cannot give me the decision boundary.

(Refer Slide Time: 25:53)



So, let us take some few examples from say binary arithmetic. So, let us take the case of an AND gate. So, all of you know the truth table of an AND gate. If I have two variables x_1, x_2 and I have this output y, the truth table of the AND gate is simply

x_1	x_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

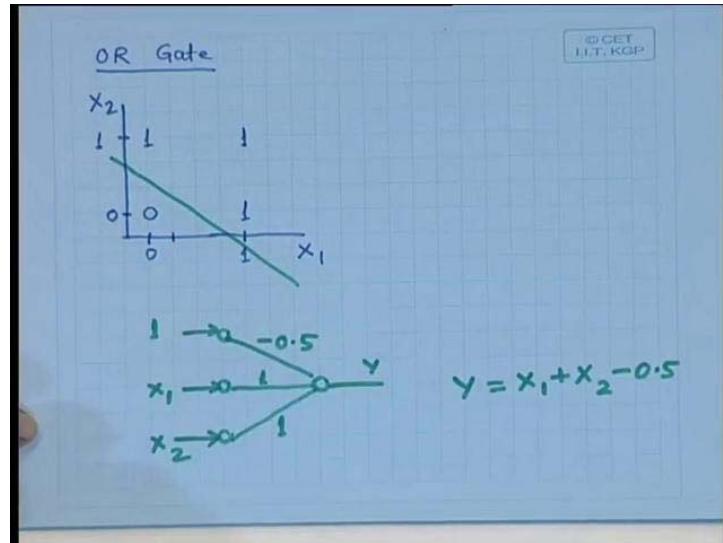
I can consider these to be a 2-dimensional binary feature vector, x_1, x_2 to be 2 dimensional binary features, which can assume one of these four different values. And I want to design a neural network, which will give me the function of an AND gate, that means output should be equal to 1, only when both the feature vectors, both the feature components are equal to 1 and output should be 0, if any or both of the input feature components are equal to 0.

So, if I plot it in the form of a graph something like this. This is my x_1 dimension, this is my x_2 dimension and this is the origin. So, for (0, 0) I take 0 some over here, this is 0 and somewhat here it is 1. So, if x_1 is 0 and x_2 is 0, so the output should be equal to 0, if x_1 is 0, x_2 is 1, then also output should be equal to 0, if x_1 is 1, x_2 is 0 then also output should be equal to 0, if x_1 and x_2 both of them are 1 then output should be equal to 1. Something like this, is it okay? So, you find that given this sort of the outputs, I can define a boundary line somewhere over here.

So, this is my classifier, right? And given the sort of classifier, you find that I can have a single layer, single output neural network, which will give me this sort of decision boundary. And one of the possible neural networks that can give me the sort boundaries is like this. I will have three inputs and one output, so one of these three inputs will have input is equal to 1 and the other inputs will have my feature components x_1 and x_2 . Right? The connection weights can be something like this, I put -1.5 here, 1 here and 1 here and this is my output y . Right?

So, find that this Y actually, gives me the equation of the straight line which is nothing but $Y = x_1 + x_2 - 1.5$. Right? So, here find that if any of the components or both of the components are 0, output is -1.5. If any of the components is 0 the other one is 1, then the output is -0.5. Now, in all three cases output is less than 0, whereas if both this component are 1 then output is 0.5 which is greater than 0. So, whenever this $Y > 0$, I can set output to be equal to 1, whenever $Y < 0$, I can set output to be equal to 0, whenever $Y = 0$, I can set output is equal to one. So, this simple neural network can perform the function of an AND gate.

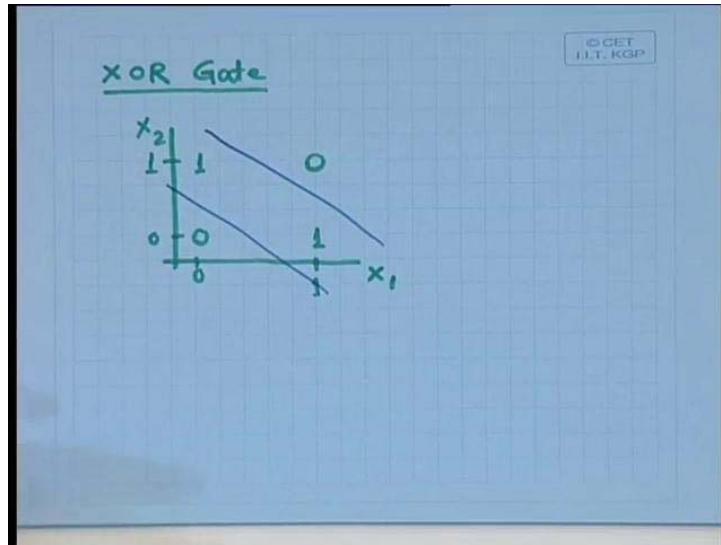
(Refer Slide Time: 30:32)



Similarly, if I take another example that is an OR gate. In case of an OR gate, the similar plots should be something like this, that is this is x_1 , this is x_2 . I put 0 here, 0 here, 1 somewhere over here, 1 somewhere over here. So, in case of OR gate, you find that only when both the inputs are 0, the output is 0; if any of the input is 1 or both the inputs are 1, in that case output is 1. So, I have 1 here and 1 here and I will have one here. Again in this case, I can define a straight line, which will separate between these 2 regions. And again a simple neural network like this with three input layer nodes and one output layer node, where this y again gives me linear equation, which is nothing but $x_1 + x_2 - 0.5$.

So, here you find that both x_1 and x_2 both of them are 0, the output is -0.5 which is less than 0. In all other cases if any or both of them becomes 1, the output becomes greater than 0. So, following the same logic if the output is greater than 0, I put it to 1, if the output is greater than 0, I put it to 0. Then you find that this particular neural network will give me an OR function.

(Refer Slide Time: 32:50)



Now, if I take another gate of similar nature that is X-OR gate. In case of X-OR gate, if I plot similar functions x_1 , x_2 , I put 0 here, 0 here, 1 here and say 1 here. In case of X-OR gate if the inputs are 0 0, the output is 0, if the inputs are 1 1 then also output is 0, if the inputs are 0 1 or 1 0 then only output is 1. So, I will have a 1 here and I will have 1 here. Now, given this sort of situation, you find that I cannot draw a single straight line which will separate this space where in one of the half space, the output 0 and the other half space, the output is 1.

Effectively, what I need is I have to define two straight lines here so that I can put the 0 output in one region and the output 1 to be in another region. So, this is very simple example and I cannot have the equation of a single straight line, which can divide this space into such a sort of region. So, naturally of single layer neural network cannot give me a solution for a simple problem something like this, where the region cannot be divided into 2 half spaces using linear equation.

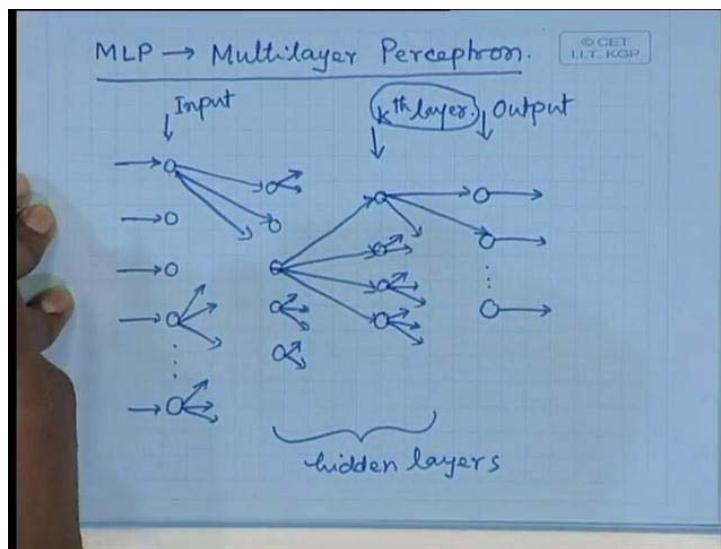
So, exactly for this kind of situation where the region is not linearly separable, I have to go for multilayer neural network, the single neural network does not work in these type of cases. So, I had to have multiple layer neural network where one of the layer is obviously the input layer, which we do not consider in deciding the number of layers and the other layer is the output layer. In case of single layer, we have the output layer and in between I have to have a number of hidden layers.

And it is observed that if you have 2 hidden layers, 1 or 2 hidden layers, two at the most that is sufficient for most of the practical problems. So you should go for multilayer neural

network with one output layer and 2 hidden layers or if the problem is simpler, we can have one output layer and one hidden layer. The number of nodes in the input layer will be same as the dimensionality of the feature vector plus 1, the number of nodes in the output layer will be same as the number of classes, for which we are designing the classifier.

And the number of nodes in the hidden layers those are actually variable. And so far there is no formal way in which you can decide that how many number of nodes in the hidden layer, which we have to have that type of problem. So, it depends upon how complex the decision boundaries are among the different classes. So, in case of this multilayer neural network, this is also called multilayer perceptron or MLP.

(Refer Slide Time: 36:32)

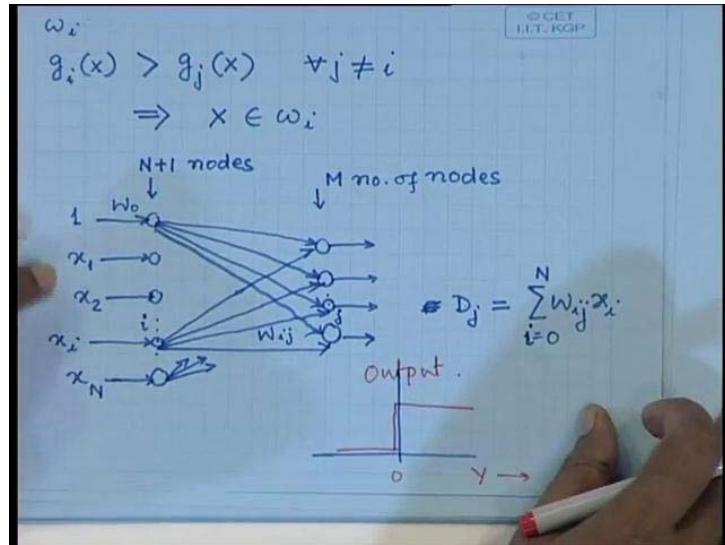


This is multilayer perceptron, we have obviously one layer which is input layer, we have one layer which is the output layer and we have either 1 or 2 hidden layers like this. So, this is my input layer, this is the output layer and these 2 are the hidden layers. So, as before the outputs from the k^{th} layer goes to the inputs of the $k+1$ layer. And every neuron, output of every neuron in the k^{th} layer is connected to the input of every neuron layer in the $k+1$ layer, so I have put it like this.

So, similarly over here, so all of them are connected. Similarly, here and finally, you get the output from the output layer nodes. Is that okay? So, when I consider any layer say k^{th} layer. Now, find that the number of weight vectors that we have is much more than the number of weight vectors that we had in earlier cases, because from the nodes of every layer to the nodes of the upper layer I have set of weight vectors. And the number of weight vectors or

the sets of weight vectors that we have depends upon how many layers you have in this neural network. So, if I consider say node, and not only that each of the nodes, in all the layers except the input layer gives a non-linear function, employs a non-linear function, which earlier case we said that this nonlinearity was hard non-linearity or threshold non linearity. That is, over there are non-linear function in this cases were something like this.

(Refer Slide Time: 39:58)



The non-linear function that we had was of this form that is, if $Y < 0$ then output is 0, if $Y > 0$ then output is 1. So, this is my zero line and this is Y and this is the output. So this was also a non-linear function, but this nonlinearity was harder nonlinearity. Whereas in case of multilayer neural network, the nonlinearity which is used is not a hard non-linearity, but it is a sort of sigmoidal function.

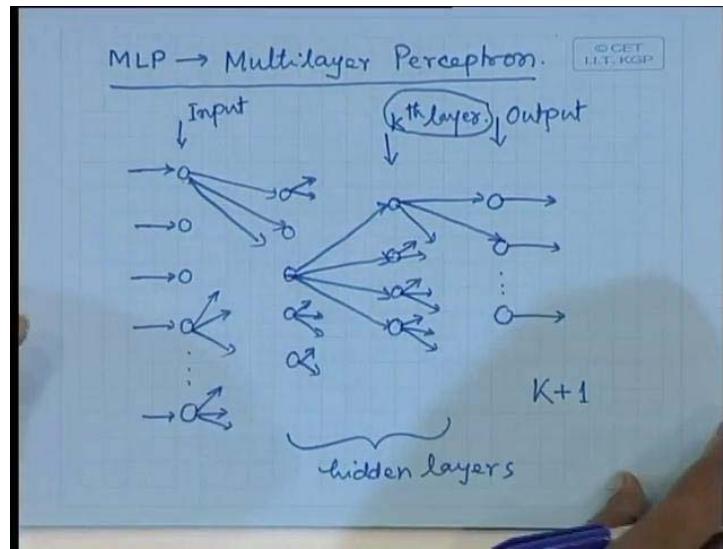
(Refer Slide Time: 40:46)

The diagram shows the sigmoid function $R(s) = \frac{1}{1 + e^{-s}}$ plotted against s . The curve is labeled $R(s)$ and passes through the point $(0, 0.5)$. The derivative of the sigmoid function is given as $\frac{\partial R(s)}{\partial s} = R(s)[1 - R(s)]$.

And the sigmoid function is given by this expression $R(s) = \frac{1}{1+e^{-s}}$, s is a argument, then R is a sigmoid function. And if you plot this $R(s)$ versus s , this is the s and this is $R(s)$, the plot will be like this. Where you find that when $s \rightarrow -\infty$, $R(s)$ tends to be 0 because as s tends $-\infty$, this e^{-s} that tends to be ∞ . So, $\frac{1}{1+e^{-s}}$ tends to be 0. Similarly, as $s \rightarrow \infty$, then this $R(s)$ tends to be equal to 1.

So, I have the output function which is limited between 0 and 1, as $s \rightarrow -\infty$ $R(s)$ tends to be 0, as $s \rightarrow \infty$ $R(s)$ tends to be 1 and at s is equal to 0, the value of $R(s)$ is equal to 0.5. So, it is a symmetrical non-linear function around the values $s = 0$. The advantage that we get by using this type of nonlinearity are that, this function is differentiable. In case of hard nonlinearity because I have a discontinuity at $s = 0$, it is not differentiable. Whereas this function is a continuous function, it is differentiable and not only that, if I take the differentiation of $R(s)$ $\frac{\partial R(s)}{\partial s}$, that gives us a very simple form, this is nothing but $R(s)(1-R(s))$. If you differentiate this $R(s)$ with respect s you will get this sort of differential output, right?

(Refer Slide Time: 43:09)



So, here I assume that in this multilayer neural network, I have total capital $K+1$ number of layers. One of them is input layer and I have capital K number of layers including the hidden layers and the output layers. So, the total number of layers is capital $K+1$. And when you say, we say that it is K layer MLP, where k indicates the number of layers excluding the input layer.

Now, given this sort of situation, I have to when I talk about the input to a particular node or when I talk about output to a particular node, I also have to identify that which layer it is. So, I have to identify which layer and at the same time, which neuron in that particular layer. So, if I say the output of the k^{th} layer neuron or say the i^{th} node in the k^{th} layer neuron.

(Refer Slide Time: 44:22)

$$x_i^{(k)} \rightarrow \text{Output of } i^{\text{th}} \text{ neuron in } k^{\text{th}} \text{ layer.}$$

$$x_j^{(k+1)} = R \left(\sum_{i=0}^{M_k} w_{ij}^{(k+1)} x_i^{(k)} \right)$$

Then that will be represented by $x_i^{(k)}$, this is output of i^{th} neuron in k^{th} layer. Let me put it as lower case k , so it is a variable. So, we have said that capital K is number of layers. So, $x_i^{(k)}$ as k superscript and i subscript, this represents output of i^{th} neuron in the k^{th} layer. And obviously this output is feed as an input to a node in the $(k+1)^{\text{th}}$ layer. So, when I talk about the output of the $(k+1)^{\text{th}}$ layer or the output of the j^{th} neuron in the $(k+1)^{\text{th}}$ layer, that I can write as $x_i^{(k+1)}$.

This is output of the j^{th} neuron in the $(k+1)^{\text{th}}$ layer, which gets input this sort of input for different values of i . So, this I can write as $x_i^{(k+1)} = R \left(\sum_{i=0}^{M_k} w_{ij}^{(k+1)} x_i^{(k)} \right)$ because from every i^{th} node in the k^{th} layer, this j^{th} node in the $(k+1)^{\text{th}}$ layer gets input. So, the w_{ij} and between which 2 layers so I write it as $k+1$ indicating that it is the connection weight between the k^{th} layer neuron and the $(k+1)^{\text{th}}$ layer neuron. So, this is w_{ij} and between which two layers so I write it as $k+1$ indicating that it is the connection weight between the k^{th} layer neuron and the $(k+1)^{\text{th}}$ neuron.

And $i = 0$ indicates that, if I want to incorporate any bias in the k^{th} layer also, I can have bias only in the input layer, I can also have bias in every layer as well. So, M_k is the number of nodes in the input layer and plus 1, so the total number of nodes in the input layer is $M_k + 1$. So, I set $i = 0$ to M_k and on top of this, I have to have this non-linearity.

So, the whole thing will be a non-linear function of this, so R of $\left(\sum_{i=0}^{M_k} w_{ij}^{(k+1)} x_i^{(k)} \right)$. So, output of the j^{th} neuron in the $(k+1)^{\text{th}}$ layer is the weighted sum or non-linear function of the weighted sum of outputs of all the neurons in the k^{th} layer where, this weight is given by the corresponding connection weight $w_{ij}^{(k+1)}$. And this is valid for all the hidden layer nodes and the output layer nodes. So, every node in every layer gets outputs from the nodes in the previous layer, except the input layer node.

To the input layer node, we directly feed the inputs and the output of the output layer nodes, that we get directly as outputs that is not fed to the neurons of any other layers. So, given this sort of formulation, or this sort of architecture of the multilayer neural network, here again I have to find out what will be my training procedure. So, to do the training procedure, I follow the similar type of approach that is the gradient descent approach. So, for that I have to find out what is the error I get at the outputs of every layer.

(Refer Slide Time: 49:12)

The image shows a handwritten derivation of backpropagation equations for a neural network layer. At the top, the error function E is defined as:

$$E = \frac{1}{2} \sum_{j=0}^{M_k} (x_j^{(k)} - d_j^{(k)})^2$$

Below it, the partial derivative of the error with respect to the weight $w_{ij}^{(k)}$ is shown:

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = (x_j^{(k)} - d_j^{(k)}) \cdot \frac{\partial x_j^{(k)}}{\partial w_{ij}^{(k)}}$$

Further down, the derivative of the activation function $x_j^{(k)}$ with respect to its weight $w_{ij}^{(k)}$ is derived using the chain rule, involving the function R :

$$\begin{aligned} \frac{\partial x_j^{(k)}}{\partial w_{ij}^{(k)}} &= \frac{\partial}{\partial w_{ij}^{(k)}} R \left(\sum_{i=0}^{M_{k-1}} w_{ij}^{(k)} x_i^{(k-1)} \right) \\ &= x_j^{(k)} \cdot (1 - x_j^{(k)}) \cdot x_i^{(k-1)} \end{aligned}$$

So, finally, when I am saying that capital K indicates my output layer. Let us see that what is the error at the output layer. This function R , every neuron except the input layer, in input

layer it is unity. Just linear in an input, whatever is the input, to pass it to output, but in all other layers this nonlinearity is present. Which in case of single neuron, we said that this nonlinearity is present only in the output layer nodes not in the input layers node and there we have considered the nonlinearity to be hard threshold d .

Here we are considering this nonlinearity to be a sigmoidal function, it is not a hard threshold. There is reason for it. No, that is what I am saying, we will come to realize that, why this nonlinearity we are putting. We can do definitely because that is also linear, but the problem was there is if you do not put any such limit, then your network may have become saturated. So, that is why you put the limit at every neuron. And we are not putting the hard threshold, but we are putting this continuous non-linearity, that is will be obviously we talked about the training algorithms.

Student: Sir, number of nodes in hidden layer that may vary layers to layers?

May vary from layer to layer.

Student: On what factors it may vary?

That is what I said, there is no analytical approach to find out what is how many number of layers you need or the how many nodes in every hidden layers you need, there is no such analytical factors.

If we increase the numbers means the quality of output, up to some level the quality of output will increases beyond that the quality may even fall. This nonlinear function is different for different layers. It can be different for different layers or it can be same as well. The usual practices you use the same non-linear function.

What is needed is in each of this layers the nonlinearity that you introduced, the nonlinearity should be a continuous function or it should be a derivable function or it should be able to find its derivatives, its derivatives should exist, that is the constraint. Why as I said that will be clear, when I talk about the training algorithms. Why we want that this function should be differentiable function?

So, what is the error, let us assume that in k^{th} layer, as before the sum of squared error E will define as $E = \frac{1}{2} \sum_{j=0}^{M_k} (x_j^{(k)} - d_j^{(k)})^2$, take the summation for j varying from 0 to M_k , where this is

the number of nodes in the k^{th} layer. Now, what is this $x_j^{(k)}$? This $x_j^{(k)}$ is of this form. Here in place of $k+1$, you would just take k and accordingly modify this expression. So, that is what is $x_j^{(k)}$, that is what I get from the j^{th} neuron in the k^{th} layer.

So, to apply my training algorithm following the gradient descent procedure I have to take the derivative of this with respect to $w_{ij}^{(k)}$. So, what to find out is, I have to take $\frac{\partial E}{\partial w_{ij}^{(k)}}$, what will be this? Again following the similar derivative approach, this will be simply $(x_j^{(k)} - d_j^{(k)}) \cdot \frac{\partial x_j^{(k)}}{\partial w_{ij}^{(k)}}$. Now, here we find that, why I am putting this $\frac{\partial x_j^{(k)}}{\partial w_{ij}^{(k)}}$, which I did not put because of single layer.

The reason is this $x_j^{(k)}$ is again a weighted sum of the output from the neurons from $(k-1)^{\text{th}}$ layer. And outputs of the $(k-1)^{\text{th}}$ player are actually weighted by the corresponding $w_{ij}^{(k)}$. So, I have to do this and not only that this $x_j^{(k)}$ is now on non-linear function of the weighted sum of the outputs of the $(k-1)^{\text{th}}$ layer neurons. So, when I take this gradient, this gradient has to take care of the nonlinearity as well, right? So, when I compute this, as I said earlier that the nonlinearity that I have is of this form $R(s) = \frac{1}{1 + e^{-s}}$.

And if I take the derivative of this with respect to s , the derivative is like this $R(s)(1 - R(s))$.

So, if I concentrate on this term that is $\frac{\partial x_j^{(k)}}{\partial w_{ij}^{(k)}}$ so this will be nothing but $\frac{\partial R}{\partial w_{ij}^{(k)}} \left(\sum_{i=0}^{M_{k-1}} w_{ij}^{(k)} x_i^{(k-1)} \right)$.

Where this summation will be for $i = 0$ to M_{k-1} . Is that okay? So, if I take this directive you will find that, this will be nothing but $x_j^{(k)}(1 - x_j^{(k)}) \cdot x_i^{(k-1)}$. Is that okay? So, if I put this in this expression.

(Refer Slide Time: 57:14)

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = (x_j^{(k)} - d_j^{(k)}) \cdot x_j^{(k)} \cdot (1 - x_j^{(k)}) \cdot x_i^{(k-1)}$$

What I get is $\frac{\partial E}{\partial w_{ij}^{(k)}} = (x_j^{(k)} - d_j^{(k)}) x_j^{(k)} (1 - x_j^{(k)}) x_i^{(k-1)}$. So, this will be the derivative of the error

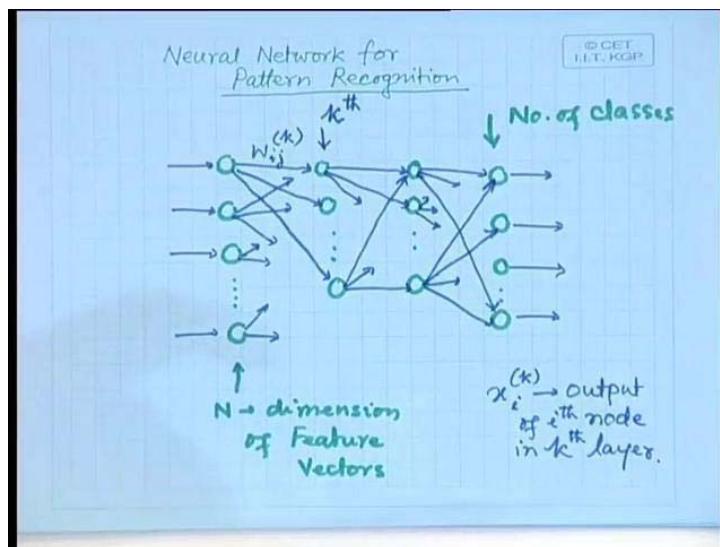
with respect to $w_{ij}^{(k)}$. So, naturally when I go for the iterative training algorithm, I have to use this derivative for updation of the weight vectors and this has to be done from layer to layer. And in case of single layer output, it was only one layer now I have multilayer perceptron so this weight updation has to be done from layer to layer. So, let us stop here today, we will continue it in the next class.

Thank you.

Lecture - 26
Neural Networks for Pattern Recognition
(Contd.)

Good morning. So, in the last class we have started our discussion on multi-layer neural network or multilayer perceptron, which is used for pattern recognition purpose. And we were talking about how to adjust the vectors or how to train the neural network, so that it is ready for the recognizing the unknown feature vectors. So, what we were talking about is something like this.

(Refer Slide Time: 00:47)



So, we have multilayer neural network with a number of input nodes, and the network has a number of output nodes, where the number of input nodes is same as the dimensionality of the feature vectors. So, it is dimension of feature vectors and at the output we have say the number of nodes which is same as number of classes. And we have one or more hidden layers and hidden layers will also have a number of nodes, and typically as we said the number of hidden, if we have say around 2 hidden layers. So, total we have 4 numbers of layers, including the input layers and output layer that is mostly sufficient for most of the practical applications.

And what you do is, you feed the feature vector to the input layer and get the output from the output layers and every node of the input layer is connected to, is feeding the input to every node in the layer above it. So, it goes like this, with this similarly, here. So, if I consider a k^{th} layer node, the output of the i^{th} node, in the k^{th} layer is represented by $x_i^{(k)}$. So, this is the output of i^{th} node in k^{th} layer and the connection weights from an i^{th} node in $(k-1)^{\text{th}}$ layer to a j^{th} node in the k^{th} layer is represented by, the $w_{ij}^{(k)}$. So, given this and as we said that for training the neural network as we are inputting all the training feature vectors and we know that, what is the class belongingness of the training feature vector.

So, from that node is I know that, if I feed a known vector or training vector as input to the neural network, I know what we will be the corresponding output, because for every training vector, its class belongingness is known. So, if I get any vector from the output, which is not same as the expected output then the difference between these two vectors is the error vector. And from that we compute, the squared error and training of the neural network aims to adjust the vectors in such a way that, the sum of squared error that is minimized. So, what we have assumed is say, we have capital K number of layers so that, the output layer is labeled as capital K^{th} layer.

(Refer Slide Time: 05:05)

$$E = \frac{1}{2} \sum_{j=1}^{M_K} (x_j^{(K)} - d_j^{(K)})^2$$

$$\frac{\partial E}{\partial w_{ij}^{(K)}} = (x_j^{(K)} - d_j^{(K)}) x_j^{(K)} (1 - x_j^{(K)}) \cdot x_i^{(K-1)}$$

$$w_{ij}^{(K)}(t+1) = w_{ij}^{(K)}(t) - \eta \underbrace{(x_j^{(K)} - d_j^{(K)})}_{S_j^{(K)}} \underbrace{x_j^{(K)} (1 - x_j^{(K)})}_{x_i^{(K-1)}} \cdot x_i^{(K-1)}.$$

So, given that the sum of squared error is defined to be $E = \frac{1}{2} \sum_{j=1}^{M_K} (x_j^{(K)} - d_j^{(K)})^2$, where this K is the capital K , indicating that this K is the output node, $d_j^{(K)}$ that is the target output,

expected output of the j^{th} node in the k^{th} layer. Take the square of this, where this j will vary from say 1 to M_k , where M_k is the number of nodes in the K^{th} layer. And we have seen that for a designing or for deciding, adjusting the weight vectors we will follow the gradient decent approach and have to take the derivative of this, with respect to $w_{ij}^{(k)}$. And using that derivative we have to for, adjusting the weight vector following the gradient descent approach.

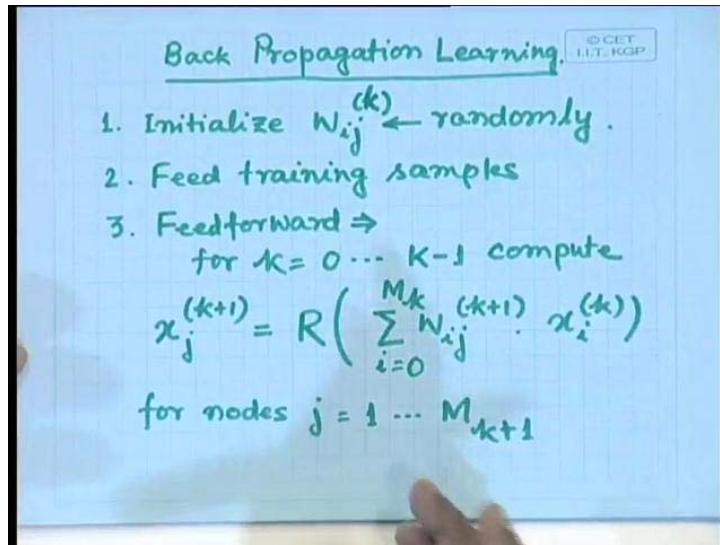
So, what we have done is taken the gradient of this, $\frac{\partial E}{\partial w_{ij}^{(k)}}$ and we have seen that, this derivative comes out to be $(x_j^{(k)} - d_j^{(k)})x_j^{(k)}(1 - x_j^{(k)})x_i^{(k-1)}$. So, this is what we have seen in the last class and following this, the weight updation rule, will be of this form $w_{ij}^{(k)}$ in iteration say, $t + 1$ from the weight vector in iteration t . This will be simply, $w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) - \eta(x_j^{(k)} - d_j^{(k)})x_j^{(k)}(1 - x_j^{(k)})x_i^{(k-1)}$. This is the weight updation rule. Now, over here, for simplicity of expression, this particular term $(x_j^{(k)} - d_j^{(k)})x_j^{(k)}(1 - x_j^{(k)})$, this we can represent as $\delta_j^{(k)}$.

So, our weight updation rule will be $w_{ij}^{(k)}$ in iteration $(t+1)$, will be $w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) - \eta\delta_j^{(k)}x_i^{(k-1)}$. So, you find that, for adjusting the weights what you do is, you start the error at the output layer, start with error at output layer, take the gradient of it with respect to the $w_{ij}^{(k)}$ then you adjust all the weights, connection weights from the $(k-1)^{\text{th}}$ layer to the k^{th} layer. Then what ever error you get in that layer, using that you adjust the second layer, $(k-2)^{\text{th}}$ layer to $(k-1)^{\text{th}}$ layer. Then you come to $(k-3)^{\text{th}}$ layer and so on. So, we find that we have an information flow in two directions, one is from input side to output side, when you go for the computation of the outputs of different nodes.

And secondly for weight updation, what you do is to basically propagate the error term from output layer to the input layer. So, that is why, this sort of network is called as feed forward and back propagation network. And the corresponding neural network learning is called as back propagation learning. So, your vector information of input, vector information or output computation proceeds, from the input layer to the output layer. Whereas, while training the weight updation that proceeds from the output layer to the input layer that is, in the backward

direction. So, this is called feed forward, but back propagation network, so accordingly the learning that is also called back propagation learning.

(Refer Slide Time: 10:26)



So, let us see that, what will be the steps in the backward propagation learning. Obviously in the first step, I have to initialize $w_{ij}^{(k)}$ for every weight vector, for every vector connection, for every node i, for every node j and for every layer k. And this initialization, has to be made randomly so it set or initialize the weight vector $w_{ij}^{(k)}$ randomly.

So, after we initialize the weight vectors, the next step will be that, I have to feed training vectors or training samples to the input layer. So, once I feed the training samples, then using these training samples and whatever the random weights, that has been initialized, using that I have to compute the outputs from all the nodes, from all different layers. So, the next step, will be the feed forward pass and in feed forward pass, I have to compute the outputs of every neuron in every layer. So, for the $k = 0$ to capital $K - 1$, I have to compute, I have to compute

$x_j^{(k)}$, which is nothing but $R \left(\sum_{i=0}^{M_k} w_{ij}^{(k+1)} x_i^{(k)} \right)$, where $M_k + 1$ is the number of nodes in the k^{th}

layer because one of the node will be used to take care of the bias weight. So, this I have to compute for every node, $j = 1$ to $M_k + 1$, one of the other nodes will have a constant input.

So, this is what I have to do for in the feed forward pass, where I am computing the outputs from every node, in every layer depending upon, the weights that I have till that time and what is the input vector that is fed in. So, once all these outputs are computed. Now, you come to the output layer node at the output layer node, I know what is the target output and what is the output that is computed from here. So, from these two, I will get the error and using this error. Now, you follow the back propagation part, the back propagation procedure to keep on adjusting the weights so that, error is minimized.

(Refer Slide Time: 14:55)

4. Back Propagation:

© CSET
I.I.T. KGP

For nodes in the output layer
 $j = 1 \dots M_K$ compute

$$\delta_j^{(k)} = x_j^{(k)}(1 - x_j^{(k)}) (x_j^{(k)} - d_j)$$

For layers $K-1, \dots, 1$ compute

$$\delta_i^{(k)} = x_i^{(k)}(1 - x_i^{(k)}) \sum_{j=1}^{M_{k+1}} \delta_j^{(k+1)} w_{ij}^{(k+1)}$$

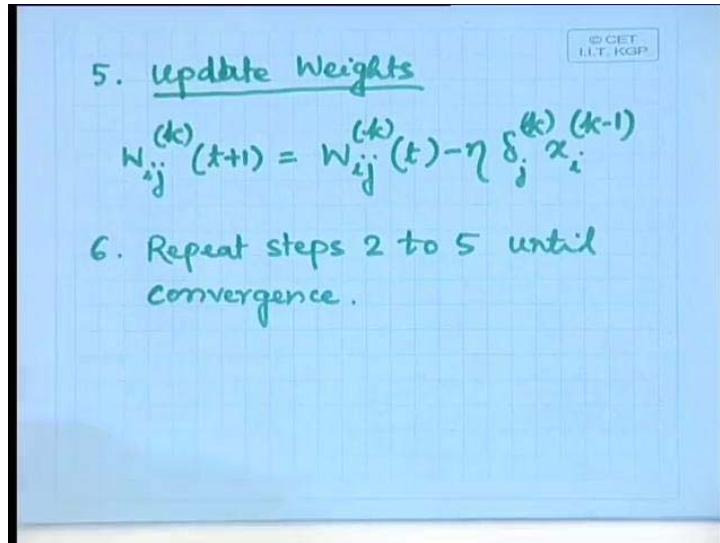
for $i = 1 \dots M_k$

So, the fourth step will be the back propagation part. So, in the back propagation what we do is, first we have to consider the output layer nodes. So, for nodes in the output layer, that is $j=1$ to M_K , where K , this K is capital. I have to compute $\delta_j^{(k)}$, which is equal to $x_j^{(k)}(1 - x_j^{(k)}) (x_j^{(k)} - d_j)$, which is exactly same as this term, $(x_j^{(k)} - d_j) x_j^{(k)} (1 - x_j^{(k)})$. So, this is what you do for the nodes in the output layer, but you find that computation of this term, for the hidden layer nodes has to be slightly different, because there the contribution to error is because of propagation from various other nodes.

So, for other nodes, for layers the $K-1$ up to layer 1, this computation will be slightly different. So, here we say, $\delta_i^{(k)}$ that will be is equal to $x_i^{(k)}(1 - x_i^{(k)}) \sum_{j=1}^{M_{k+1}} \delta_j^{(k+1)} w_{ij}^{(k+1)}$, I have to have this summation term because I have the contribution of various nodes leading to this particular error. And this have to compute for $i = 1$ upto M_k , where this k is lower case. Even

for the output layer at the output layer, I can directly compute there because I know what is the target output, but in case of hidden layers. I do not know what is the target output from hidden layers, because that is coming due to contribution from various other nodes. So, this direct computation, that have done at the output layer that is not possible in the hidden layers. I am not showing this derivation of this, but following the summation from various outputs and taking the derivative of this, you can find out that this $\delta_i^{(k)}$ will come out to, can be of something like this. Because at the output layer, I know what is the target output, but in the hidden layers I do not know what is the target output because it is getting combined from various nodes from the layer below that. So, once I have this delta terms then comes the weight updation part.

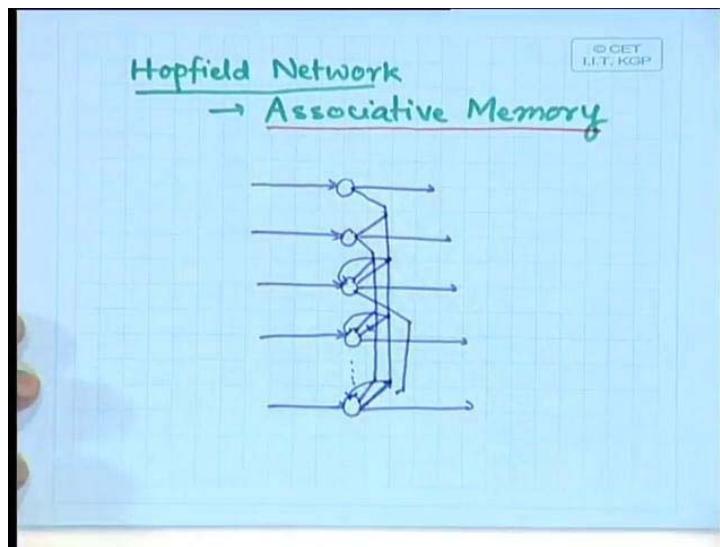
(Refer Slide Time: 19:16)



So, I have to update the weights and that weight updation is simply $w_{ij}^{(k)}$, that will be equal to $w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) - \eta \delta_j^{(k)} x_i^{(k-1)}$. So, why I have put this delta term is that, having different computations of this delta value, I can have a single expression for weight updation rule. And all these steps are to be repeated, so sixth will be repeat steps 2 to 5 until, convergence. So, what is our condition for convergence? The condition of convergence can be either all the training samples are correctly classified without the updation of the weight vectors, without the updation of the connection weights, in a single pass. Or the output error that I get, the output error is limited, it is up to a tolerance level.

So, if one of these two conditions is satisfied then we can say that the algorithms are conversed and whatever the connection weights we have at that point of time, the network has to work with those connection weights. So, that is what is the back propagation neural network and you find that the back propagation neural network, works with the continuous inputs that means, the input vector can be of any form, it can be a discrete output vector or a continuous output vector. There is another kind of neural network, which is also used for pattern recognition purpose and that neural network can be used for, recognizing the binary patterns.

(Refer Slide Time: 22:10)



So, this is what is called Hopfield network or this also works as, the weight works is something like an associative memory. Now, what is this associative memory, you find that suppose you have met one of their friends, one of your school level friend, at this age. Possibly after your 4th standard or 5th standard, you have not met him at all, right? And you find that during this period, there is been lot of changes. So, some Thomas who was your classmate, when you were in 4th standard, it is not the same Thomas, when he was aged around 20-22 or so. There has been lot of changes, but still you are able to recognize him, may be that you have forgotten his name that is ok, but still you can say that I know this person, I have met him somewhere before.

So, some sort of association you can do with some pattern that is stored in your memory. So, that is what is associative memory, so given a pattern even if the pattern is not

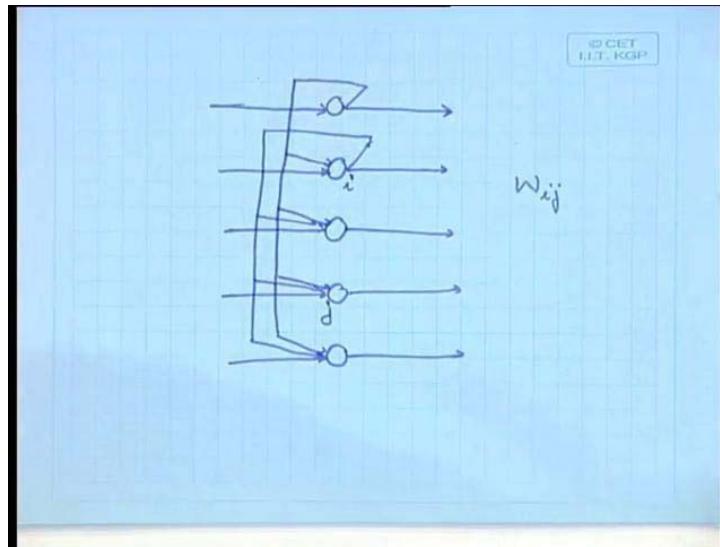
perfect, it is an imperfect pattern still, if you have a pattern stored in the memory and wants an imperfect pattern, imperfect form of the same pattern, or distorted form the same pattern is presented. Then you can recollect from the memory, the perfect pattern which has been stored, which is very close to this imperfect pattern. So, that is what is the concept of associative memory, I can associate an imperfect pattern with a perfect pattern, which is stored in memory.

So, here we are assuming that the pattern vectors or the feature vectors that we have are binary feature vectors and whenever I have a binary feature vector, the, normally by binary, what we mean is the different feature components will be 0's and 1's. So, in this particular case, everyone will be represented by +1 and 0 will be represented by -1, but still it is binary because only two digits, one corresponding to +1 and the other one is -1. So, the feature vectors that we have a binary feature vectors and suppose there are n number of components of the binary feature vectors, that means I have n bit patterns.

Now, the difference between the back propagation network or multi-layer perceptron and this case Hopfield neural network is that, in case of back propagation network, your information flows only from the input layer to the output layer. A node in the $(k - 1)^{\text{th}}$ layer will always feed to a node in the k^{th} layer, a node in the k^{th} layer will always feed to a node in the $(k + 1)^{\text{th}}$ layer, but a node in the k^{th} layer does not give information, does not feed a node in the $(k - 1)^{\text{th}}$ layer. So, that back propagation or backward information flow, only we use for training the network, but not for computation. In case of Hopfield network, output of every node is connected to the input of every other node.

So, it is something like this, see if I have n number of nodes, every node obviously has an input that is a binary pattern. They have outputs and output of every node is connected to the input of every other node. And this connection is bi directional that means, information flows in both the directions. So, similarly from here, sorry this goes to the input side not from the output side, so this connection will go to the input side.

(Refer Slide Time: 27:47)



Let me draw a fresh figure. So, like this output of every node is connected to the input of every other node and as before, the output of i^{th} node is connected to the input of say j^{th} node via a connection with w_{ij} . And this is the same connection with, which is used for connecting output of the j^{th} node to the input of i^{th} node. So, w_{ij} and w_{ji} are same for i and j will be same and what you do is by training as before, here what we are doing is, we are actually storing a number of patterns. A number of binary patterns in this associative network, or in this Hopfield network by adjusting the weights properly. So, it is the weights w_{ij} , which actually stores the patterns and when you feed an input, because it is a feedback network from every node, I get input to, I get input to every other node.

So, it is the feedback network and all the notes can be fired or can be activated asynchronous that is, in which order the nodes actually operate that is not specified. They can work in any order, so finally when you feed an input there will be a number of iterations, which occur asynchronously among all the nodes. And finally, when the nodes stabilize, when the network stabilizes the output vector that in, that will get it is one of the vectors, which is already stored in the network, by adjusting the connection weights, which is closest to the input vector, that we have given. So, when train the network, we train a network in the network using the perfect feature vectors, perfect vectors and while retrievable, even if I give input vector, which is not perfect.

The network will retrieve one of the patterns, which is perfect and stored and closest to the imperfect input vector that we have presented. So, that is how this Hopfield neural network

works and while changing the output, while retrieving a particular pattern the output of a network will be set to class one or it will be positive, if some of the weighted inputs is greater than 0 and it will be negative, if it is not greater than 0.

So, for less than or equal to 0, the output of the node will be negative, for weighted input, sum of weighted imports if it is greater than 0, the output is positive. So, you will find that during iteration the output of a node can vary many times, before it actually stabilizes because every change in output of one node is reflected to the input of every other node.

And as it is reflected to the input of every other node, so our change of output of one of the nodes may effectively change the output of some other nodes. And that change again may reflect the output of the previous node because this change is again reflected to the input of the previous node. So, the network will go through a number of iterations, iteratively and during the iterations output of the nodes may vary from + to - from + to - many times, before the network actually stabilizes.

And when it stabilizes, the output pattern that we get, that is the closest patterns, which is stored in the network and closest to the input pattern, that has been presented to this neural network. So, actually it is associating the input pattern to one of the stored patterns so that is why it is associative memory or associating network. Now, let us see that, how we can store the patterns in the weight vectors or the connection weights from the i^{th} node to the j^{th} node. So, when we set the weight vector w_{ij} I have to consider the, training patterns that has been given, that has been provided.

(Refer Slide Time: 33:21)

$$X^{(P)} = (x_1^{(P)}, x_2^{(P)}, \dots, x_n^{(P)})$$

$$p = 1, \dots, m$$

$$w_{ij} = \begin{cases} \sum_{p=1}^m x_i^{(P)} x_j^{(P)} & ; i \neq j \\ 0 & ; i = j \end{cases}$$

So, suppose I have this p^{th} training pattern $X^{(p)}$ and as I said that this $X^{(p)}$ of the training vectors, the sample vectors are n -dimensional vectors. So, it will have n number of components and let us say of those n number of components are x_1^p, x_2^p up to x_n^p . As I have n number of components in the feature vector, each of the x_1^p, x_2^p, x_n^p they can assume either a value +1 or a value -1, because they are binary patterns and our convention is that bit 0 will be represented by -1. So, every component can have a value either +1 or -1 and this is the p^{th} training vector, feature vector. And I have say m number of feature vectors that means this p , it can vary from 1 to m , as I have a m number of feature vector, or m number of training vectors.

Then, using this training vector, I have to compute the connection weights w_{ij} . So, w_{ij} is given by $\sum_{p=1}^m x_i^{(p)} x_j^{(p)}$, where this p varies from 1 to m for all $i \neq j$ and it will be equal to 0, if $i = j$.

So, that is how, you set w_{ij} , so what does it mean, you find that whenever $i \neq j$, your w_{ij} will be computed by this $\sum_{p=1}^m x_i^{(p)} x_j^{(p)}$. If $i = j$ then $w_{ij} = 0$, you find that when I have from the schematic of this Hopfield network. I have not feedback to the input of the same node. So, that means w_{ii} for $i = j$, is equal to 0 means, that a node does not feedback the signal to itself.

The output of a node is connected to the input of every other node, via this connection weights w_{ij} , but output of a node is not connected to the input of the same node. So, that is what it means, and from here you find that, what is this term, $x_i^{(p)} x_j^{(p)}$ you find that, both these components are coming from the same feature vector p . And the feature vectors are binary feature vectors having values +1 or -1, so this simply says, that if $x_i^{(p)}$ and $x_j^{(p)}$ both of them are same, both of them are positive or both of them are negative.

Then, this term will be equal to +1, if one of them is positive other one is negative then this term will be -1. So, overall this summation $\sum_{p=1}^m x_i^{(p)} x_j^{(p)}$, so this simply tells you that the number of times a bits are same minus the number of times bits are different. So, that is how and that is computed over, all the training patterns because it is $p = 1$ to m , it is computed over, all the training patterns and that is stored as w_{ij} , the connection between the i^{th} node and the j^{th} node. So, let us take a very small example.

Student: Sir is there is any rule that the connection should not be there from the upper nodes to the lower nodes.

No, there is no such rule I did not put it intentionally because I said that it is the same connection, which is used for bidirectional communication. Actually, in that diagram, second one should be connected to first. It should be there, output of the second one should be connected to the first one, from second one to the second one there should not be any connection, from first one to the first one there should not be any connection. But the connection weight will remain same the value w_{ij} is same as w_{ji} . Connection weight remains the same.

(Refer Slide Time: 38:39)

Example

$x^{(1)} = ++++++$ $x^{(2)} = +-+---$

$x^{(3)} = ++--$

$w_{12} = +1$	$w_{23} = +1$	$w_{45} = +1$
$w_{13} = +1$	$w_{24} = +1$	$w_{46} = +3$
$w_{14} = -1$	$w_{25} = -1$	$w_{56} = +1$
$w_{15} = +1$	$w_{26} = +1$	
$w_{16} = -1$	$w_{34} = -1$	
	$w_{35} = +1$	
	$w_{36} = -1$	

So, let us take a very small example to see how this weight vectors are really assigned. So, let us take that I have say, three feature vectors, one is x_1 and this feature vectors are given by +, +, +, +, +, +, a six-bit feature vector. The second one x_2 , which is given by so +, -, +, -, +, - and the third feature vector is x_3 , which is given by +, +, +, -, -, -. So, all the three are six-bit feature vectors. So, I have to compute the different connection weights, so what will be the value of the w_{12} , so here you find that I have to consider the first bit and the second bit. So, from the first feature vector both of them are positive.

So, when I compute this term $\sum_{p=1}^m x_i^{(p)} x_j^{(p)}$, where $i = 1, j = 2$, that will be simply $x_1^{(1)}$ that is $+$, $x_2^{(1)}$ that is also $+$. So, I get product to be positive, coming to the second one, one bit is positive the other bit is negative, so it will be -1 . Coming to the third one, both of them are positives so I will get $+1$. When I summed over all these three patterns for p varying from 1 to 3, my result will be equal to 1. So, this w_{12} this will be equal to $+1$ similarly, and w_{12} will be same as w_{21} that you can easily verify., similarly When I compute w_{13} , I had to consider the first bit and third bit, over here it is $+1$, over here it is also $+1$, over here it is also 1.

So, w_{13} I get that is nothing, but $+3$, what is w_{14} here it is $+1$, here it is -1 , here also, it is -1 , so effectively I get the w_{14} to be equal to -1 . Then w_{15} here it is $+1$, here it is $+1$, here it is -1 . So, it will be 1 similarly, w_{16} that will be $+1, -1, -1$, it will be -1 , is that okay, this computation. So, likewise you will find that, I have to compute w_{23} , I would not compute w_{21} because the w_{21} is same as w_{12} , $w_{22} = 0$. So, I have to compute w_{23} so second and third here it is $+1$, here it is -1 , here it is $+1$. So, $1 + 1 - 1$, w_{23} becomes $+1$. Similarly, w_{24} take the second and the fourth bit, here it is $+1$, second and fourth bit here also it is $+1$ because both the bits are negative.

Second and fourth bit, it is -1 , so that gives me w_{24} also equal to $+1$. Similarly, w_{25} second bit and fifth bit $+, +1$ here, second bit and fifth bit -1 here, second bit and fifth bit -1 here. So, I get overall -1 similarly, w_{26} here it is $+1$, here it is $+1$, here it is -1 . So, w_{26} is $+1$, right? Now when I compute w_{31} is same as w_{13} , I do not have to be recompute, w_{32} is same as w_{23} I do not have to be recompute, $w_{33} = 0$. So, I have to compute w_{34} , w_{34} here it is $+1, 3$ 4 here it is $-1, 3$ 4 here it is -1 . So, w_{34} becomes -1 , w_{35} it is $+1$, here also it is $+1$, so here it is -1 , so w_{35} becomes $+1$ w_{36} here it is $+1$, here it is -1 , here also it is -1 .

So, w_{36} becomes -1 similarly, I have to go for the w_{45} because others I need not compute. So, $w_{45} + 1 - 1$, here it is $+1$, so w_{45} becomes $+1$, $w_{46} + 1 - 1$ and $+1$. So, this becomes $+3$ then I have to compute w_{56} , because others I need not compute, $w_{56} + 1 - 1 + 1$ so it effectively becomes $+1$. So, these are all the weight vectors that I compute, which stores these three different patterns. So, once I have this weight vectors, then using this weight vectors.

Now, if I feeding an unknown feature vectors, after iteration then when this network will converge it will give me an output vector, which is close to the output vector, will be one of the three. And that will be closest to the input feature vector, that you have presented even if

the input feature vector is not one of this three. So, it will try to extract one of the stored patterns, which is closest to the input vector. Now, the question is why should this network work, how does it work really?

(Refer Slide Time: 46:24)

Y = ($y_1, y_2 \dots y_n$)
 Output of i^{th} node
 $\text{sgn} \left(\sum_{j=1}^n w_{ij} y_j \right)$
 $\sum_{j=1}^n w_{ij} y_j = \sum_{j=1}^n \underbrace{\sum_{p=1}^m x_i^{(p)} x_j^{(p)}}_{w_{ij}} \cdot y_j$

Suppose, I feed in an input feature vector, binary feature vectors, which is given as Y, which is nothing but y_1, y_2 up to say y_n . It should also have the same number of components and given this, the output of i^{th} node, it will be either +1 or -1. And this output will depend upon, you have to compute w_{ij} times y_j for, j varying from 1 to n . And depending upon the sign of this, if it is greater than 0 and output of i^{th} node will be +1, if it is less than 0 then output of the i^{th} node will be -1, that will be repeated many times, during the iteration. And finally, it will converge, so this sign of the output of the i^{th} node, will actually change many times not only once and that will change asynchronously.

I cannot say that, after change of the output of the first node, then only the output of the second node will change. There is no such guarantee because everything works asynchronously and all of them are working in parallel. So, you find that, if you analyze this

term, that is this summation $\sum_{j=1}^n w_{ij} y_j$, where j varies from 1 to n , if I analyze this, this will

simply be sum of what is this w_{ij} ? w_{ij} is $\sum_{p=1}^m x_i^{(p)} x_j^{(p)}$, where p varies from 1 to m . If I have m

number of patterns, where j varies from 1 to n , so this is what my w_{ij} , right?

(Refer Slide Time: 49:29)

$$= \sum_{p=1}^m x_i^{(p)} \left(\sum_{\substack{j=1 \\ j \neq i}}^n x_j^{(p)} y_j \right)$$

And this term, I can write as $\sum_{j=1}^n \sum_{p=1}^m x_i^{(p)} x_j^{(p)} y_j$ where j varies from 1 to n and $j \neq i$. And here, p

varies from 1 to m , so I can write this whole from in that manner. So, it is just rewriting this

expression into this form $\sum_{p=1}^m x_i^{(p)} \sum_{j=1}^n x_j^{(p)} y_j$, so here you find that these value, within this I will

not be written because I want the i^{th} output, so my variable will be j . It is the i^{th} output. So, I have to sum it over j .

So, if I compute for all values of i , I will get the vector, is that okay? So, here you find that depending upon the closeness of this vector x and y , this is the product of $x_j^{(p)} y_j$ and you are taking the summation for $j = 1$ to n , where $j \neq i$, right? So these value will be close to n , where n is the dimensionality, if the vector $x_j^{(p)}$ and y_j , if the values $x_j^{(p)}$ and y_j are matching most of the time. And this will be close to $-m$, if $x_j^{(p)}$ and y_j they do not match most of the time.

So, if they match most of the time, I get a positive output and that means, that this i^{th} bit whatever is the output, that should be signed by the i^{th} bit. And if they do not match most of the time, then it should be the reverse. So, if you go on iterating on this, a large number of times when the network converges, the output of the network will be the one, which is closest to the pattern, which is stored into the network. So, that is the logic behind, why this network

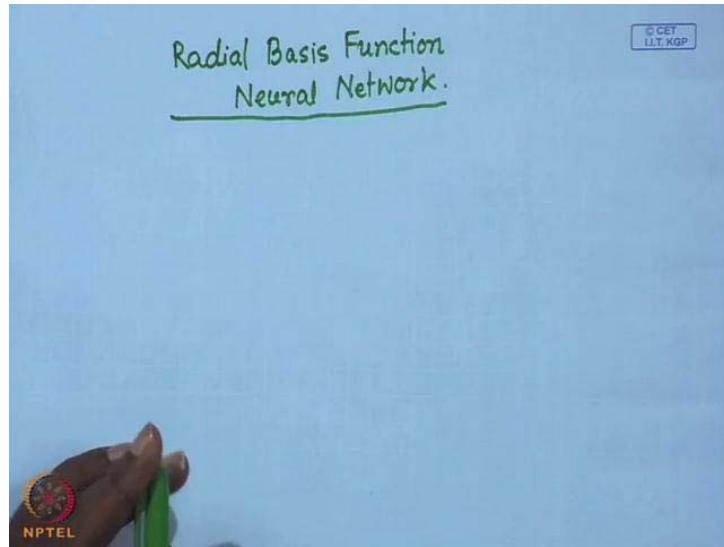
should work as an associative network. So, I will stop this lecture today, next day we will continue with some other topic.

Thank you.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 27
RBF Neural Network

(Refer Slide Time: 00:22)

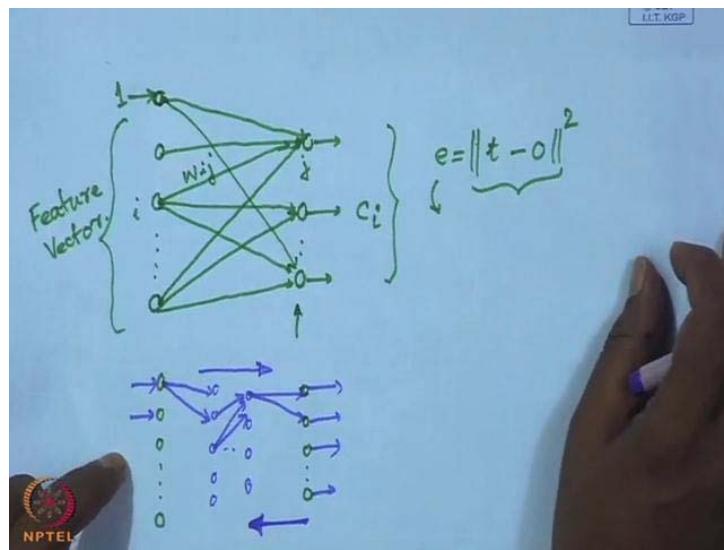


Hello. So, in the last few classes we have talked about the neural networks and the application of neural networks for pattern recognition. Particularly, we have talked about two types of neural networks; one was the Hopfield neural network, as we have said that Hopfield neural networks are basically meant for binary feature vectors. So, if I have a number of binary feature vectors the, Hopfield neural network basically learns or remembers those feature vectors by updation of weights. So, learning process is basically updation of weights and in case of back propagation neural network, the learning is also by means of updation of weights.

But for back propagation or feed forward neural network, the neural network is not confined to the binary feature vectors, only it considers the feature vectors which contains real components also. Now, when we talked about the feed forward neural network or back propagation neural network, we have said that it is a multi-layered neural network, which has got one layer, which is input layer and it is just a buffer layer, whose responsibility is to feed, is to just pass the information to the layers above it. And in the

simplest case of the back propagation neural network, we had single layer perceptron, which consists of only one output layer, in addition to the input layer. So, the neurons in the output layer so if I just draw it in case of a single layer perceptron, the network architecture of single layer perceptron was something like this.

(Refer Slide Time: 02:25)



We had an input layer and we had an output layer so the number of nodes in the input layer is same as, the dimensionality of the feature vector or if the feature vector is of the dimension d then the number of nodes in the input layer is $d+1$. So, one node is to take care of the bias term and the number of nodes in the output layer is same as, the number of classes. So, the network architecture was something like this, from each input layer node we had a connection to every output layer node so it was something like this and so on. So, a feature vector is feed to the input layer nodes and one of the node was kept at a constant value, which is equal to 1.

That is meant for feeding the bias term and outputs of this different nodes in the output layer, they indicated the corresponding classes C_i . Every node in the input layer is connected to every other node in the output layer, through a connection weight say w_{ij} . So, i th node in the input layer is connected to the j th node in the output layer, through a connection with w_{ij} . So, during training what we had done is, we had fed the feature vectors for which the classes are known. And if the class of a feature vectors known, that means I know that what should be the output corresponding to that particular feature vector.

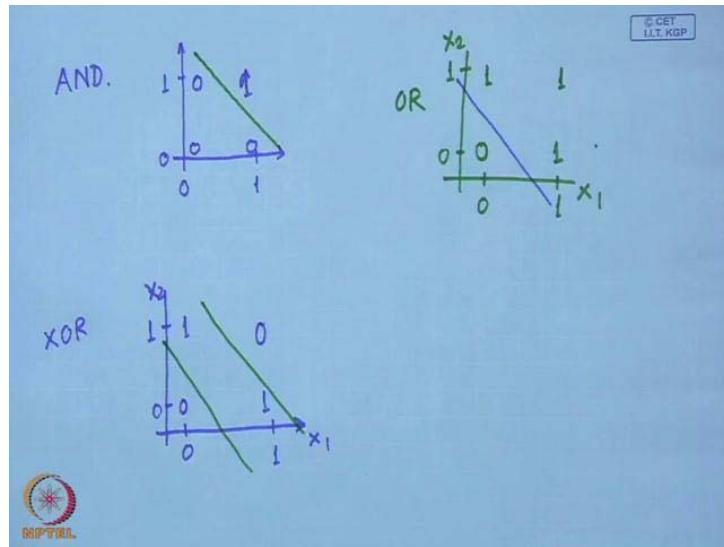
So, that output is an output vector say t and when the network is not properly learnt, in that case for an input vector, I get an output which may not be equal to t . So, I get an output which is o whereas t is the target output. So, I have an error e which is equal to $e = \|t - o\|^2$ square of this, right? So, during training of this back propagation neural network what we had done is, we tried to minimize this error e by means of updating this connection weights. So, this was expressed as a function of a set of connection weights w_{ij} and then through gradient descent procedure, we have reduced the error by updation of the connection weights w_{ij} .

In case of multilayer perceptron, we had the same input layer, input layer is always possible, and we also had the number of nodes in the output layer, which is same as the number of classes. In addition, we had one or more hidden layers. However, the connection pattern remains the same, and every node in the input layer is connected to every other node in the layer above it. So, it continued this way and so on and finally, we get the output. However, the training for this back propagation neural network was same as, a single layer perceptron that is we, for unknown feature vector which is input to this neural network.

We know what the output is or we know what is the target output. And when the neural network is not properly learnt we get an output for the same vector, same input feature vector, which is not same as the target output. So, there again we get an error and this error when I express in the form of a function, of the weight matrix, or the connection weights then will try to reduce the error by optimizing or by adopting the weights. And when we modified the weights, when we adapt the weights the error information for reduction of the error, that propagates from output layer to input layer so it to propagate in the reverse direction.

So, this is like this, the error information or the weight updation information that propagates from the output layer to the input layer. So, that is why it is called back propagation learning, but during the classification, the information moves from the input layer to the output layer. So, the information is feed from input to output so that is why it is feed forward network whereas, the learning algorithm is called as a back propagation learning algorithm. Now when we discussed about this, feed forward network or back propagation network, we have said that if the patterns or if the classes are linearly separable then a single layer perceptron is sufficient. But if the classes are not linearly separable then single layer perceptron is not sufficient, we have to have one or more hidden layers. And what these hidden layers do is, it represents a non-linear boundary by a set of piecewise linear boundaries and to explain this, we have taken few examples.

(Refer Slide Time: 08:53)



We have taken an AND function. So, in case of AND function if I represent the AND function as, a two-dimensional binary vector. So, I have the input components (0 0), (0 1) (1 0) and (1 1) so if the function is an AND function, only when the input is (1 1), the output is 1. In all other cases (1 0), (0 0) or (0 1) output is 0 and now, if this AND function we consider to be a classification problem then you find that it is quite obvious. I can draw a straight line separating the set of 0's and the set of 1's, so that means the input vector (1 1) that is put in one class. And all other combinations of the input components, that is (1 0), (0 0) or (0 1) that is, those are put in another class.

And I can separate these two classes by a straight line so that means that this particular function, AND function is a linearly separable function. Similarly, if I take an OR function here again I put it as $(x_1 \ x_2)$ somewhere here, let us put (0 1), (0 1) so only in case of (0 0) both x_1 and x_2 components, when they are 0 the output is 0. In all other cases the output is 1 so I have this type of situation and again here, we find that I can separate the 1's and the 0 by a straight line. So, which clearly indicates that, this is also a linearly separable problem, but we have faced difficulty in case of XOR function.

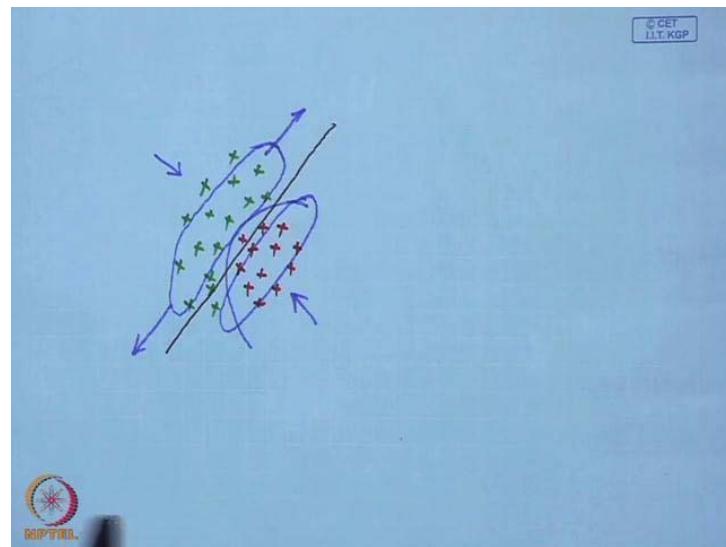
So, in case of XOR function if I put it in the same fashion, I have $(x_1 \ x_2)$ as 0, 1, 0, 1 so when the input is (0 0) the output is 0, when the input is (1 1), output is also 0, when the input is (0 1) or the input is (1 0) the output is 1. Now given this type of situation, you find that I can no longer separate the outputs 0's and 1's by single straight line. Rather, I need to have two different straight line over here, to separate the 0's and 1's. So, this clearly says that the XOR

function is not linearly separable. And when I discussed about this back propagation neural network, we have said, we have shown the AND function and OR function they can be represented, they can be implemented using single layer perceptron's.

But when you go for XOR function, the XOR function cannot be implemented by single layer perceptron rather, we need at least one hidden layer for implementation of XOR function by a feed forward neural network because it is not linearly separable. So, now, today will discuss about another kind of network which is called radial basis function network. So, what radial basis function network does is, it performs a non-linear transformation over the input vectors, before the input vectors are fed for classification. So, by using such non-linear transformation, it is possible to convert a linearly non separable problem to a linearly separable problem.

It is also possible what this, RBF network or radial basis function network does is, it increases the dimensionality of the feature vectors. So, I will come to that later first let us see that, how this introduction of non-linear function to the input feature vectors, before we go for classification, can make a linearly non separable problem, can convert a linearly non separable problem to a linearly separable problem. So, let us just take a situation something like this.

(Refer Slide Time: 14:09)



Say, I have a set of feature vectors, which are of this form. So, here you will find that, if I say that all these green crosses represent feature vectors belonging to one class and all the red crosses represent feature vectors belonging to one class, another class. So, clearly you will

find that, these two feature vectors cannot be separated by straight line rather, what I need to have is, I have to have a curve, a quadratic curve at least to separate these two classes. Now if I put, if I impose on non-linear transformation something like this, that a transformation which will expand the feature vectors in this direction and contract the feature vectors in this direction.

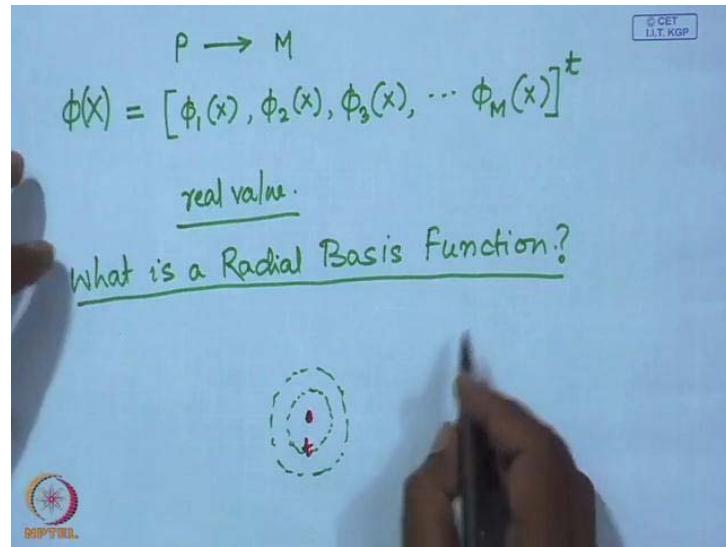
So, if I do this these feature vectors will be converted, will be clustered into a cluster something like this and these feature vectors, it is possible they will be clustered into a cluster like this. And because of this non-linear transformation, now it may be possible that, I will be able to find out a straight line, which separates these feature vectors from these feature vector. So, that is why I say that, if I use a non-linear transformation, if I impose a non-linear transformation to the feature vectors. Before I put them, I go for classification purposes then it may be possible to that a non-linearly separable problem can be converted to a linearly separable problem.

The other one as I said that the RBF network, the radial basis function network also increases the dimensionality of the feature vector. Because it has been found that, if we increase the dimensionality of the feature vectors then classification problem which is linearly non separable in a lower dimensional space. It may be possible that, they will be linearly separable in higher dimensional space. And more and more you increase the dimension, the linear separability, the possibility of linear separability increases. So, it is something like this, if I put it in this form where the tips of my fingers, they represent different feature vectors.

And the tips of the fingers of the left hand, they represent feature vectors belonging to one class and the tips of the fingers of the right hand, they represent feature vectors of another class. And these feature vectors they are coplanar so all of them are lying in the same plane. So, if the situation is like this obviously, this set of feature vectors are not linearly separable.

However, if I introduce a third dimension something like this, I push in the forward and the other one back ward. And in this third dimension, now we find that they are linearly separable so this is just a simple logic that, if I increase the dimensionality. Then it is possible to increase the possibility of linear separability, among the classes which are linearly non separable in lower dimensional space. So, the RBF network actually imposes both, it imposes a non-linear function, a non-linear transformation to the feature vectors. And it also increases the dimensionality of the feature vectors.

(Refer Slide Time: 18:48)



So, if I have a set of feature vectors whose original dimension is the dimension P. I increase the dimension of the feature vectors to dimension M. So, how it is to be done, a given feature vector X is subjected to function ϕ . So, $\phi(X)$, where this $\phi(X)$ is the form of $\phi(X) = [\phi_1(x), \phi_2(x), \phi_3(x), \dots \phi_M(x)]^t$. If the original feature vector X is of dimension P, for each such feature vector X, I impose this set of ϕ functions, n number ϕ functions. And each of these ϕ functions produce a real value so as I have M number of ϕ functions so by application of this phi function onto this feature vector X.

I create a number M of real components so what I get is an M dimensional feature vector and if $M > P$ then obviously, I am increasing the dimensionality of the feature vector from P to M. And in case of radial basis function, each of these ϕ function, whether it is $\phi_1(x)$, $\phi_2(x)$, $\phi_3(x)$ up to $\phi_M(x)$ each of this ϕ functions are radial basis functions. Then what is a radial basis function? So, let us try to see what a radial basis function. Every radial basis function has a receptor, the radial basis function has a receptor say t and as I move radially away from this receptor, the value of the function goes on increasing or goes on decreasing.

And the value at a point, depends upon the radial distance of the point from the receptor t. Value of the function is either maximum or minimum at location t so if I put concentric circles around this receptor t, the value of the function ϕ on each of this concentric circles are constant. So, there are different choices of radial basis functions.

(Refer Slide Time: 22:38)

Multiquadratics:
 $\phi(r) = (r^2 + c^2)^{1/2} \quad c > 0$

Inverse Multiquadratics:
 $\phi(r) = \frac{1}{(r^2 + c^2)^{1/2}} \quad c > 0$

Gaussian Function:
 $\phi(r) = \exp\left[-\frac{r^2}{2\sigma^2}\right] \quad \sigma > 0$

DRPTEL

I can put those radial basis functions as, one of them is called multi quadrics, which is given by $\phi(r) = (r^2 + c^2)^{1/2}$ for $c > 0$. Then I can have inverse multi quadrics, where the function

$\phi(r)$ is given as, just inverse of quadrics that is $\frac{1}{(r^2 + c^2)^{1/2}}$ for $c > 0$. I can also have use

Gaussian functions, where $\phi(r) = \exp\left[-\frac{r^2}{2\sigma^2}\right]$ for $\sigma > 0$. And this r is a measure of the distance

of the point from, what I said as the receptor of the radial basis function. So, this r if I want to compute this radial basis function at a point say X , coming over here.

(Refer Slide Time: 24:24)

$P \rightarrow M$

$\phi(x) = [\phi_1(x), \phi_2(x), \phi_3(x), \dots, \phi_M(x)]^t$

real value.

What is a Radial Basis Function?

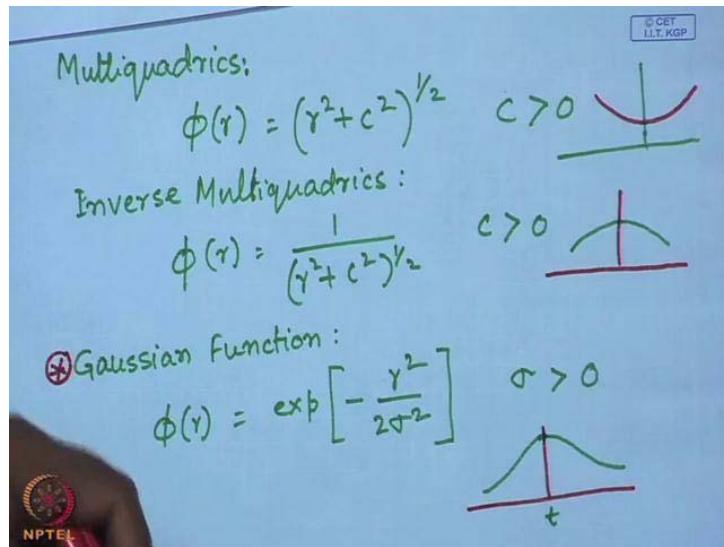
$r = \|t - x\|$

$\|x - t\|$

DRPTEL

If I want to compute, the radial basis function at a point X, where t is the receptor of this radial basis function. Then r is nothing but $\|t - X\|$ or $\|X - t\|$ which is nothing but the distance between x and t. So, you will find that in all these different cases if I use this multi quadrics then at $r = 0$ the value of the radial basis function is equal to c, which is the minimum. And as far r goes on increasing, the value of the radial basis function goes on increasing.

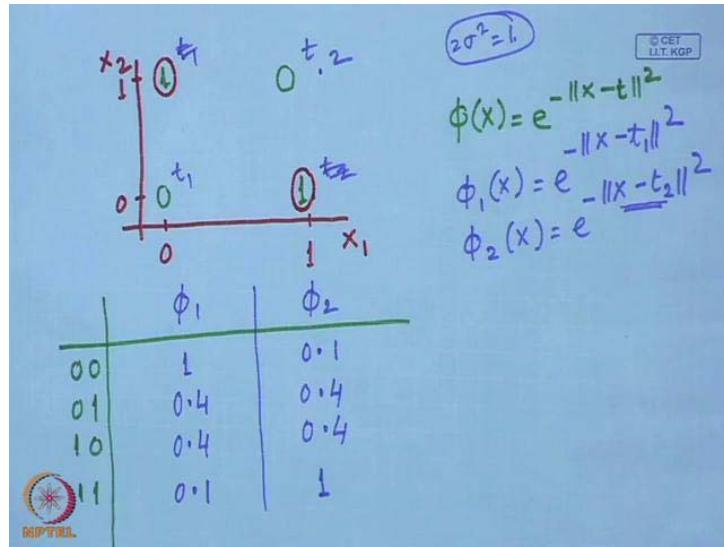
(Refer Slide Time: 25:16)



So, the nature of the radial basis function for multi quadrics will be like this, when I go for the inverse multi quadrics, at $r = 0$, that means when the point X and the point t coincide. That is at the receptor point, the value of the inverse multi quadrics is minimum which is nothing but $\frac{1}{c}$, this is the maximum which is nothing but $\frac{1}{c}$. And as r goes on increasing, the value of radial basis function goes on decreasing so the kind of, the nature of the function with inverse multi quadrics will be like this. Example of inverse multi quadrics is case of a Gaussian function you all know that, this has a very, very common form, the Gaussian function, will be something like this.

So, at the receptor location t, the value of the function phi will be maximum, and as you move away from the receptor point, the value of the function $\phi(r)$ that goes on reducing following this pattern. And it is this Gaussian function, which is most commonly used in case of radial basis function. Now, let us say that how by using this radial basis function neural network, we can convert a linearly non separable problem to a linearly separable problem, even keeping the dimensionality of the feature vector same. And so for doing that I will use the same XOR problem.

(Refer Slide Time: 27:23)



So, I have this x_1, x_2 here I have 0, here I have 1, again 0, here it is 1, I have 0 over here. So, these are the outputs of an XOR function, what I will do is, I will use a radial basis function to nonlinearly transform these feature vectors into another space. And the radial basis function that I use is, a Gaussian basis function so the form of the radial basis function will be $\phi(X)$, which will be given by, $e^{-\|X-t\|^2}$ and this t is the receptor and as I said that, I will not increase the dimensionality. So, I will use two of the receptors, one of the receptor I consider is this point at location (0 1) and other receptor I consider this point at location (1 0).

So, by using this if I compute the non-linear transformation then the values that I will have will be something like this. I use this non-linear transformation so I will put this inputs my inputs are (0 0), (0 1), (1 0) and (1 1) and I will have the output functions. One is ϕ_1 and the other one is ϕ_2 so if I say that this is my t_1 , the receptor t_1 and this is my receptor t_2 . So, $\phi_1(X) = e^{-\|X-t_1\|^2}$ and $\phi_2(X) = e^{-\|X-t_2\|^2}$.

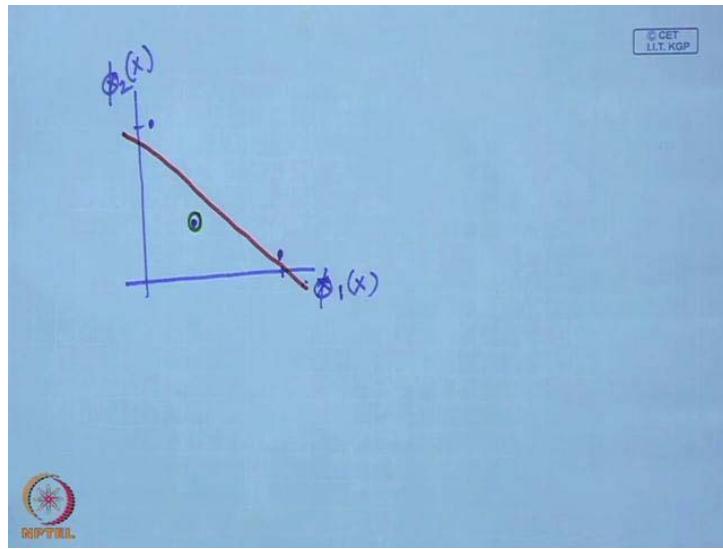
So, here you will find that, I assume that $2\sigma^2$ that is equal to 1. However, that does not matter for this non-linear transformation. So, I define that when it is (0 0) then what will be my $\phi_1(0,0) = e^{-\|X-t_1\|^2}$, $\phi_1(0,0) = e^{-\|X-t_1\|^2}$ will be simply equal to, sorry let me use the receptor like this. Let this be t_1 and let this be t_2 so $\phi_1(0,0) = 1$ and if you compute this term $\phi_2(0,0) = 0.1$. $\phi_1(0,1) = 0.4$ and $\phi_2(0,1) = 0.4$, $\phi_1(1,0) = 0.4$, $\phi_2(1,0) = 0.4$. And for $\phi_1(1,1) = 0.1$ and $\phi_2(1,1) = 1$. So, this is the kind of situation that I have so what have used is

this (0 0) has been used as one of the receptors of the radial basis function and (1 1) has been used as another receptor of the second radial basis function.

So, obviously (0 0) as it coincides with my receptor t_1 , so I have the situation that $\|X - t_1\|^2$, that will be equal to 0 so this is e^0 , which is equal to 1. So, for $\phi_1(0,1)$, $\|X - t_1\|^2$ which is equal to 1 similarly, if I compute $\phi_2(0,0)$ that will be equal to 0.1. For $\phi_1(0,1)$ will be 0.4, $\phi_2(0,1)$ will also be 0.4, for (1 0) which is this point, $\phi_1(X)$ will be equal to 0.4, $\phi_2(X)$ will also be equal to 0.4.

Because of these exponential functions, for (1 1) if $X = (1 1)$ this coincides with t_2 so this distance be equal to 0 now, $\phi_2(X) 1$ and $\phi_1(X)$ will be equal to 0.1. So, you find that (0 0) has been mapped to (1 0.1), (0 1) have been mapped (0.4 0.4), (1 0) has been mapped to (0.4 0.4), (1 1) has been mapped to (0.1 1).

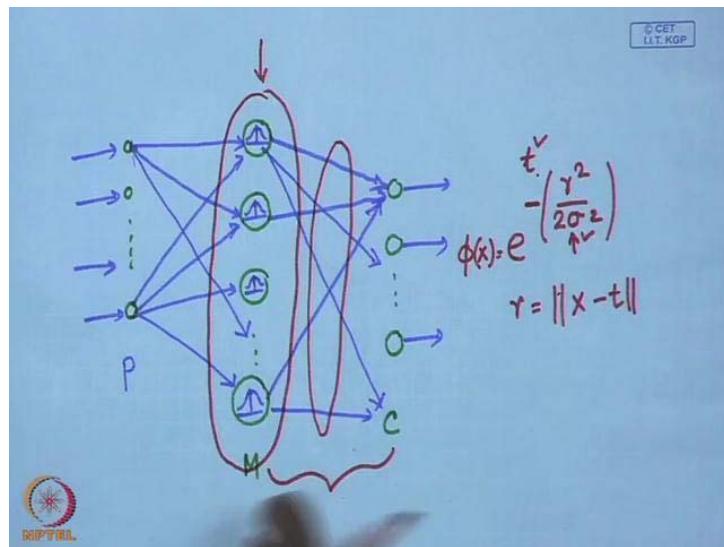
(Refer Slide Time: 33:48)



So, if I plot this, the kind of plot that I have is (0, 0) is now mapped to (1, 0.1) which will be somewhere over here, this is my x_1 , this is my x_2 . Now, this will be phi 1 instead of x_1 it will be phi 1 and this will be phi 2. So, this is $\phi_1(X)$ and this is $\phi_2(X)$. So (0, 0) has been mapped to (1, 0.1), (0, 1) has been mapped to (0.4, 0.4) which will be somewhat here. (1, 0) that is also mapped to (0.4, 0.4) it is the same point so both the points up to be, now has been mapped to the same point in $\phi_1(X)$, $\phi_2(X)$. And that has happened because of the non-linear transformation whereas, (1, 1) has been mapped to (0.1, 1) somewhere over here so this is 1.

This being the situation now you find that, I can draw a straight line separating these points say, problem which was nonlinearly, which was not linearly separable in original space, it has now been linearly separable in ϕ space, simply by using: non-linear transformation. And the possibility of the linearly separability, between the classes increases as we go on increasing the number of dimension of the feature vectors, by using these ϕ functions. So, by using this now the architecture of the network will be of this form.

(Refer Slide Time: 35:49)



We have the input layer as before, it always has been present because this is the layer which accepts the input vectors and feeds to the next higher layers. We will have only one hidden layer and the number of nodes in the hidden layer will be equal to M , where M is the dimension, to which we want to increase the dimensionality of the feature vectors. So, if I have P number, P is the dimension of the input feature vectors, we have P number of nodes at the input layer and if M is the increased dimensionality then M is the number of nodes in the hidden layer. Each node in the hidden layer, actually performs a radial basis function and as we said, it is something like this, each node represents or computes a radial basis function.

And in the output layer, we will have number of nodes which is same as the number of classes C , now from each node in the input layer I have a connection to each node in the hidden layer. And from each node in the hidden layer, I have a connection to each node in the output layer. Finally, I have outputs from the classifying neurons and here I feed the input feature vectors so this is, what is the architecture? So, the nodes in the hidden layer, they actually perform the radial basis function and nodes in the output layer, they perform linear

combination of the outputs of the hidden layer nodes. So, here I will have a linear classifier so actually classification is done only at the output layer.

Whereas, at the hidden layer only we perform the radial basis function so that is a transformation of the input feature vector to a hidden space. Because this is my hidden layer so you can say that it is a transformation to a hidden space, so when I have a number of nodes in the hidden layer which performs radial basis function. So, I have two levels of training, the first level of training is, training of the hidden layer nodes. So, what does this training mean, as every node in the hidden layer is a radial basis function so for each of the radial basis function I have to have a receptor t . So, one of the purpose of training of the hidden layer nodes is, to find out that what should be this receptor, t .

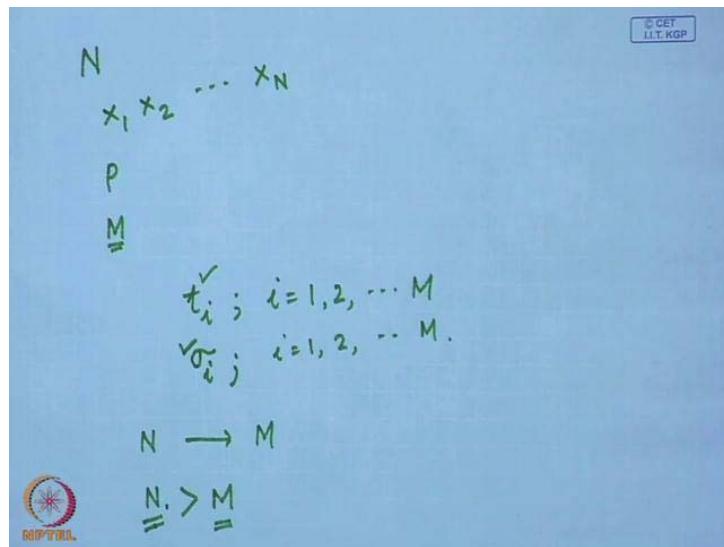
And then if I use the Gaussian function then we have said that, the radial basis function is of the form $e^{-\frac{r^2}{2\sigma^2}}$, so that is what is my $\phi(X)$, where this r is nothing but the distance between x and t . So, this σ actually indicates that what is the spread of this radial basis function, so that is also what is to be trained. So, by saying that I want to train the hidden layer nodes, for each of the nodes I have to find out what is the t and for each of the nodes I have to find out what is the σ ?

So, that is what is meant by training of the hidden layer nodes and at the second level I have another training operation, by which I have to train these weight vectors, which connects the outputs of the hidden layer nodes to, the output layer nodes because it is the linear combination of the outputs layer nodes of the hidden the nodes, which decides that to which class a sample will belong. And that will be outputted, that with the available at the output of the, output layer nodes. So, let us see that, how you can train a node in the hidden layer so by saying that the hidden layer nodes are to be trained, what, let us see the situation that we have. If I go for supervised classification or supervised learning what I have is, I have set of feature vectors or which are labelled with its class belongingness, which are to be used for training the neural network.

In case of multi layered perceptron, in case of single layered perceptron we did not have any non-linear transformation so from the label of the input feature vector and actual label that you got, at the output of the output layer nodes, we have defined an error function. And our aim was that by updation of the connection weights, we have to reduce that error function or we have to minimize that error function. So, the algorithm was that in order to minimize the error function, how the connection weights are to be modified.

And that modification of the connection weights was carried out from the output layer to the input layer, in the backward ways. whereas in the case of RBF network, the training consists of two parts, the first one is training of the hidden layer nodes. And when I say it is the training of the hidden the nodes that means, it is for every hidden layer which represents a radial basis function, I have to find out that what should be the receptor. And also I had to find out what is the spread of the radial function, of the radial basis function which is σ . So, if we are given a number of training vectors.

(Refer Slide Time: 43:45)



So, we are given N number of training vectors, vectors from say x_1, x_2 up to say x_N , we have N number of training vectors. Where each of these vectors is of say dimension P and I want to transform these vectors to a vector of dimension capital M. So, naturally at the hidden layer of the node, at the hidden layer I have to have capital M number of nodes. And when I have this M number of nodes so for each of the nodes I have to have a receptor. That means for the i^{th} node, I have to have receptor t_i so i varies from 1, 2 up to M and the i^{th} hidden layer node, I have to have σ_i again, i varying from 1, 2 up to capital M. So, I have to have M number of receptors, I have to have M number of variances or standard deviations.

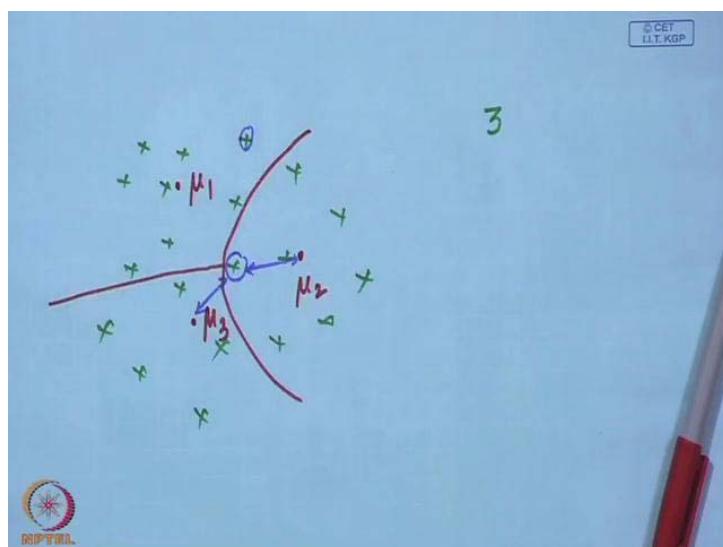
So, this can be done in various ways, the simplest way is at a random from the training samples, you pick M number of samples, which are to be used as receptors, which is not very logical rather, what appears to be more logical is, you go for clustering of this capital N number of training samples into M number of clusters. So, when I go for clustering, I basically get a space where it is more likely to have a feature vector. Clustering we have not discussed yet, but I will just for the purpose of completion of this lecture, I will simply

indicate what does this clustering mean. So, I have capital N number of samples, from that I have to form capital M number of clusters.

So, it is understood over here, that this N has to be greater than capital M, if $N < M$ that is number of samples training, samples is less than the number of clusters that are to be formed, it is not possible. Because, even if in a single cluster I assume that, there has been only one feature vector, at the most I can have N number of feature vectors or at the most I can have N numbers of clusters.

So, I cannot form N number of clusters which is more than the number of feature vectors, which is provided. So, obviously our assumption is that the number of feature vectors N is greater than, the number of clusters that one has to form that is M. Now, there are various clustering algorithms, one of the clustering algorithm is say initially, I partitioned this N number of samples into capital M number of clusters. So, at every cluster will have a number of samples, for each of the created clusters say.

(Refer Slide Time: 47:39)



Suppose I have got a number of feature vectors, which are given like this. So, these are the set of feature vectors now, suppose I want to form three clusters so initially what I do is at random arbitrarily you divide this set of samples. You partition the set of samples into three different clusters so this cluster contains these samples, this cluster contains these samples, and this cluster contains these samples and so on. Now, once I have these initial clusters then what I can do is for each of the cluster so formed, I compute the cluster center. So, which will be that representative of that cluster, of the samples belonging to that cluster.

So, if over here suppose the center of this cluster comes out to be somewhere over here, the center of this cluster comes out to be somewhere over here and the center of this cluster comes out to be somewhere over here. So This will be the center cluster, center μ_1 , this will be the cluster center μ_2 , and this will be cluster center μ_3 and so on. Now, once I form this cluster centers in the next iteration what I do is for each of the samples, I find out which cluster is nearest to that particular sample. So, here you find that it may be possible that this class, this particular sample say this particular sample it's distance from μ_3 is less than its distance from μ_2 .

So, initially though this sample was put in cluster 2, in the second iteration because its distance from the cluster center 3 is found to be less than, its distance from the cluster center 2, this sample will be more from cluster 2 to cluster 3. And this operation you will perform for every sample so for each such sample you find out that what its nearest cluster center is and you move this sample to, the cluster whose center is nearest to it. And you perform this for each of the samples so at the end of second iteration you will find that, initial clusters that we had formed, the samples in the initial cluster have changed. I still have three numbers of clusters, but the sample values have changed.

So, in the second iteration at the end of second iteration, again I compute that with this new set of samples what is its cluster center. After computing, re-computation of this cluster centers, again I go for the reassignment of the feature vectors, based on its distance from this new cluster centers. And this operation you will perform in a number of iterations.

And finally, when it stabilizes then we say that the clustering is complete and at that point whatever cluster centers I have, those cluster centers, have those cluster centers are my receptors. So, by this what I have done is, for each of the radial basis functions, or each of the nodes in the hidden layer I have computed the receptors, these receptors are nothing but the cluster centers.

(Refer Slide Time: 51:55)

$$\sigma_j = \sqrt{\frac{1}{P} \sum_{i=1}^P (t_j - t_i)^2}$$

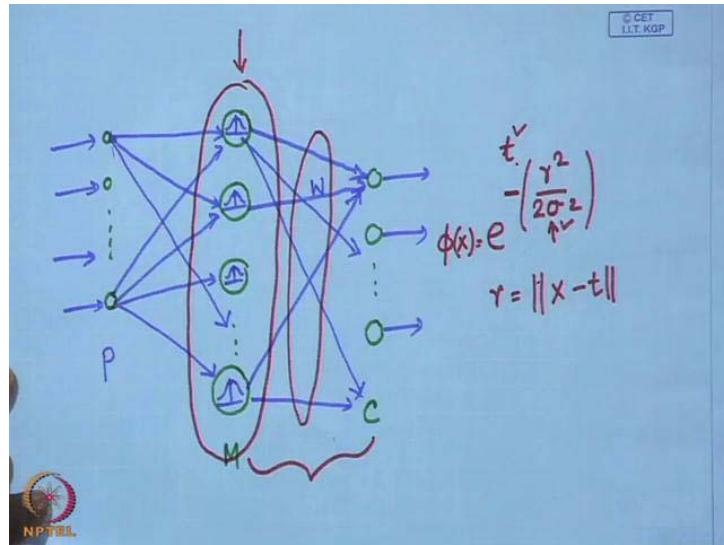
Once I do that, next what I have to do is, I have to, so for the every i^{th} node I have to compute a t_i , I found out what is t_i . Next I have to find out what is σ_i so then what I can do is, I can go for a nearest neighbor rule. From every t_i , I can find out say P number of nearest receptors and the distance of those P numbers of nearest receptors. So, I get P number of distances and root mean square of all these distances, I can take as, that as the variance or the standard deviation σ_i . So, if I am interested to find out what is σ_j , what I will

do is, I will take $\sigma_j = \sqrt{\frac{1}{P} \sum_{i=1}^P (t_j - t_i)^2}$, sum for $i = 1$ to P , as I have P number of nearest t_i 's.

This is one of the way in which I can compute σ so I have the receptors t_i and I have σ_i . So, once I have this t_i and σ_i then you will find that in the middle layer every hidden layer node is defined, for every node I have t_i , for every node I also have σ_i . So, here also have t_i , I have say t_j , here also have σ_j so this is defined for every node.

So Now, if I feed any feature vector X to this, I have the corresponding $\phi_1(X)$, I have the corresponding $\phi_2(X)$, I have the corresponding $\phi_M(X)$. So, the original feature vector X is now converted to M dimensional feature vectors now, using this feature vectors I have to go for classification. So, for that I have to train the output layer of this radial basis function neural network because as you see, if you remember the architecture of the radial basis neural network over here.

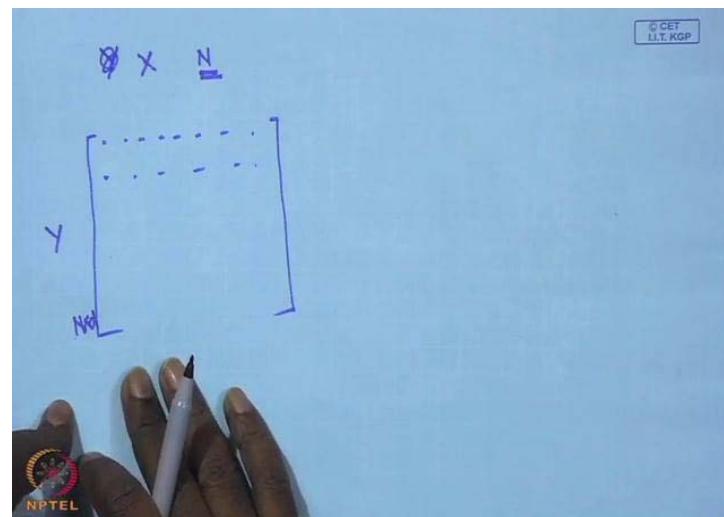
(Refer Slide Time: 54:37)



I had a set of weights W, so I have to compute this Ws for classification purpose. Now, here as we said that, as I increase the dimensionality and I, as I imposed a non-linear transformation and increase of dimensionality taken together. That increases the possibility, that the classes will be linearly separable in the higher dimensional space. And as I increase the dimension, as I increase the value of M, the possibility of linear separability goes on increasing.

But I do not know that, how many such radial basis functions I have to use so that, my original problem becomes linearly separable. So, though I have increased the possibility of linear separability, I have not been able to guarantee that, they are linearly separable. So, the best way to find out this connection weights at the output layer is, by means of least mean square error technique. So, what we have is, we have already discussed about the LNS technique.

(Refer Slide Time: 56:14)



And then, we have seen that if Y is a set of feature vectors and if I have a set of feature N number of feature vectors Y then using this Y , I can form a matrix. So, let me say this feature vector is X so if I have N number of feature vectors, every feature vector when I put in the form of a row, I have a matrix of say $N \times d$, if d is the dimensionality of the feature vectors.

And we call this matrix as Y and the logic that I put is, when I want to train say, first output node of my classifier, only the samples if I feed a sample which belongs to class one, only that output should be 1, the rest of output should be 0. So, for every input feature vector I can define an output vector so using each of these outputs, I can define what is, what in case of this LMS algorithm, we have said as a bias vector. And then by using the pseudo-inverse technique, I can find out what is the output to it. So, this discussion I will stop here today, I will continue with this discussion in our next class.

Thank you.

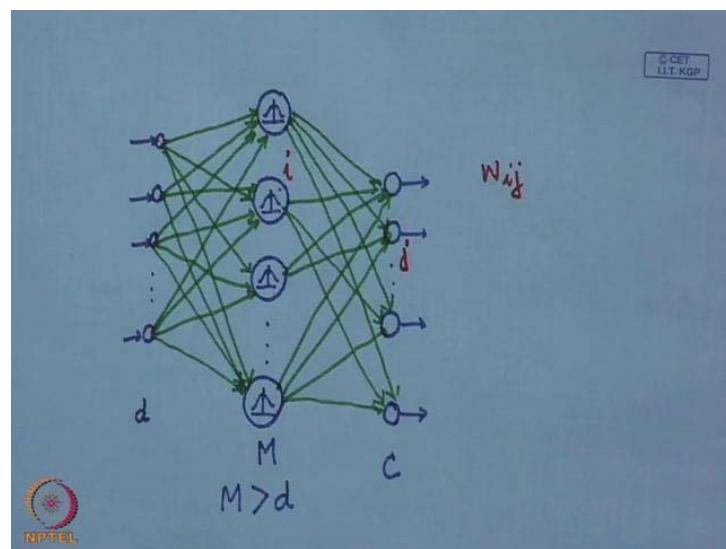
Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture No. – 28

Hello. So, in the last class we have started discussion on radial basis function neural network. We have seen that a radial basis function neural network consists of three layers, 1 is of course, the input layer and one of the three layers is the output layer and in between the input layer and output layer we have a hidden layer. So, unlike in case of multilayer perception, where we can have one or more hidden layers. In case of the radial basis function network we have only one hidden layer, and every neuron in the hidden layer computes a radial basis function.

So, when I have the neurons in the hidden layer. For every neuron, which computes a radial basis function, radial basis functional value for an input feature vector. Every radial basis function has got two parameters, one is called the receptor and other one, which defines the spread of the radial basis function. So, the architecture that we have is something like this.

(Refer Slide Time: 01:49)



We have one input layer, so the input layer contains a number of neurons and the number of neurons in the input layer is same as the dimensional, dimensionality of the feature vector. So, that if the feature vectors of are of dimension d , I will have d number of nodes in the input layer so there will be d number of nodes. When the dimensionality of the feature vector is d , in the hidden layer I will have a number of nodes.

And suppose the number of nodes in the hidden layer is say capital M . So, as we discussed in our previous class that the purpose of the hidden layer nodes is to project the d dimensional feature vector into a higher dimensional feature vector. So, as I have n number of nodes in the middle layer, so obviously this M , the number of nodes in the hidden layer is greater than the dimensionality of the feature vector which is d . And as we say it that every node in the hidden layer computes a basic radial basis function.

Like this and at the output layer, which are basically the classifying neurons, I have the number of neurons of the number of nodes, which is the same as the number of classes that we have. So, if I have c number of classes then at the output layer I will have c number of neurons.

So, there we have c number of neurons. Where c is the number of class in which the pattern has to be classified. Then every node in the input layer is connected, is feeding input to every node in the hidden layer and output of the hidden layer every node output from the hidden layer is connected to every node in the output layer. So, I have the connections, which is something like this.

So, these are the connections from the input layer nodes to the hidden layer nodes. And because the purpose of this connection is simply to forward the input feature vector to the nodes in the hidden layer, we can assume that weight of each of this connection is equal to 1 and that is a difference with the connections from the hidden layer to the output layer nodes. Because, in every output layer node, computes a linear combination of the outputs of the hidden layer node. So, the connection from the hidden layer nodes to the output layer nodes is something like this.

Where we can say that every field i^{th} node in the hidden layer is connected to the j^{th} node in the output layer to a connection weight, which is equal to w_{ij} . So, because of this, every node in the output layer computes a linear combination of the outputs of the hidden layer. Based on this, the value of this linear combination the output layer nodes decides to which class the input vector should be classified.

Now, what can be done is, these output nodes can also impose a non-linear function to ensure that if a particular input feature vector belongs to class ω_j . In that case only the output of the j^{th} node will be equal to 1 and output of all other output layer node will be equal to 0. Similarly, if a feature vector, input feature vector belongs to say class 1 then only the output of the first node in the output layer will equal to 1 and outputs of all other nodes will be equal to 0.

So, as we discussed in the previous class that such a radial basis function network and RBF network incorporates two types of learning. 1 is, we have to learn that for every node in the hidden layer, because every node in the hidden layer represents a radial basis function, what should be the receptor of that radial basis function and what should be the spread of that radial basis function.

(Refer Slide Time: 08:28)

$$\phi_i(x) = e^{-\frac{\|x - t_i\|^2}{2\sigma_i^2}}$$

So, if the radial basis function is a Gaussian function, that is if it is something like this, say

$\phi_i(X) = e^{-\frac{\|X - t_i\|^2}{2\sigma_i^2}}$, where t is the receptor and σ^2 which is the variance. It decides that what is the spread of radial basis function. So, for every i^{th} radial basis function $\phi_i(X)$ t_i is the receptor and σ_i is the spread, so I have to know that what is the receptor for every radial basis function and what is the spread of every radial basis function. So, this is one level of learning and the second level of learning is, once through these radial basis functions a d dimensional feature vector is projected onto a M dimensional feature vector.

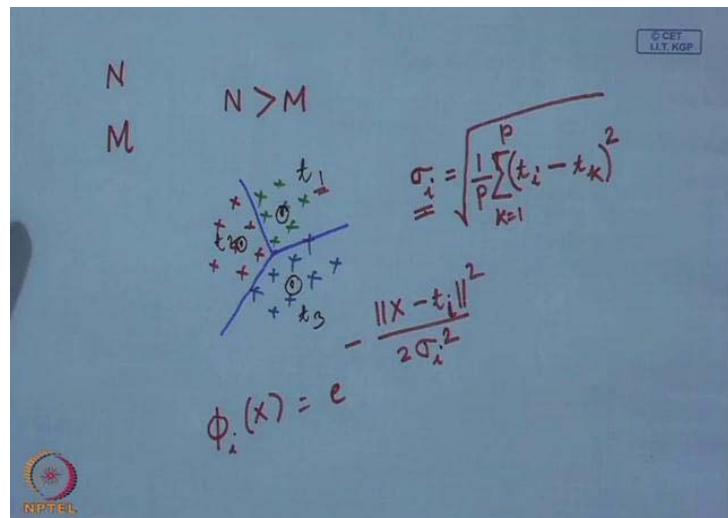
So basically, what we are doing is, we are increasing the dimensionality of the feature vector. As we have indicated in our last class that the purpose of increasing dimensionality is that if the feature vectors are linearly non separable in the d dimensional space then we cast them into a higher dimensional space. Then the possibility that they will be linearly separable in a higher dimensional space increases and this possibility increases with the value of M. So, as we increase the dimensionality more and more, the possibility of linear separability of the feature vectors also increases.

So, the feature vectors in the d dimensional space, which are not linearly separable, when I asked them into an M dimensional space at M is greater than d, it is more likely that those feature vectors, will be linearly separable in an M dimensional space. And once the feature vectors are linearly separable in the M dimensional space, then the linear combination of the outputs of this hidden layers is likely to give me a class belongingness. And that linear combination is decided by the weight vectors by the connection weights from the hidden layer nodes to the output layer nodes.

So, we also have to learn that what should be the connection weight w_{ij} from say i^{th} node in the hidden layer to the j^{th} node in the output layer. So, this is the second level of learning, so in the first level of learning for every radial basis function, we try to learn what is the receptor and what is the spread of radial basis function.

And for, in the second level, we try to learn what is the connection weight from the hidden layer nodes to the output nodes. As we have discussed in the previous class that the usual way, a common method of learning the radial basis function is if you are given a set of feature vectors of the training purpose. Suppose, value of $M = 3$, so what we do is, we partition weight of cluster the set of feature vectors into the number of clusters.

(Refer Slide Time: 12:29)



So, if we have M number of nodes in the hidden layer and I have say N number of feature vectors, N number of feature vectors, which are given for training purpose and I have M number of nodes in the hidden layer, obviously in this case N has to be greater than M. Otherwise, clustering N number of feature vectors into N number of clusters does not make any sense.

So, I have to have more number of feature vectors than the number of clusters that I have to form. So, I cluster this N number of feature vectors into M number of clusters and I can assume that centroid or mean of every cluster represent the corresponding receptor. So, if I take i^{th} cluster represents the receptor or the i^{th} radial basis function, so the situation is something like this.

If I have a set of feature vectors, say these are the feature vectors belonging to different classes. Typically, what I do is, I cluster this feature vectors into three different clusters. Every cluster center now represents a receptor, so this is one receptor, this is one receptor, this is one receptor. So, this is the receptor t_1 , this is the receptor t_2 and this is receptor t_3 .

So, the first operation we have to perform is the clustering of the feature vectors and these clustering operations we will discuss in details in future lectures. Now, once I have these different receptors, to find out what should be the spread of a particular radial basis function, what you do is for say i^{th} receptor, I find out P number of nearest neighbors or P number of nearest receptors and for this P number of nearest receptors, I compute what is the mean distance or root mean square distance.

So, there are different possibilities I can take any value I can choose any value out of these P number of receptor. So, what I do is the way I compute σ_i for the i^{th} cluster, for the i^{th} radial basis function is I have t_i , which is the receptor for the i^{th} radial basis function and then I take P number of nearest receptors, which are nearest to t_i . So, suppose one such receptor is t_k , so

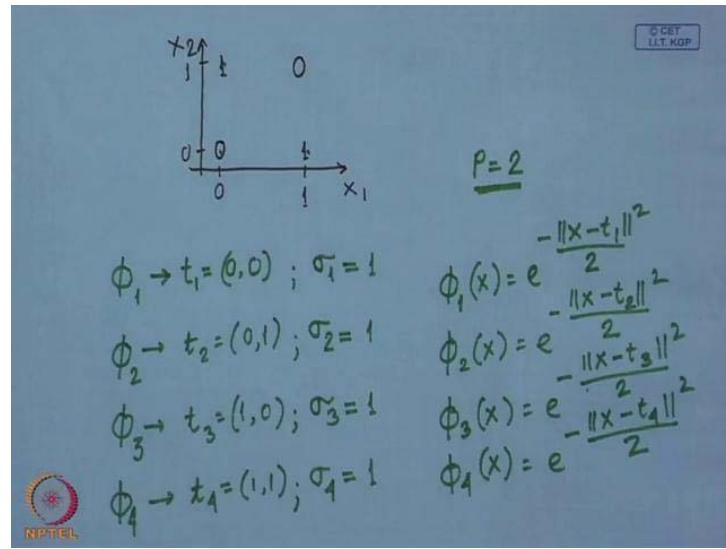
what I do is I compute $\sigma_i = \sqrt{\frac{1}{P} \sum_{k=1}^P (t_i - t_k)^2}$. So, this defines the spread of the i^{th} radial basis

function.

So, for every i^{th} radial basis function, I have t_i and have σ_i and once these 2 are known then

my radial basis function $\phi_i(X) = e^{-\frac{\|X-t_i\|^2}{2\sigma_i^2}}$. Now, let us see that by using this concept whether I can make a linear classifier using the radial basis function concept for the XOR problem and XOR is very, very common problem, which is used for illustrating such operations.

(Refer Slide Time: 17:19)



So, as we have said earlier, if I take an XOR function I have a 2-dimensional feature vector binary feature vector having components x_1 and x_2 . Suppose, this represents 0, this is x_1 equal to 1 here. I have x_2 is equal to 0 and here I have $x_2 = 1$. The value of the XOR function when it is $(0, 0)$ is equal to 0, $(0, 1)$ the value is 1. $(1, 0)$ the value is 1 and $(1, 1)$ again the value is equal to 0. So, find that here I have 2 dimensional binary feature vectors and So what I do is this 2 dimensional feature vector, I want to cast into a four dimensional space, by using four radial basis functions. So, I have the radial function, radial basis functions ϕ_1 , ϕ_2 , ϕ_3 and ϕ_4 .

For ϕ_1 , I choose $t_1 = (0, 0)$ that is the receptor of the radial basis function ϕ_1 . Similarly, for ϕ_2 , I can choose $t_2 = (0, 1)$, that is the receptor of the radial basis function ϕ_2 . Similarly, the receptors of other radial basis functions I can choose as $t_3 = (1, 0)$ and for this I choose $t_4 = (1, 1)$. So, these are the four receptors for the four radial basis functions.

Next, I had to choose the spread σ_1 . For the first radial basis function I have to choose σ_2 for the second radial basis function, σ_3 for the third radial basis function and σ_4 for the fourth radial basis function. Now, for this for every receptor I have to find out P number of nearest receptors and suppose I choose that the value of $P = 2$. Now, here you find that for every receptor there are three neighbors, two of the neighbors are at a distance, are at distances of 1 and one of the neighbors is at a distance of 1.4, that is $\sqrt{2}$. So, that is easily verifiable from

here I have receptor over here, which is t_1 , t_2 is at a distance 1, t_3 is at a distance 1, but t_4 is at a distance of $\sqrt{2}$, which is 1.4 or 1.414.

So, when I take $P = 2$, I have to take two nearest neighbors both of them are at distance 1 and root mean square distance of these two distances will also be equal to 1. So I have spread $\sigma_1=1$, I have spread $\sigma_2 = 1$, have spread $\sigma_3 = 1$ and have spread $\sigma_4 = 1$. So, I get $\phi_1(X)$, which

is of the form $e^{-\frac{\|X-t_1\|^2}{2}}$ and σ_1 being equal to 1.

Similarly, for $\phi_2(X)$, I will have $e^{-\frac{\|X-t_2\|^2}{2}}$, $\phi_3(X)$ will be $e^{-\frac{\|X-t_3\|^2}{2}}$ and $\phi_4(X) = e^{-\frac{\|X-t_4\|^2}{2}}$. So, if I compute to these values for each of the feature vectors taking $(0, 0)$ is 1 of the feature vector $(0, 1)$ as another feature vector $(1, 0)$ as another feature vector and $(1, 1)$ as another feature vector the functional values will be something like this.

(Refer Slide Time: 22:44)

Input	ϕ_1	ϕ_2	ϕ_3	ϕ_4	$\sum w_i \phi_i$	Output
0 0	1.0	0.6	0.6	0.4	-0.2	0
0 1	0.6	1.0	0.4	0.6	0.2	1
1 0	0.6	0.4	1.0	0.6	0.2	1
1 1	0.4	0.6	0.6	1.0	-0.2	0
	-1	+1	+1	-1		



So, I put that in the form of a table here I have input feature vectors, inputs are $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$ and I have the RBF functions ϕ_1 , ϕ_2 , ϕ_3 and ϕ_4 . So, when you input the feature vector $(0, 0)$ to ϕ_1 you find that your $X = t_1$. So, this exponent is equal to 0, which means that $\phi_1(X) = 1$. So, this $\phi_1(X)$ over here this will be 1.0. Similarly, for $\phi_1(X)$ my X is $(0, 0)$, t_2 is $(0, 1)$, so if I compute this $\phi_2(X)$, you will find that this $\phi_2(X) = 0.6$.

Similarly, I just put the values over here. $\phi_3(X)$ will also be 0.6 and $\phi_4(X)$ will be 0.4. When the input vector is (0, 1), $\phi_1(X)$ will be 0.6, $\phi_2(X)$ will be 1.0, $\phi_3(X)$ will be 0.4, $\phi_4(X)$ will be 0.6. For (1, 0) $\phi_1(X)$ is 0.6, $\phi_2(X)$ is 0.4, $\phi_3(X)$ is 1.0, $\phi_4(X)$ is 0.6 again and for the input feature vector (1, 1), I have $\phi_1(X)$ is equal to 0.4, $\phi_2(X)$ will be 0.6, $\phi_3(X)$ will be 0.6 and $\phi_4(X)$ that will be 1.0.

So, you find that given a 2 dimensional feature vector (0, 0) this has been cast into a 4 dimensional feature vector where the components of this 4 dimensional feature vector are (1.0, 0.6, 0.6, 0.4). Similarly, (0, 1) is a 2-dimensional input feature vector, which has been cast into a 4 dimensional feature vector the components being (0.6, 1.0, 0.4, 0.6). So, every input feature vector where the input feature vector is a 2-dimensional feature vector. Every 2-dimensional input feature vector is converted to a four dimensional feature vector by using four radial basis functions. Now, if I take the linear combination of this and for linear combination for $\phi_1(X)$, if I give an weight of, say I give the weight for $\phi_1(X)$, I give an weight of -1, for $\phi_2(X)$, I give an weight of +1 for $\phi_3(X)$, I give an weight of +1, for $\phi_4(X)$, I give an weight of -1.

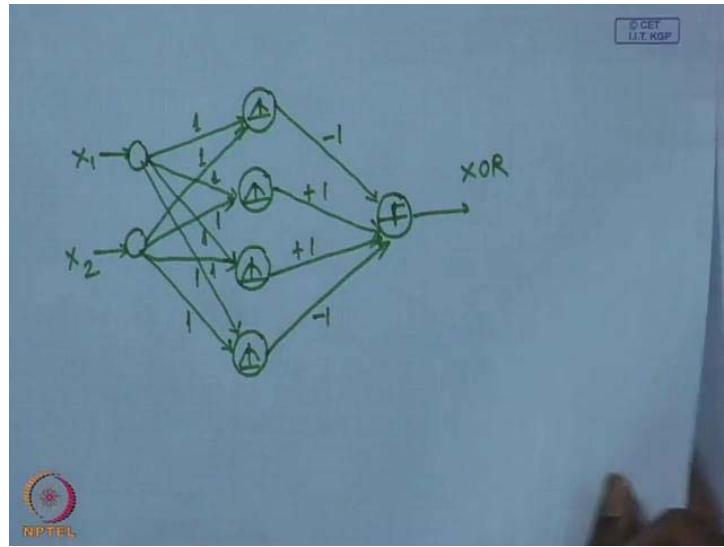
So, function that I will finally, compute at the output at a node in the output layer will be $-\phi_1(X) + \phi_2(X) + \phi_3(X) - \phi_4(X)$ and if I compute this, let us see what are the values that I get.

So, here I will write, $\sum_{i=1}^4 w_i \phi_i(X)$, where i varies from 1 to 4. So, here it will be 0.6+0.6 is 1.2

-1.4, this will be -0.2. Similarly, here it will be 1.4 - 1.2, so it will be +0.2. Here, again it will be 1.4 - 1.2, so this is +0.2 and here it will be 1.2 - .4, so this is - 0.2.

And if I take a decision that if the value is more than 0 the output will be 1, if it is less than 0 the output will be 0, then the final output that we have is here I write output. This will be 0, this will be 1, this will be 1 and this will be 0. So, which is nothing, but the XOR function output. So, over here the architecture of the radial basis function that we have used is we had two input layer nodes.

(Refer Slide Time: 28:09)

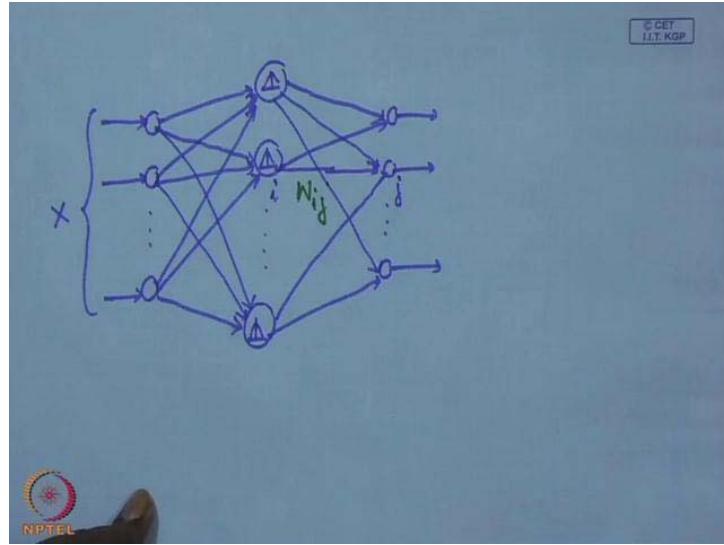


Where x_1 is fed to one node and x_2 is fed to another node, I had 4 nodes in the hidden layer, which computes the radial basis function. And I had 1 node in the output layer which I can say that it is finding out, it is a non-linear operator or a threshold operator. The connections are like this, where each of this connection has a connection weight is equal to 1.

And over here these connections are as you can see over here ϕ_1 two output layer node has a connection weight of -1, ϕ_2 two output layer node has a connection weight of +1, ϕ_3 two output layer node again has a connection weight of +1, ϕ_4 two output layer node has a connection weight of -1. So, here the connections are -1, +1, +1, -1 and this output actually gives me the XOR function, okay?

So, this example clearly shows that by casting the two-dimensional feature vectors into a four-dimensional feature vector. I can implement the XOR function using a linear network or a single layer perceptron because this part is nothing, but a single layer perceptron. Now, let us theoretically try to find out or try to find out an expression for the training of the output layer or how do I find out this connection weights. So, in general I have a network something like this.

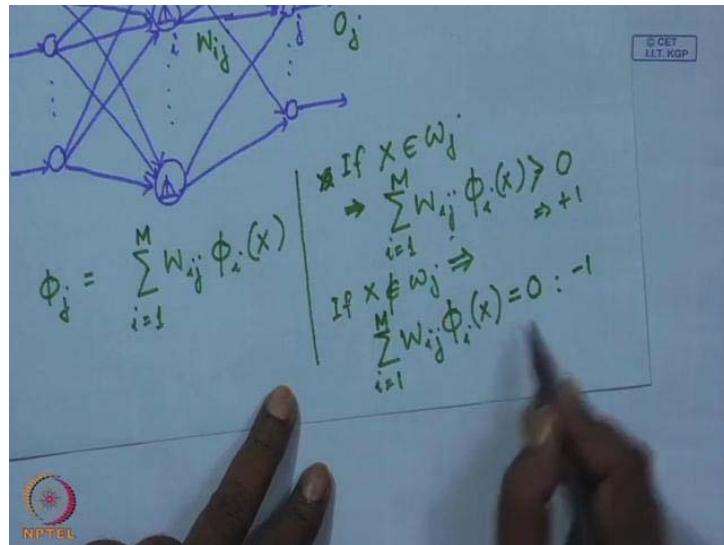
(Refer Slide Time: 31:04)



I have a set of input layers, set of input layer nodes, I have a set of hidden layer nodes and I have a set of output layer nodes. The feature vector is fed to the input layer nodes, so here I feed x , which are fed to the hidden layer nodes through connection weights, which are one and outputs of this hidden layer nodes. These are my radial basis functions. Outputs of the hidden layer nodes are connected to the output layer nodes.

So, like this and I take the output from every output layer node. So, if my input feature vector x belongs to say i^{th} class then output of the i^{th} output layer node will have a high value likely to be 1 and outputs of all other output layer nodes will have a low value likely to be 0. I assume that the i^{th} node in the input layer is connected to the j^{th} node of the output layer, through a connection weight say W_{ij} .

(Refer Slide Time: 33:07)



So, given this, if I say the output of the j^{th} layer node is o_j , I will have o_j is equal to $\sum_{i=1}^M w_{ij}\phi_i(X)$ for an input vector X and this summation I have to compute over all nodes in the hidden layer.

So, here I will have this summation has to be computed over i 1 to M , as I have M number of nodes in the hidden layer. And naturally over here, if this feature vector, so I will write, if X belongs to class ω_j then I had to have $\sum_{i=1}^M w_{ij}\phi_i(X) > 0$.

I will put this as +1 and if X does not belong to ω_j that indicates that $\sum_{i=1}^M w_{ij}\phi_i(X) < 0$ or I can

also put it as -1. So, let us assume that if X belongs class ω_j $\sum_{i=1}^M w_{ij}\phi_i(X) = 1$ and if X does not belong to ω_j this has to be 0 and that is what have to be the output from the j^{th} node in the output layer.

Now, taking this, now I can go for training of the output layer that means I have to find out what should be the values of this w_{ij} . Now, if I compute only the connection weights, if I right now consider only the connection weights, which are connected to the j^{th} node in the output layer then for every vector X_k , suppose I have capital N number of vectors.

(Refer Slide Time: 35:58)

$$x_k ; k=1 \dots N$$

$$\phi_i(x_k) \rightarrow \phi_{ik}$$

$$\sum_{i=1}^M w_{ij}\phi_{ik} = +1 \text{ if } x_k \in \omega_j$$

$$= 0 \text{ if } x_k \notin \omega_j$$

So, I have vectors X_k for k varying from 1 to capital N . I have capital N number of input vectors, which are given for training purpose or for learning as we are using supervised

learning, then $\phi_i(X_k)$. For simplicity I will write this as ϕ_{ik} . now, by using this I can, as I

said that $\sum_{i=1}^M w_{ij} \phi_{ik}$

So, my condition is, if you remember this 1, if you remember this by 1. So, $\sum_{i=1}^M w_{ij} \phi_{ik}$, for i

varying from 1 to M. This has to be equal to +1 if X_k belongs to class ω_j , and this has to be 0 if X_k does not belong to ω_j . So, this is the output that I expect, so for X_k I have for every X_k , I have such a kind of linear equation that this summation will be either +1 or 0 and all those capital N number of equations now, I can write in the form of a matrix.

(Refer Slide Time: 38:04)

So, in the matrix form this can be written as, let me write the matrix equation that ϕ_{11} , which means $\phi_1(X_1)$, ϕ_{21} that is $\phi_2(X_1)$.

$$\begin{bmatrix} \phi_{11} & \phi_{21} & \phi_{31} & \dots & \phi_{M1} \\ \phi_{12} & \phi_{22} & \phi_{32} & \dots & \phi_{M2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{1N} & \phi_{2N} & \phi_{3N} & \dots & \phi_{MN} \end{bmatrix} \begin{bmatrix} w_{1j} \\ w_{2j} \\ \vdots \\ w_{Mj} \end{bmatrix} = \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{Nj} \end{bmatrix}$$

Whatever is the output of individual middle layer nodes or hidden layer nodes. This equation simply makes a linear combination of outputs of the hidden layer nodes for the input feature 1 or X_1 . Where every b_{ij} will be equal to 1, if the corresponding X_i belongs to class ω_j and that will be equal to 0, if the corresponding X_i does not belong to class ω_j . So, every b_{ij} will assume a binary value either 0 or 1. So, this b_{ij} will be equal to 1. If X_i the corresponding

input vector X_i belongs to class ω_j , the j^{th} class or it will be equal to 0, if X_i does not belong to ω_j . So, this is the kind of situation that I have.

(Refer Slide Time: 41:43)

$$\begin{aligned}\phi w_j &= b_j \\ w_j &= \phi^+ b_j \\ e &= \phi w_j - b_j \\ J(w_j) &= \| \phi w_j - b_j \|^2 \\ \nabla J(w_j) &= 2 \phi^t (\phi w_j - b_j) \\ w_j &= (\phi^t \phi)^{-1} \phi^t b_j\end{aligned}$$

This whole expression, this matrix equation I can write in a short form, that is $\phi W_j = b_j$ where this ϕ is this matrix, W_j is weight vector which are connected to the output layer node j and b_j is the output of the j^{th} node in the output layer, which is represented in the form of a vector like this for different input vectors. So, if the network is properly trained, that is of all the w_{ij} has got the trained value then this equation should be satisfied.

But, what we are trying to do is, we are trying to train the network that means we are trying to set the weights W_j , so you cannot expect that this equation will be satisfied initially. So, this equality is not satisfied then what I can do is I can define an error e which is nothing but, $\phi W_j - b_j$. Now, training involves adaptation of this weight W_j , so that this error can be minimized.

So, in order to do that as we have done one earlier for mean square error optimization for mean square error technique for classifier learning or classifier training. I can also define here a function, criteria function $J(W_j)$, which is given by $\| \phi W_j - b_j \|^2$ and then I take the gradient with respect to W_j , so $\nabla J(W_j)$ which will be simply $2\phi^t(\phi W_j - b_j)$ and by equating this to 0 what we get is $W_j = (\phi^t \phi)^{-1} \phi^t b_j$.

And as you have seen earlier this $(\phi^t \phi)^{-1} \phi^t$, this is what is called pseudo inverse and that is represented as ϕ^+ . So, we have $W_j = \phi^+ b_j$, where this b_j is defined beforehand, every component of b_j will be either 1 or 0.

It will be equal to 1 in the corresponding feature vector, input feature vector belongs to plus ω_j and the component will be equal to 0 if the corresponding input feature vector does not belong to plus ω_j . So, I have this vector b_j , ϕ actually indicates that what should be the output of the hidden layer nodes for every feature vector, so from that I compute what is my matrix ϕ . So, once I have this matrix ϕ^+ and I have this b_j I can compute what will be the connection weights for different nodes in the hidden layer to the j^{th} output layer node.

This if I do for every output layer node I can compute what is the connection weight from different outputs of the hidden layer nodes to different output layer nodes and that is what completes my training of the RBF neural network and the RBF neural network will be ready for classification. Now, if you compare this RBF neural network with against say multilayer perceptron you find that the training of the RBF neural network is faster than the training in multilayer perceptron, because in case of multilayer perceptron the training is done by back propagation algorithm which takes large number of iterations. So, the training of the RBF neural network will be faster than training of the multilayer perceptron.

The second advantage is that I can easily interpret, what is the meaning or what is the function of every node in the hidden layer, which is difficult in case of multilayer perceptron. I cannot easily interpret the role of different nodes in the hidden layer in case of multilayer perceptron. And not only that I also cannot easily decide that what should be the number of hidden layers and what should be the number of nodes in every hidden layer. So, those are the difficulties in case of multilayer perceptron, which is not there in case of RBF network.

However, RBF network has a disadvantage that though the training is faster, but you find that the classification takes more time. In case of RBF network than in case of MLP, because in case of RBF network every node in the hidden layer has to compute the radial basis functional value for the input feature vector, which is time consuming. So, the classification in case, the classification in case of RBF network takes more time than the classification time in case of multilayer preceptor, okay? So, with this so we come to a conclusion on the new radial basis function neural network. Now, over here I will just briefly discuss about another kind of classifier which is called a support vector machine.

(Refer Slide Time: 48:38)

The image shows handwritten notes on a blue background. At the top, it says "Support Vector Machine". Below that, the formula $g(x) = w^t x + b$ is written. It is followed by two conditions: $w^t x + b > 0 \Rightarrow x \in \omega_1$ and $< 0 \Rightarrow x \in \omega_2$. Below these, there is a row of symbols: x_i , $y_i = \pm 1$, and $\underline{y_i(w^t x_i + b) > 0}$. The NPTEL logo is visible at the bottom left.

So, I briefly discuss support vector machine. So, support vector machine is another type of linear classifier. So, if you remember what we discussed in case of a linear classifier, that given a two class problem, we have said that I can define a discriminating function say $g(X)$, which is of the form say $W^t X + b$ and we have said in case of linear discriminator that if this $g(X)$ and $W^t X + b$ this is greater than 0 that indicates that feature vector X belongs to class ω_1 . If this is less than 0 then feature vector X belongs to class ω_2 .

So, here we find that for classification purposes the actual value of $g(X)$ is not really very important, but what is important is what is the sign of $g(X)$. If the sign is positive I infer that X belongs to class ω_1 if the sign is negative I infer that X belongs to class ω_2 , right? So, over here with every X , if I or if with every X_i , I indicate a number say Y_i that Y_i can be either +1 or -1.

In that case this $Y_i(W^t X + b)$ it will always be greater than 0. If the sample X_i is properly classified, which is quite obvious, because if I say that Y_i equal to +1 for a sample X_i which belongs to class ω_1 and for a sample which belongs to class ω_1 this $W^t X_i + b$ is greater than 0, Y_i is also positive.

So, $Y_i(W^t X + b)$ will obviously be greater than 0, if X_i belongs to class ω_2 then $W^t X_i + b$ will be less than 0. For that I have set $Y_i = -1$, $Y_i(W^t X + b)$ will obviously be greater than 0.

And this is a concept that actually we have used when we have discussed about the perceptron criteria or designing the linear classifier, that is for every feature vector belonging to class ω_2 , we have negated the feature vector before we try to design the classifier, so that for every feature vector irrespective of whether the feature vector belongs to class ω_1 or the feature vector belongs to class ω_2 , my discriminant function value will always be positive.

If the feature vector is correctly classified, so that is true if the feature vector belongs to class ω_1 or even if the feature vector belongs to class ω_2 , because of the feature vectors belonging to class ω_2 before trying to design the classifier we have negated this feature vector. So, we will discuss about the support vector machine in more details in our next class.

Thank you.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 29
Support Vector Machine

Hello. In the last two classes, we have discussed about the radial basis function neural network. And We have also discussed the about if I compared the radial basis function neural network with multilayer perceptron, what is the advantage and disadvantage of the radial basis function neural network with respect to the multilayer perceptron? So, we have said that in case of radial basis function neural network, the network consists of 3 different layers. One of the layer was which is present in all types of neural network that is the input layer, which basically accepts the input feature vector and forward that input feature vector to the layers above it.

In case of RBF neural network, we have 1 hidden layer and 1 output layer. The neurons in the output layer basically determine that what the class belongingness of the input feature vector. The function of the neurons in the hidden layer is to compute our radial basis function of value and through this radial basis function, what we effectively do is we map the input feature vector from its original dimension to a higher dimensional space through a non-linear mapping or a non-linear transformation.

This non-linear mapping is actually done by radial basis functions. And the purpose why we perform this transformation a non-linear transformation from a lower dimensional space to a higher dimensional space is that if you increase the dimensionality of the feature vectors, then the feature vectors belonging to different classes, if they are not linearly separable in a lower dimensional space, it becomes quite likely that they become linearly separable in higher dimensional space.

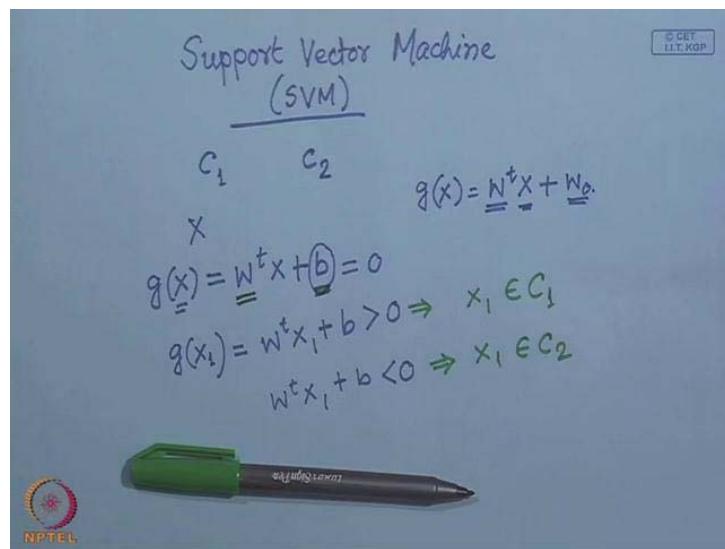
So, that was the basic motivation that we have in case of a radial basis function neural network. As by increasing the dimension of the input feature vectors, we are increasing the possibility of linear separability. So, the output layer of the radial basis function neural network simply computes linear combination of the outputs of the middle layer nodes. Based on this linear combination of the output of the middle layer nodes, it decides to which class the input feature vector X should be classified.

So, as a result, the output layer nodes that we have been starting from the middle layer nodes to the output layer nodes. So, that section of the radial basis function neural network is effectively a linear classifier. So, the advantage that we have in case of radial basis function neural network is that I knew how many hidden layers I have to have. It is only 1. In case of multilayer perceptron, I cannot easily decide that how many hidden layers I should have.

Similarly, how many nodes per hidden layer that in case of multilayer perceptron that is also cannot be easily decided bias. Whereas in case of RBF network radial basis function neural network, I can set how many nodes in the hidden layer I need to have because it is simply the dimensionality of the space to which I want to cast my input feature vectors. The other advantage is the interpretation of the functions of the hidden layer nodes, which is quite easy in case of RBF neural network. But the interpretation of the functionality of the nodes in the hidden layers is not that clear in case of multilayer perceptron.

Also, the training in case of hidden layer is faster than the training in case of multilayer perceptron. The only disadvantage, apparently disadvantage of the RBF neural network is that the classification task in case of RBF neural network takes more time than the classification in case of multilayer perceptron. So, this is what we have discussed over last 2 classes. Today, we are going to discuss about another kind of classifier, which is called a support vector machine.

(Refer Slide Time: 05:26)



So, we will discuss today the classifier known as support vector machine or SVM. So, when we start our discussion on support vector machine, let us recapitulate one of the previous classifiers, the linear classifiers that we have discussed that is a linear discriminant function.

So, during, one of the early lectures, we have said that if we have an input feature vector x and we define a linear discriminant function say $g(X)$. So, for simplicity, let us consider that we are considering a two class problem.

We have two classes. One is class c_1 , the other one is class c_2 . So We are talking about two class problems, class c_1 and class c_2 . An unknown feature vector say X is to be classified as either belonging to class c_1 or belonging to class c_2 . And for doing this, when we talked about the linear discriminant function, we have defined a linear discriminant function say $g(X)$, which was put in the form $W^t X + b$ or maybe in the early lectures, we had put $g(X)$ as $W^t X + W_0$. It was something like this or X is the input feature vector, W is the weight vector and the W_0 is a bias term.

So, this is a linear function in a 2-dimensional space. If our feature vector is a 2-dimensional vector, then this linear equation represents a straight line. If our input feature vector is a 3-dimensional feature vector, then this linear equation, if I put this equal to 0, then I get a linear relation $W^t X + b = 0$. So, this linear equation in 2 dimensions represents a straight line. This linear equation in 3 dimensions represents a plane. Whereas if I increase the dimension where the dimensionality of the feature vector is more than 3, this linear equation actually represents what is called a hyper plane and W is nothing but a vector, which is perpendicular to that hyper plane.

So, this vector W it represents the orientation of the hyper plane in my d dimensional space where d is the dimensionality of the feature vector. Whereas, this term b or coming to the previous term w naught which is a constant it represents, what is the position of that hyper plane in my d dimensional space. So, the vector w represents the orientation of the hyper plane and the value d or W_0 it represents the position of that hyper plane in my d dimensional space.

So, this b , term b is usually known as a bias term, which is biasing the position of the hyper plane in the d dimensional space. Now, coming to our classification problem for every feature vector X , I want I have to compute this linear function $W^t X + b$. If this X lies on positive side of the hyper plane, then I will have $g(X)$. Let us take a specific vector say $g(X_1)$, this is a vector, specific vector. The first vector, which will be equal to $W^t X_1 + b$, if this X_1 is on the positive side of the hyper plane, then I will have $W^t X_1 + b$, will be equal to greater than 0.

And if X_1 belongs to negative side of the hyper plane, then I will have $W^t X_1 + b$, which will be less than 0. And if X_1 lies on the hyper plane, then I will have $W^t X_1 + b = 0$. So, this is a hyper plane, which defines my d dimensional space into 2 half spaces. In one of the half space, if I take a vector $g(X)$ for that particular vector X will be positive, if I take the vector X into the other half space, then $g(X)$ for that vector will be negative. I can have my classification rule that if $W^t X_1 + b > 0$, then I infer that this X_1 will be classified to class c1 that is it belongs to class c1, where as if $W^t X_1 + b < 0$, then I can infer that this X_1 belongs to class c2.

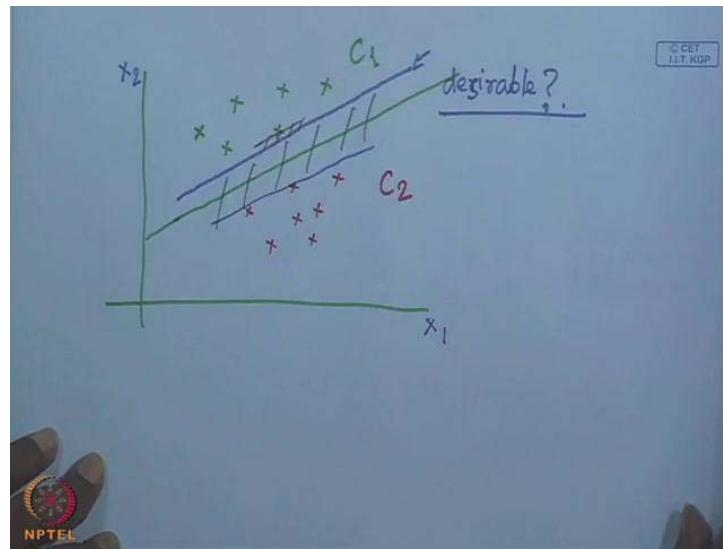
So, this is my classification rule. So, what you said is initially we had to train this classifier that means I have to find out what is this w vector w and what is this bias term b, so that this $g(X)$ with that W, the trained value of w and the trained value of b can be used as a classifier. So, for training, we will have used supervised learning so we have been given a large number of samples, some of the samples coming from class c1 and some of the samples coming from class c2.

So, using these training samples, when I try to train the network, then training of the network was done in an iterative fashion. So, for every training sample belonging to class c1, we started with an initial value of W and b and for every training sample belonging to class c1, we have tried to see that whether $W^t X_1 + b > 0$ or not because we have taken a sample from class c1. So, it has to be greater than 0.

If it is not greater than 0, then we have modified W and b in such a way that the position of the hyper plane or as well as the orientation of the hyper plane is so modified that that particular X, which is taken from class c1 is moved to positive side of this hyper plane. Similarly, if I take a vector from class c2, we have checked whether this $W^t X + b$ where X is taken from class c2 is negative or not. If it is not negative, then again we have modified W and b.

So, there again, we have to see that if this is negative or not if it is not negative, then we have modified W and b in such a way that that particular X is moved to negative side of the hyper plane. So, this is what we have done in case of linear discriminator. And we have also said there that if a sample is just beyond this hyper plane $W^t X_1 + b = 0$, even then it might be correctly classified, but the classifier that I so obtain, so I can have a situation something like this.

(Refer Slide Time: 14:47)



I have 1 feature vector. Let us take 2 dimensional feature vectors something like this. So, these are the feature vectors, which say belong to class c_1 . And I can have a set of feature vectors something like this, which belong to class c_2 . This is my x_1 dimension. If I have a 2-dimensional feature vector, and this is my x_2 dimension. So, these are the 2 components of the feature vector. Now, if I have a classifier or a hyper plane into dimension, which is a straight line as we said earlier, it is something like this. And these are the feature vectors, which are given for supervised planning.

So, if I have a situation something like this, even here you find that this straight line in d dimension, it is a hyper plane it correctly classifies all these feature vectors, which are given for training of the classifier but, whether this type of classifier is desirable. Is it desirable? Obviously, a classifier of this nature is not desirable because this classifier gives a large bias in favor of class c_2 , whereas it puts a penalty against class c_1 . The reason is this entire space from here to here this is given to class c_2 whereas the margin to class c_1 is only this much. Rather than this classifier, I would prefer to have a classifier somewhere over here.

So, that the training vectors both from class c_1 and class c_2 , both are equally apart from my classifier or the plane separating these two different classes and the support vector machine actually tries to find a classifier, which will be positioned in a form something like this. Now, let us see how such a support vector machine can actually be designed. So, the basic aim of the support vector machine is given.

If I am standing on one side of the boundary, and if there is a possibility that I can cross the boundary on the other side, I will feel safer, if my distance from the boundary is larger. If my distance from the boundary is very small, so this is my boundary. I am standing somewhere over here, and then a little disturbance or a little noise can push me to the other side of the boundary in which case I will be misclassified but if I am standing somewhere over here where this is my boundary surface, then the margin that I have is quite large.

So, to push me to the other side of the boundary to make me misclassified, there has to be a large disturbance. So, I feel that I am safer if my distance from the boundary is very large and I am not safe if my distance from the boundary is very small. So, this is the kind of situation that I have. So, what the support vector machine does is, the way of the support vector machine tries to design the classifier is that it tries to maximize the distance of the separating boundary between the two classes by maximizing the distance of the separating plane from each of the feature vectors whether the feature vector belongs to class c1 or the feature vector belongs to class c2. And out of this, when I talk about the support vectors, I will come to that point a bit later.

(Refer Slide Time: 19:36)

x_i

$$w^T x_i + b > 0 \rightarrow x_i \in C_1$$

$$w^T x_i + b < 0 \rightarrow x_i \in C_2$$

$$x_i, y_i \quad y_i = \pm 1$$

$$y_i(w^T x_i + b) > 0$$

$p \rightarrow$ unknown vector.

NPTEL

So, what I have is suppose that I have a vector X_i , if this vector X_i belongs to class c1, then I will have $W^T X_i + b > 0$, I can also put in the form of a dot product $W \cdot X_i + b > 0$ because this operation is same as this operation $W^T X_i + b > 0$, if this X_i belongs to class c1.

Then I have the situation $W^t X_i + b < 0$, if X_i belongs to class c2, so for c1, $W^t X_i + b$ will be positive here, $W^t X_i + b$ will be negative for c2.

Now, what I can do is when I go for designing of the classifier, I know to which of the class the sample X_i belongs. I know whether X_i belongs to class c1 or X_i belongs to class c2 so along with every X_i , I can assign class belongingness. So, what I can put is along with every X_i , I can also give a y_i , where this y_i can be either +1 or -1.

So, if X_i belongs to class c1, the corresponding y_i will be positive. If X_i belongs to class c2, the corresponding y_i will be negative. And if I feed the data in this form, I compute $y_i(W^t X_i + b)$, this will always be positive irrespective of whether X_i belongs to class c1 or X_i belongs to class c2.

Because if X_i belongs to class c1 then $W^t X_i + b$ is positive, y_i is also positive, which is specified. So, this product will be positive. If X_i belongs to class c2, then $W^t X_i + b$ will be negative, y_i is also -1. So, look at the product of these two, the product then will be positive.

So, this is what I will have. And using this concept, I can go for the designing of the classifier 1 and get w and b. Then for unknown feature vectors, say let us take an unknown feature vector p. This is an unknown vector, which has to be classified. I have to put this unknown feature vector p into either c1 or c2 using that w and b that has been obtained after designing the classifier. So, once if do that, I do not have anyone y_i because p is unknown. So, I simply compute $W.p + b$ where W and b, they have already been decided during the training process.

So, if this $W.p + b$ becomes greater than 0, I would classify this p to belong to class c1. If this is less than 0, I classify p to class c2. So, this is the kind of situation that I have. And as I said that it is of support vector machine, my aim is that I want to maximize that distance of the hyper plane or the separating boundary from each of the feature vector, so that every feature vector feels that they are safer so far as this classifier is concerned and the distance and to do that we have made a modification in our classifier design, linear classifier design. When we talked earlier that we consider the $W^t X_i + b$ should be greater than 0

(Refer Slide Time: 24:17)

x_i

$w^T x_i + b > 0 \quad w^T x_i + b >= \gamma$

$w^T x_i + b > 0 \rightarrow x_i \in C_1$

$w^T x_i + b < 0 \rightarrow x_i \in C_2$

$x_i, y_i \quad y_i = \pm 1$

$y_i (w^T x_i + b) > 0$

unknown vector.

for correct classification, we have said that I want $W^T X_i + b$ should be greater than some γ where γ we have said some margin and this margin is nothing but a measure of distance of X_i from the separating plane.

(Refer Slide Time: 24:40)

$w^T x + b = 0$

$\frac{w^T x + b}{\|w\|} \geq \gamma$

$w^T x + b \geq \underbrace{\gamma \|w\|}_{=1}$

$w^T x + b \geq 1 \text{ if } x \in C_1$

$\leq -1 \text{ if } x \in C_2$

$y_i (w^T x_i + b) \geq 1$

So, if I have a hyper plane whose equation is given $W^T X_i + b = 0$, distance of point X from

this hyper plane is simply given by $\frac{W^T X_i + b}{\|W\|}$. So, this is known from school level geometry.

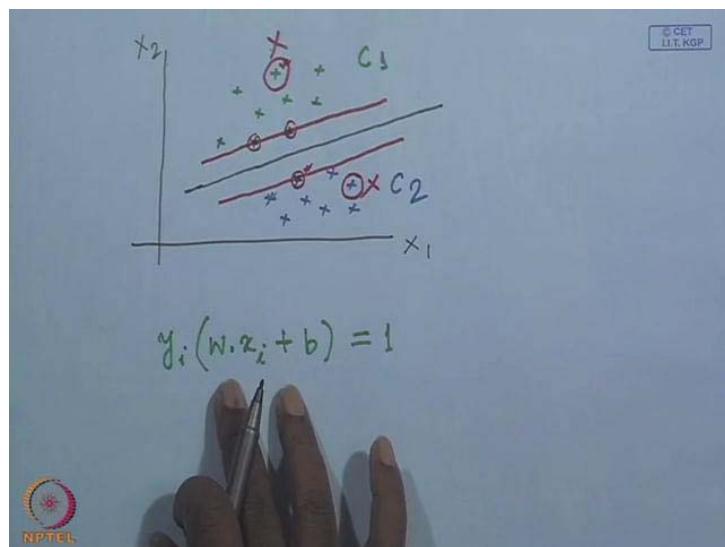
So, this is the distance of point X from this hyper plane $W^T X + b = 0$. And what we want is

we want that this has to be greater than or equal to some margin say γ . And my decision regarding the correct classification of the feature vectors is independent of scaling vector W because vector W simply tells me that what is the orientation of the plane. So, whatever scaling factor I apply to W , my decision remains the same.

So, I can simply put that this $W^t X + b$ has to be greater than or equal to γ . $\|W\|$. And by proper scaling, I can set that this is equal to 1. I can set this term γ . $\|W\| = 1$. This is simply a matter of scaling in which case for every vector I have to have a situation that $W^t X + b$ has to be greater than or equal to 1, if X belongs to $c1$. It has to be less than or equal to -1, if X belongs to class $c2$.

And now, for a vector X_i , if I multiply by the corresponding y_i during the training during the learning process or training process, I have a situation that $y_i(W^t X_i + b) \geq 1$ will always be greater than or equal to 1. And this equality $y_i(W^t X_i + b) = 1$, if X_i is a support vector. It will be greater than 1, if X_i is not a support vector. So, then the question becomes that what is my support vector. Then, so to I explain what my support vector is.

(Refer Slide Time: 28:38)



I simply go to my 2 dimensional example. So, I have 2 dimensional vectors x_1 and x_2 . I have a set of feature vectors from say class $c1$, which is like this. So, this is class $c1$. And I have a set of feature vectors coming from class $c2$. So, I put it of this form. So, these are the feature

vectors, which are from class c2. Now, you can find that I can easily find out that if I draw a straight line over here.

If I draw another straight line over here, these are the feature vectors from the training vectors which are just on the boundary. So, any disturbance to the feature vectors will greatly affect this. If this feature vector is slightly disturbed, my classification result is not going to vary that much. So, I had to put my classifier somewhere in the middle of this two. This is a classifier, which will be more reliable or it is called more generalized. Where my risk of misclassification will be less.

And you find that the position of this classifier of this hyper plane depends on the position of this feature vector. It depends on the position of this feature vector. It depends on the position of this feature vector. It does not depend upon the position of this vector, feature vector even if I remove this feature vector from my training set or even if I remove this feature vector from my training set, the position of the hyper plane remains the same.

If I remove this feature vector from my training set, the position of the hyper plane is going to be different. So, these are the feature vectors, which are called support vectors. The other vectors are not support vectors. So, this support vector machine is again a linear machine whose design is greatly influenced by the positions of the support vectors. Its position is not that much influenced by the vectors, by the feature vectors, which are not support vectors.

So, that is why I said that from this that $y_i(W \cdot X_i + b) = 1$ when I have, when this X_i is a support vector. So, I $y_i(W^t X_i + b) = 1$ if X_i is a support vector, so this equality holds only for the support vectors. And now, what I have to do is if you look at this expression that what I want to do is this w dot x_i plus b is a measure of distance of point X_i from the plane $W^t X_i + b = 0$. So, this is a factor, which I want to maximize this. So, this is a factor, which has to be used for designing of my support vector machine.

(Refer Slide Time: 31:41)

x_1 + C_1

$w \cdot x + b = 0$

$\frac{w \cdot x + b}{\|w\|} \geq \gamma$

$w \cdot x + b \geq \gamma \|w\| = 1$

$w \cdot x + b \geq 1 \text{ if } x \in C_1$

$w \cdot x + b \leq -1 \text{ if } x \in C_2$

$y_i(w \cdot x_i + b) \geq 1$

I want that this distance or over this distance, the margin has to be as maximum as possible, which can be done from this expression by minimization of mod of w and at the same time by maximization of the bias b. So, when I try to get this support vector machine, I will try to minimize this W. Simultaneously, I want to maximize this b. So, I want to minimize this W.

(Refer Slide Time: 32:17)

$\phi(W) = \underline{W^t W} = \frac{1}{2} \underline{W \cdot W}$

$\frac{1}{2} \underline{W \cdot W}$

$\underline{y_i(w \cdot x_i + b) = 1} \leftarrow \underline{\text{Constraint}}$

So, this is, this W or the weight vector that I want to minimize. So, the minimization of W is same as if I want to minimize a function of w. This is nothing but $\phi(W) = \frac{1}{2} W^t W$ or $\phi(W) = \frac{1}{2} W \cdot W$. These two are same operations and for mathematical convenience which

will be clear later on. I just put a term half. So, half of w dot w that I want to minimize and if I want to minimize this, the trivial value will obviously be $W = 0$, which is not the solution that solution that I am looking for.

So, for minimization of this, I have to look for what is the other constraint that I have. My constraint is that for support vectors, I have to have $y_i(W \cdot X_i + b)$ have to be equal to 1. So, this is my constraint. So, I want to minimize W or I want to minimize $\phi(W)$ subject to the constraint that $y_i(W \cdot X_i + b) = 1$ or I say that this $y_i(W \cdot X_i + b) > 1$.

If X_i is a support vector, then this has to be equal to 1 and because my support vector machine depends upon the support vectors. So, I will not take this inequality, rather I will take this equality that is $y_i(W \cdot X_i + b) = 1$. That is the constraint and under this constraint, I had to minimize my weight vector W . So, because it is a constrained optimization problem, this problem can be converted to an unconstrained optimization problem by using the Lagrangian multiplier.

(Refer Slide Time: 34:36)

$$L(w, b) = \frac{1}{2}(w \cdot w) - \sum \alpha_i [y_i (w \cdot x_i + b) - 1]$$

$\alpha_i \rightarrow \text{Lagrangian Multiplier.}$

$$\frac{\partial L}{\partial b} = ?$$

$$L(w, b) = \frac{1}{2}(w \cdot w) - \sum \alpha_i y_i (w \cdot x_i) - \underbrace{\sum \alpha_i y_i b}_{+} + \sum \alpha_i$$

$$\frac{\partial L}{\partial b} = - \sum \alpha_i y_i = 0$$

$$\sum_{i=1}^m \alpha_i y_i = 0$$

So, I can put Lagrangian multiplier. I can define a function of the form $L(W, b)$ because as I said that I want to minimize W , I want to maximize b , $L(W, b) = \frac{1}{2} W \cdot W - \sum \alpha_i [y_i (W \cdot X_i + b) - 1]$, right? And I have to optimize this Lagrangian where this α_i is the Lagrangian multiplier. So, when I go for optimization of this Lagrangian

from our school level mathematics, we know that this optimization has to be done by taking the derivative of the Lagrangian with respect to W and by taking the derivative of the Lagrangian with respect to b .

So, if I take the derivative of this Lagrangian with respect to b , then what I have? So, what I want to compute is $\frac{\partial L(W, b)}{\partial b}$. This is what I want to compute. I have to equate this to 0. So,

for doing this, let us try to expand this equation. And let us see what this expression is. So, in the expanded form, I will have $L(W, b) = \frac{1}{2} W \cdot W - \sum \alpha_i y_i (W \cdot X_i) - \sum \alpha_i y_i b + \sum \alpha_i$. So,

this is the expression or the expanded form I have. Clearly if I take the derivative of this with respect to b . So, I want to compute $\frac{\partial L(W, b)}{\partial b}$. So, this $\frac{\partial L(W, b)}{\partial b}$ obviously from here, you

find that $\frac{1}{2} W \cdot W$ is independent of b . This $\sum \alpha_i y_i (W \cdot X_i)$ is independent of b . This $\sum \alpha_i$ is also independent of b .

So, only term that involves b is this $-\sum \alpha_i y_i b$. So, $\frac{\partial L(W, b)}{\partial b}$ will be simply minus $\alpha_i y_i$ and that has to be equal to 0. So, I get 1 constraint that $-\sum \alpha_i y_i$ that has to be equal to 0 for i varying from 1 to say m , where m is the number of feature vectors, which are different for designing the classifier. So, this is one of the constraints that I have. Next what I do is I will take the derivative of this Lagrangian 1 with respect to my weight vector W . So, then this is the same Lagrangian. I put in the expanded form. I take the derivative of this.

(Refer Slide Time: 38:55)

$$L(W, b) = \frac{1}{2} W \cdot W - \sum \alpha_i y_i (W \cdot X_i) - \sum \alpha_i y_i b + \sum \alpha_i$$

$$\frac{\partial L}{\partial W} = W - \sum \alpha_i y_i x_i = 0$$

$$W = \sum_{i=1}^m \alpha_i y_i x_i \quad \checkmark$$

Let me rewrite this Lagrangian. So, $L(W, b) = \frac{1}{2} W \cdot W - \sum \alpha_i y_i (W \cdot X_i) - \sum \alpha_i y_i b + \sum \alpha_i$

each of these summations will from $i = 1$ to m before if I have m number of feature vectors provided for designing the classifier. So now, if I take the derivative of this Lagrangian with respect to w , so what I have is I have $\frac{\partial L(W, b)}{\partial W}$. So, when I take $\frac{\partial L(W, b)}{\partial W}$, this will simply be the derivative of this with respect to W is nothing but $W - \sum \alpha_i y_i X_i$.

You find that this $\sum \alpha_i y_i b$ term is independent of W . This term $\sum \alpha_i$ is independent of W . So, derivative of these terms with respect to W will be equal to 0. And once I take this derivative, I have to take this $W - \sum \alpha_i y_i X_i = 0$, which gives me $W = \sum \alpha_i y_i X_i$, where this i will be from 1 to m as m is the number of training samples. So, through this derivative process, so this is the W that I have which is my weight vector. You find that this $\alpha_i y_i X_i$ is not a dot product. So, this is why $y_i X_i$, X_i is a vector, α_i is a scalar, y_i is also a scalar. So, this entire term is a vector.

So, through this optimization of the Lagrangian, I have got 2 equations. One is this $\sum \alpha_i y_i = 0$. And I have this expression for my weight vector W where $W = \sum \alpha_i y_i X_i$ where X_i is the i^{th} feature, training feature vector for i varying from 1 to m . So, these are the 2 expressions. Now, if I put this expression in my original Lagrangian, then let us see what we have. So, by putting this in my original Lagrangian expression as you said that the Lagrangian was in the expanded form.

(Refer Slide Time: 42:23)

$$\begin{aligned}
 L &= \frac{1}{2} W \cdot W - \cancel{\sum \alpha_i y_i b} - \sum \alpha_i y_i W \cdot X_i + \sum \alpha_i \\
 &= \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j (X_i \cdot X_j) - \sum \alpha_i \alpha_j y_i y_j (X_i \cdot X_j) \\
 &\quad + \sum \alpha_i \\
 &= \boxed{\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j (X_j \cdot X_i)} \\
 \alpha_i &\geq 0 \quad \sum_{i=1}^m \alpha_i y_i = 0
 \end{aligned}$$

It was $L(W, b) = \frac{1}{2} W \cdot W - \sum \alpha_i y_i (W \cdot X_i) - \sum \alpha_i y_i b + \sum \alpha_i$. So, in the expanded form,

which we said earlier. Now, over here, this $\sum \alpha_i y_i$, this term is equal to 0. So, this term $\sum \alpha_i y_i b$ simply gets cancelled. So, the expression I have is

$$L(W, b) = \frac{1}{2} W \cdot W - \sum \alpha_i y_i (W \cdot X_i) + \sum \alpha_i.$$

And if I put this value of W into this expression. Then what I will have is this will be simply

$$L(W, b) = \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j (X_i \cdot X_j) - \sum \alpha_i \alpha_j y_i y_j (X_i \cdot X_j) + \sum \alpha_i$$

because I have the term $W \cdot W$ dot product of the vector with itself. So, I had to have to have 2 different subscripts over here, one i and the other one I will put as j .

So, this will simply be $L(W, b) = \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j (X_i \cdot X_j) - \sum \alpha_i \alpha_j y_i y_j (X_i \cdot X_j) + \sum \alpha_i$. So, this

will be simply $L(W, b) = \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j (X_i \cdot X_j)$. So, $\sum \alpha_i$ does not matter.

So, I have the Lagrangian expression, which is this. So, for design, what I have to do is I have to maximize this Lagrangian with different values of α , which are my Lagrangian multipliers and Lagrangian multipliers are always positive. So, I have 1 constraint that α_i always have to be greater than or equal to 0. And the other constraints that we have are from here that $\sum \alpha_i y_i = 0$, i varying from 1 to m that has to be equal to 0. So, I have to find out such Lagrangian multipliers, which maximize this expression this Lagrangian expression.

When you try to maximize this Lagrangian expression, it is quite likely that some of the Lagrangians will be equal to 0. Some of the, few of the Lagrangian multipliers will be equal to 0 and few of the Lagrangian multipliers value will be very high. So, if a Lagrangian multiplier and $\alpha_i = 0$ that indicates that the corresponding training feature vector x_i is not a support vector.

If a particular α_i is very high for that indicates that the corresponding feature vector x_i has a high influence over the position of my decision surface or the hyper plane. The other possibility is of course, there if I get an α_i which is extraordinarily high, I can infer that the

corresponding feature vector, which is given it is a spurious point. It might have this. It is a disturbed point or it is an outlier.

All these different types of interpretations I can have over α_i . So, obviously if $\alpha = 0$, the corresponding x_i is not a support vector. So, it does not influence the position of the hyper plane. So, what I have to do is I have to use this α_i 's or I have to use with this α_i 's. The value of W that I compute is given by this. This W goes into my decision making process. So, as a result, my classification decision will be like this.

(Refer Slide Time: 48:48)

$$D(Z) = \text{sgn} \left(\sum_{j=1}^m \alpha_j y_j x_j \cdot Z + b \right)$$

What I will compute is for an unknown Z . If I have a feature vector Z , which is unknown, the classification decision, if I say that the decision is $D(Z)$, $D(Z)$ will be simply this. I have to

compute $D(Z) = \text{sgn} \left(\sum_{j=1}^m \alpha_j y_j X_j Z + b \right)$ because you find that my W is nothing but

$W = \sum \alpha_i y_i X_i$, okay?. What I have to compute is this $W \cdot Z$. So, that is what I am doing α_j simply, subscript i is replaced by subscript j so it does not matter.

So, I have to compute $\text{sgn} \left(\sum_{j=1}^m \alpha_j y_j X_j Z + b \right)$ and only the sign of this is important for

classification. I do not want; I do not need what is of value of this function. I simply need the sign of this function. So, sign of this is important for classification. If the sign is positive, then this Z will be classified to class $c1$. If the sign is negative, then this Z will be classified to class $c2$. So now, if I write the steps of the support vector machine design, the steps of the support vector machine will be something like.

One more point that I should mention over here as you are done in case of radial basis function, if the original set of feature vectors in lower dimensional space is not linearly separable, then I cannot have the support vector machine. This is because it assumes that the samples are linearly separated. So, if they are not linearly separable in lower dimensional space, I have to cast these feature vectors into higher dimensional space by using functions like radial basis functions, which are called as kernels functions. So, once I cast them into higher dimensional space, then by taking the samples in the higher dimensional space, I can try to design the support vector machine.

Then, again for classification, when we have classified this feature vector Z before classification, I had to cast that into same higher dimensional space. And after casting that into higher dimensional space, I have to compute this term in the higher dimensional space. Then I can compare the sign of this term. If the sign is positive, then it will be in class $c1$. If it is negative, it will become to class $c2$. So, I need two terms; one is W which has been obtained by optimization as this particular expression. The other one I need will be the value of b . So, the value of b , I can be compute as.

(Refer Slide Time: 52:12)

$$b = \frac{1}{2} \left[\min_{\{i | y_i = +1\}} \left(\sum \alpha_i y_i (x_i \cdot x_j) \right) + \max_{\{i | y_i = -1\}} \left(\sum \alpha_i y_i (x_i \cdot x_j) \right) \right]$$

This is the margin. $b = \frac{1}{2} \left[\underbrace{\min \left(\sum \alpha_i y_i (X_i \cdot X_j) \right)}_{i/y_i = +1} + \underbrace{\max \left(\sum \alpha_i y_i (X_i \cdot X_j) \right)}_{i/y_i = -1} \right]$. So, that is how I get

the value of the b and this W .

This W and this b goes in this expression for classification of an unknown feature vector Z . So, this is how our support vector machine work. As I said before, the support vector machine is again nothing but a linear machine. The only thing is it helps in placing the separating boundary between the two classes or it simply states where my hyper plane should be placed. So that the classifier that I get that is more robust or more generalized. Now, instead of a two class problem, if I have a multiclass problem, then I have to have multiple numbers of support vector machines to support. So, I have to have 1 support vector machine, which tells me whether a sample belongs to class c_1 , whether it does not belong to class c_1 . Then I have to have a support vector machine, which tells me whether a sample belongs to class c_2 or it does not belong to class c_2 .

So, I have multiple numbers of support vector machines for a multiclass problem. And it can be shown that if I have n number of classes, then what I need is n minus 1 number of support vector machines like I have to have $n-1$ number of linear classifiers. So, with this, I stop here today.

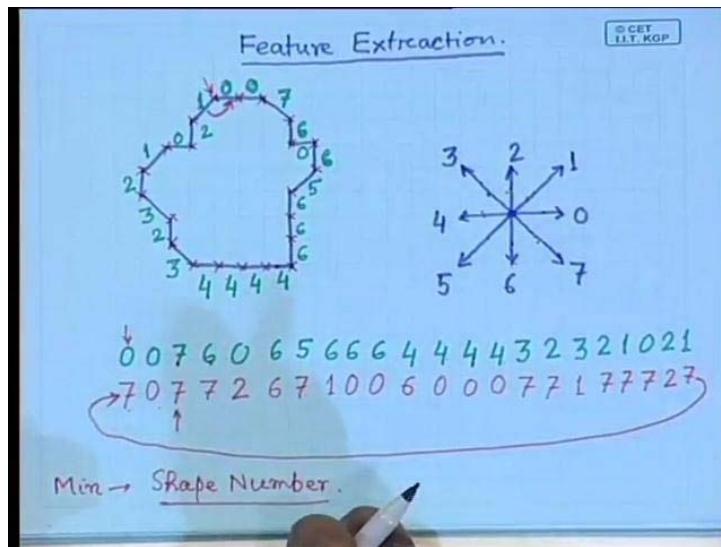
Thank you very much.

Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 3
Feature Extraction – II

Good morning. So, today we will continue with our discussion on feature extraction. In the last class we have started discussion on chain code and we have said that to make this chain code rotation invariant, we have to make some use something called Differential chain code. So, today as some of you have requested, I will just repeat the differential chain code formation, before I move to other ways of feature extraction. So, when we talk about chain code, let us have that let us assume that we have some sort of boundary pixels represented by a boundary something like this.

(Refer Slide Time: 00:59)



So, let us take a boundary of this form and to discuss about this chain code, we will assume the chain code which is formed by 8 neighbors. So, the convention that we used in the last class was like this, say these are the different directions of the line segments, which is used to piece wise linear approximate the given arbitrary boundary and this direction we took as 0, this direction we took as 1, this direction we took as 2, 3, 4, 5, 6 and 7. So, these are the 8 different directions which are used to form the chain code. So, in this figure, if I assume that I start the chain code operation from this point.

So, this is the starting point, then first movement I will have, so these are the points on my boundary actually, these are the boundary points, wherever I put across, so this becomes the set of boundary points. So, when I form this chain code starting from the first location that is this one, my initial direction of the line segment is from left to right, that is towards the east and that direction is marked as 0, so I put 0 over here. Similarly, the second line segment, that is also in the direction of 0, so I also put 0 over here. The third line segment it is in the south east direction and south east is marked as 7, so I put 7 over here.

The next movement is in the south direction which is marked as 6. The next one is again in the direction of east, so which is marked as 0; next direction is again in south, so it is marked as 6. So, this way this way if you continue you find that your chain code will be represented like this. So, starting from my starting point, the inter chain code becomes 0 0 7 6 0 6 5 6 6 6 4 4 4 4 3 2 3 2 1 0 2, then followed by 1. So, this becomes the chain code which represents a given boundary like this. Now, as I said that if the figure is rotated, in that case this chain code will be entirely different.

Say for example, instead of putting the figure like this, if I simply rotate the figure by 90^0 , if I make the figure of this form, in that case earlier one which was 0, that will now become this one which will be 7. So, the entire chain code representation which we had having the boundary position like this, if I rotate the boundary the entire chain code will be different. So, using this chain code I cannot recognize a shape which is rotated by an angle θ . So, what I need to do is I have to modify this chain code so that it becomes rotationally invariant.

So, for doing that what I said is instead of using this chain code, what I will do is to move from one segment to another segment, how many steps of rotation that I have to perform in my coding scheme that is in this scheme I will simply put that in my chain code. So, while doing this, let me start this way initially I will skip the first code that I have generated, because though I know that the first code was 0, but I do not know that from previous code to this code, when I move how many steps of rotation I have to make, because I do not know what is the previous code.

So, here, but I know that this direction is 0, next direction is also 0, so I know that from to move from 0 to 0 how many steps of rotation I have to make? So, in this case, because the previous direction is also 0, the next direction is also 0, so the number of steps of rotation will be 0, so I will put this as 0 only. The next one from 0 to 7, here I have to find out how many steps of rotation I have to perform? So, this is 0 and this is 7 and if I compute the steps

of rotation in the anti-clockwise direction, then you find that I have to make 1 2 3 4 5 6 7, 7 rotations, so this will also be 7.

To generate the next code, I know this was 7, the next one is 6, so how many steps of rotation I have to find, I have to compute? So, this is 7, this is 6, so again I have to compute 7 steps of rotation, so this will also be coded as 7. Then from 6 to 0, this is 6 this is 0, I am following anti clockwise rotation, so I have to perform two steps of rotation, so 1 and 2, so this will be coded as 2. Next, is from 0 to 6 I have to perform 6 steps of rotation, next is from 6 to 5 I have to perform 7 steps of rotation, next is from 5 to 6 I have to perform only 1 step of rotation, next is 6 to 6. So, the number of steps of rotation is 0, 6 to 6 again number of steps of rotation is again 0, then 6 to 4, so this is 6 and this is 4 which is in the west direction.

So, here you have to you find that 1 2 3 4 5 and 6, I have to perform 6 steps of rotation, that is why I put 6 over here. Then 4 to 4 again 0 steps of rotation, 4 to 4, 0 steps of rotation 4 to 4, 0 steps of rotation, then 4 to 3, 4 this is 3, so if you compute in the anti-clockwise direction, you find that I have to perform 7 steps of rotation. Then 3 to 2 again I have to perform 7 steps of rotation, then 2 to 3 following anti clockwise direction I have to perform only 1 step of rotation, then 3 to 2 again 7 steps of rotation, 2 to 1 again 7 steps of rotation, 1 to 0 again 7 steps of rotation, 0 to 2, I have to perform 2 steps of rotation, then 2 to 1, I have to perform 7 steps of rotation.

Now, I find that is 1 to 0, because the last code was 1 and the first code was 0, so when I move from this segment to this segment, from here to here rotation is 1 to 0. So, this is 1, this is 0 number of steps of rotation that I have to perform is 7. So, this completes my differential chain code. Now, again this differential chain code as I have formed, this is still dependent on the starting position. So, instead of taking the starting position as this point if I take the starting position as this point, then my chain code will start from this location. So, instead of 7 0 7 7 2 6 7 1 0 0 6 0 0 0 7 7 1 7 7 7 2 7 it will be 7 7 2 6 7 1 0 0 and this will continue and it will end with 7 0.

So, though I have, though I have made this code as independent of rotation, but still it is dependent on the starting position, the starting pixel that we considered to form this chain code. So, to make it starting position independent, what I have to do is, I have to consider this as a cycle, so it is not an open chain, but it is a cycle. So, if I break the cycle at any point what I make is a chain, so from this cycle I have to come to a chain and when I come to a chain, what I have to do is, I have to identify a point within a cycle where if I break I will get either a maximum number or a minimum number.

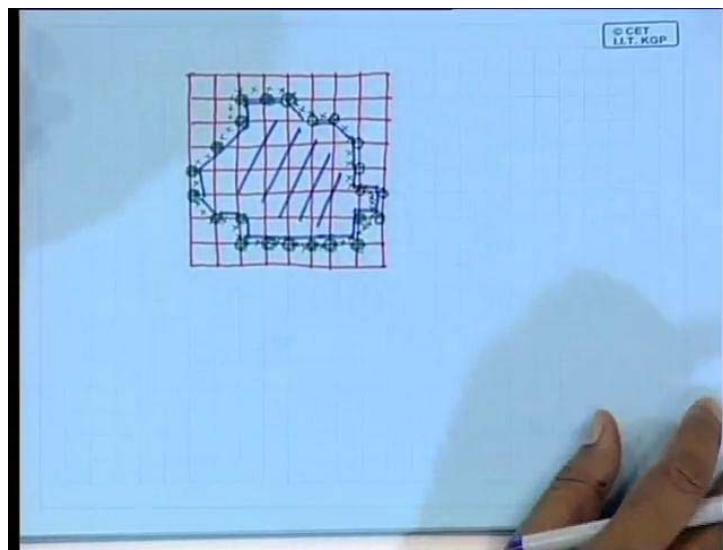
So, if I follow I will always break it in such a way that I will always get a maximum numerical value, I get unique code. If I follow the other way that wherever I break, I will get the minimum numerical value, I will also get a unique chain code. So, if I follow the concept that I will break the chain code in such a way that I will get the minimum numerical value, then the number that I get is what is called a shape number. So, is the concept of differential chain code clear now? So, from the differential chain code what we initially form is a cycle not a chain and once I have this cycle, I decide a point where the cycle can be broken, so that I can get a chain code having either maximum numerical value or minimum numerical value.

So, if I follow the concept of maximum numerical value, then always I have to form the maximum numerical value, if I follow the concept of minimum numerical value, always I have to form the minimum numerical value. Both of them will serve our purpose in terms of recognition, whether we use the maximum numerical value or minimum numerical value. Is it okay? Now, the next point comes that can we make it scale independent, we have made it rotation invariant, rotation independent by taking the differential chain code, and can we make it scale independent as well? Obviously the chain code is translational invariant, because the shape is here or the shape is here I will get the same code, so it is independent of translation.

By taking the differential chain code, we have made it independent of rotation. Now, whether can we, whether we can make it independent of scaling as well? That means, if the size of the boundary changes, if it becomes more or less, whether or this uniqueness of the chain code still can be retained. So, let us see that how we can make this chain code scale independent. So, to do this, let us assume that we have a set of boundary points something like this, so initially the set of boundary points that we have, the

distance between two consecutive boundary points is just 1 pixel, they are at a distance of 1 pixel apart.

(Refer Slide Time: 14:30)



So, if I have a boundary something like this, let us take an arbitrary boundary. So, assume that these boundary pixels are 1 pixel apart, so all these crosses are my boundary pixels. So, these points are nothing but only regularly spaced grid, where the grid spacing is just 1 pixel. So, what I will do is, I will embed this set of boundary pixels on another grid, where the grid's spacing is much more, much more than 1 pixel. So, if I put the grid spacing like this, so these are the grids of larger spacing in which I embed this boundary pixel. Now, once I have this grid, then what I will do is; these boundary pixels will be associated with one of the grid crossings of this larger spacing, whichever is nearest to it.

So, if I do that you find that this point will be associated with this, this point will also be associated with this, and this point will also be associated with this. So, the grid crossing that I will have is this one. Similarly, this is another grid's crossing, this is another grid's crossing to which some of the pixels will be associated, this will be another grid crossing, this will be another grid crossing, this will be another grid crossing, maybe this will be another grid crossing, this one will be another one, this one will be another, this will be another one, this, this, this. So, these are the grid points of this larger grid in which the points will be associated.

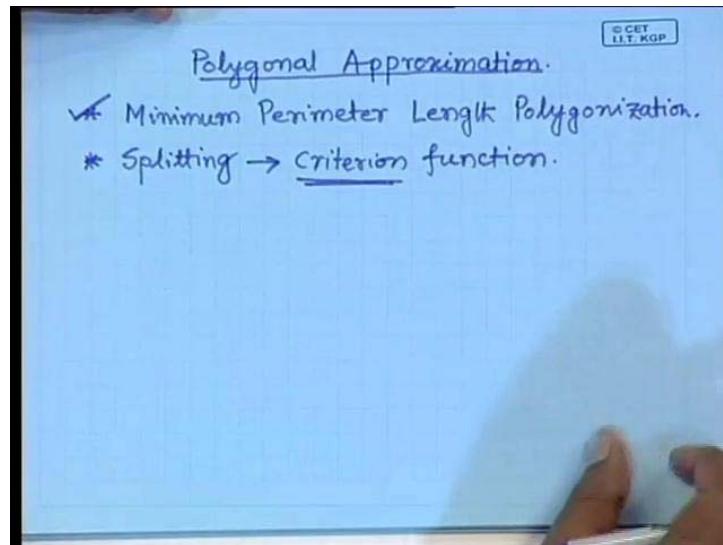
Now, I can form up approximate boundary of the boundary represented by my initial set of boundary pixels by these grid points, so my boundary now will be this. So, you find that

initially I had large number of pixel boundary, boundary pixels which are now scaled down to a lower number of boundary pixels and what I get is an approximate representation of the original boundary. So, now if I use this approximate boundary to form the chain code, then to some extent I take care of scaling, because as if I have represented the same boundary by less number of line segments.

So, like this if I increase the spacing of these grids, the number of line segments in my chain code will still be, will still be lesser, if I reduce the grid spacing, in that case number of line segments in my chain code will be larger. Of course, maximum that I can have is my initial set of boundary points, unless I go for super resolution concept, where I can interpolate a point between two boundary points. So, that is a concept of super resolution where I can still increase or improve the resolution of the given image, is that ok?

So, these are the different forms of chain code that we can form and this chain code can also be used for recognition of objects, where this recognition is based on shape, because this chain code is nothing but a shape descriptor, where this shape is obtained using only the boundary information. Because we have not used any point which is within the shape, I have not used any information which is inside the shape, I have used only the information of the boundary. So, this chain code can be used for recognition of the objects where this recognition will simply be based on the boundary information. Now, the other type of descriptor that can be used for recognition purpose is based on polygonal approximation.

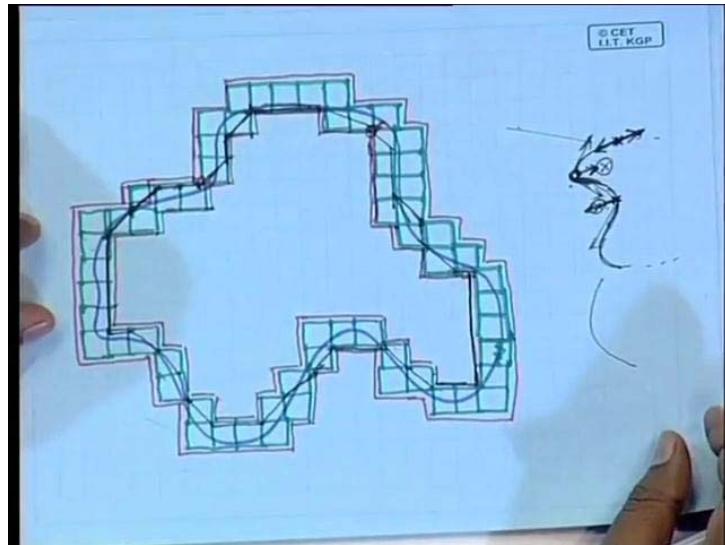
(Refer Slide Time: 21:03)



So, chain code also you can say that it is some sort of polygonal approximation, because the arbitrary boundary is represented by linear line segments, piece wise linear line segments, so it is also nothing but polygonal approximation. But when we talk about polygonal approximation, what we do is we do something else to represent an arbitrary boundary by a polygon, so how do we do that? When we go for polygonal approximation there are different types of approaches which can be used to represent a boundary by a polygon. So, out of this we will discuss here two techniques, one of the techniques is called minimum perimeter length, polygonization.

And the other kind of polygonal approximation technique is based on splitting, splitting of the boundary. Now, when I go for splitting, obviously there has to be different types of criteria based on which an arbitrary boundary has to be split into number of boundary segments, where every boundary segment will be represented by a linear segment or a straight line segment. So, when we talk about splitting, normally we have a criteria function associated with the splitting technique and based on the criteria function I can have different types of splitting algorithms. So, we will discuss them one after another, so firstly let me take this minimum perimeter length polygonization approach, what I mean by that?

(Refer Slide Time: 23:42)



So, to do this let me again take a boundary of arbitrary shape, say I have a boundary something like this or let me make it a bit more complicated. Say, I have a boundary

something like this, so to go for this minimum perimeter length polygonization, what we will do is, we will embed this boundary into a set of concatenated cells. So, what are those set of concatenated cells? Say, the cells are, I can use this set of concatenated cells, again it is nothing but a concept of grid. So, this is the set of concatenated cells in which this boundary will be placed and this is nothing but if I put the same concept of grids over here and then I have to identify that which of the cells, actually enclose this particular boundary.

So, once I have this set of concatenated cells, and then you find that I can define something like an inner boundary and outer boundary, this cell will also be there. So, my inner boundary will be this one, this is the inner boundary and this one is the outer boundary or inner wall and outer wall. So, this cell will also be included, because so the outer wall will be like this. So, I have an inner wall and outer wall. Now, if I assume this boundary to be a rubber cord, a closed rubber cord which is kept within this inner wall and outer wall.

Then you know that a rubber cord under tension which is kept under a constant place like this, it always try to attain the minimum length. So, as a result this cord will try to shrink in whichever ways possible and it will try to attain the minimum perimeter. So, you find that this part of the cord will try to be contracted and come towards this direction; similarly, this part of the cord will try to be contracted and move towards this direction. So, likewise if you find out at each and every point on this boundary, how the cord will move, whether it will move in the outer direction or it will move in the inner direction, until and unless it reaches the limiting points.

So, when it tries to move in the inner direction at this point the limiting point is this, it cannot move further because this is the wall. Similarly, over here when this point tries to move outer, in the outer direction this is the limiting point, it cannot move beyond this. So, if you continue this way you will find that I will have a boundary representation of this particular boundary which is given, which will be given approximate representation something like this. So, this is my approximate representation of this boundary when this cord tries to shrink, because it will try to move both in the outward direction, as well as in the inward direction wherever is possible, until and unless the cord reaches the limiting point. So, what I get is a polygonal representation of this boundary.

Now, so far as the concept is concerned it is very, very clear, but how do we implement it? For implementation, what you have to do is you find that at every point, either the cord is

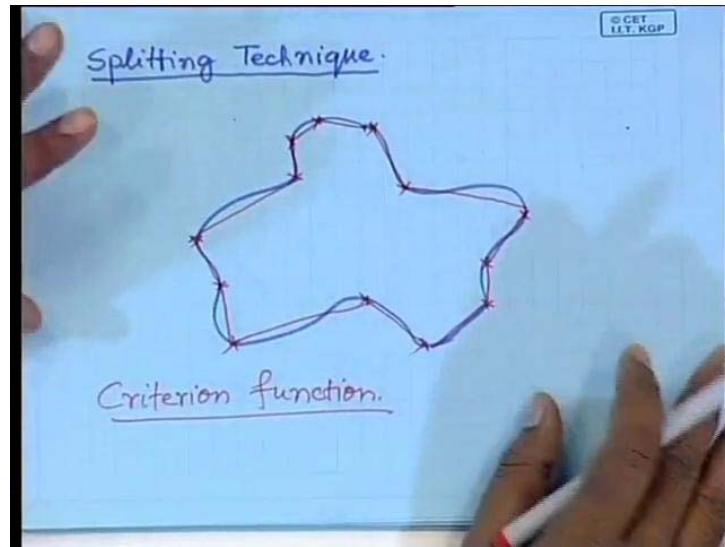
straight or the cord is concaved or the cord is convex. So, if I have a boundary position something like this, where there is continuing in this direction, over here the cord is convex, over here the cord is concave, over here the cord is convex and I can have a linear portion something like this.

So, if I consider any point on this cord segment, if the point lies on the linear part, then at this point I will have tension in this direction, as well as in this direction, in two opposite directions. That means, if this point has to move, it has to move only along the cord, because I have tension either in this direction or in this direction, in two opposite directions.

So, if I assume that the tension in both the directions are same, this point will not move at all, however this point will move due to the influence of the movement of this point. Here it is in the convex part and I have tension in this direction and I have tension in this direction. So, as a result the resulting tension which is acting at this point is in this particular direction. So, this point will try to move in this direction and here I have to find out that what is the limiting point, limiting points are given by these inner walls and outer walls. So, I have to identify that what are the limiting points and I have to allow this point to move until and unless it reaches a limiting points.

Similarly, a point at this location it has tension in this direction and it also has tension in this direction, so the resulting tension is in the direction like this. So, under the influence of this resulting force, this point will also try to move in this direction and it will move until and unless it reaches a limiting point somewhere over here. So, when this point comes up to this position I have to stop movement of this point. So, that is how I can find out that what are the limiting points up to which this cord can move, while it tries to attain the minimum perimeter and those limiting points in the inner wall or in the outer wall gives you the vertices of the polygon. So, if you represent, if you simply connect those vertices by straight line segments you get a polygonal approximation of this arbitrary boundary. So, this is what minimum perimeter length polygonization. Any question? No. So, next let us see that, what is the other type of polygonization, that is splitting technique.

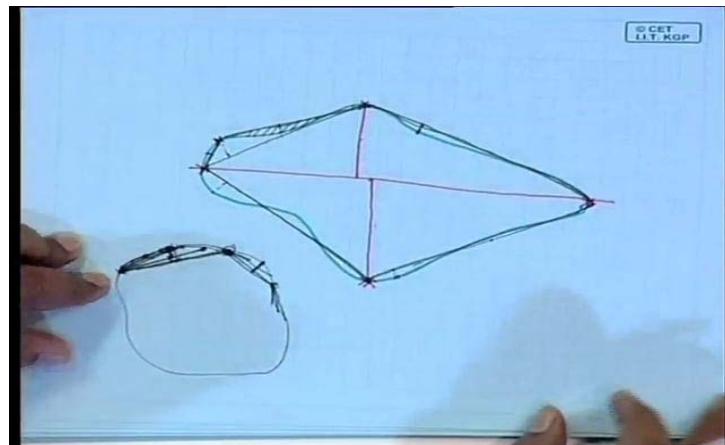
(Refer Slide Time: 36:39)



In splitting technique again if I have an arbitrary boundary something like this, what I have to identify is, I have to identify a set of points on this arbitrary boundary along which this boundary can be split. And these points at which the boundary can be split into different boundary segments they become the vertices of the polygon. So, if somehow I select this set of points along which this boundary can be split then the polygonal approximation of this arbitrary boundary is this one. Now, the criteria function as I said comes into picture which helps to decide that what are the points on this arbitrary shaped boundary are to be taken along which this boundary can be split.

So, there are different criteria functions, the different functions give you different result. The results may not be identical, but they are more or less similar and help in all practical applications of this polygonization technique. So, let us talk about, say one or two such criteria functions based approach.

(Refer Slide Time: 38:50)



So, let us take a boundary something like this, so one of the criteria function says that initially you decide two points on the boundary along which this boundary can be split. So, once I decide two points on the boundary then the boundary splitted into two sub boundaries or two segments of the boundary. So, first let us assume that somehow we have been able to select one point over here and one point over here. So, these are the two points along which the boundary can be split, so once I select these two points, then I have one boundary segment which is the upper part of the boundary and I have another part of the segment which is the lower part of the boundary.

Then the criteria function says that once I select these two points, I can have a straight line passing through these two points, so this straight line divides the boundary into two halves. Now, for each of the halves what I do is, I find out the maximum distance, perpendicular distance of a boundary points from this straight line. So, on this half you may find that this will be a point where the perpendicular distance from the straight line joining these two split points is maximum. Similarly, on this half this may be a point from where the perpendicular distance to this straight line may be maximum, right? Then the criteria function says that if this maximum distance is greater than its threshold, then this point I have to take as the next splitting point.

So, suppose in this case both these distances are greater than threshold, so this will be one split point, this will be another split point, but I have to take it one by one, because I have to write a algorithms. So, I take the maximum distance first, so I assume that this is my next split point. So, once I select the next split point, in that case I have divided this boundary in three boundary segments. So, once I have these three split points, again I can have straight lines passing through pair wise split points, so this is one straight line and this is another straight line.

Again I follow the same concept, same criteria that for each of these straight lines I find out a point on the boundary between these two split points whose distance from this straight line segment is maximum. So, possibly I will get a point somewhere over here, whose distance is maximum. And now, suppose this distance is less than the threshold that is predefined, so if this distance, because this is maximum, so all other points on this boundary will have a distance from this line segment which is less than that. So, if this maximum distance is less than threshold, then all other points on this boundary segment is within a distance which is less than the threshold. So, because it is less than the threshold I do not have to try to split this boundary anymore.

Similarly, over here I find out the maximum distance, if the maximum distance is less than the threshold, I do not have to split this boundary anymore. However, for this boundary segment, this was my distance and suppose, this distance is greater than the threshold, so this becomes my fourth splitting point. Once I have this fourth splitting point, again I join the straight line segments, the straight lines to this splitting point. Now, this is the maximum distance of this particular segment from this straight line and this is the maximum distance of this segment from this straight line.

Suppose, this distance is less than the threshold, so this part of the segment is not to be splitted anymore. And if I assume that this is greater than the threshold, then this will be my fifth splitting point. So, once I have this fifth splitting point, again I will have line joining these vertices, over here this is the maximum distance and over here this is the maximum distance. Now, if these distances are less than the threshold, then neither I have to split this nor I have to split this. So, my final polygon now becomes this one, so this is the polygonal representation of this boundary that I have obtained. Is that okay?

Similarly, there are many other criteria functions which can be used for polygonal approximation. One of the criteria functions tries to find out, what is the area of the portion which is enclosed between the line, this straight line approximation and the original boundary and that criteria function tries to minimize this area.

The other criteria function can be something like this that given a boundary segment of this form, I start with a, from an initial point, then go on considering the points across the boundary and try to approximate them by straight lines like this. Every time I approximate them by straight line, again I have a concept of threshold that is I find out the maximum distance of a boundary point from this particular point.

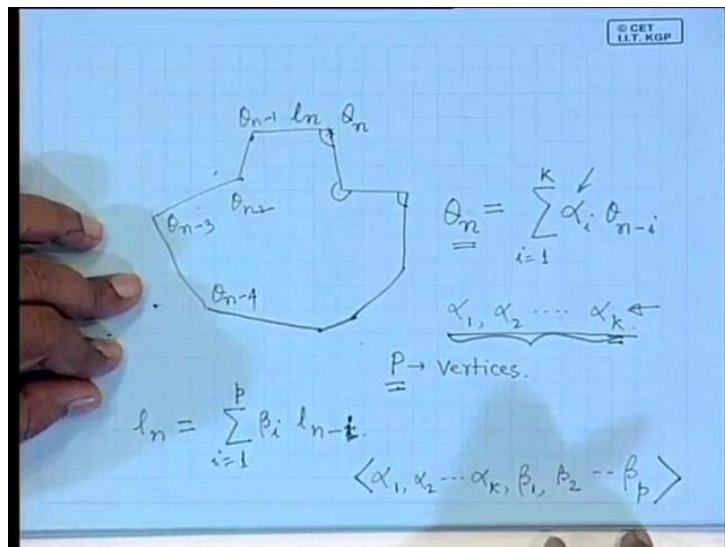
When it exceeds the threshold then my starting point becomes this, so suppose I started from here, initially I have considered this straight line to approximate this boundary segment, here the distance was less than the threshold, so this is not my vertex, I go to the next point. Again find out what is the maximum distance between this line segment and a point on the boundary. Suppose, this is also less than the threshold, I consider the next point, join, try to approximate this boundary segment by this straight line. Now, suppose this maximum distance is greater than the threshold or has been equal to the threshold.

So, the moment I get that I decide that this line segment cannot be expanded further. So, this part of the boundary, this line segment is represented by this line segment. Now, I start the

same operation taking this as my starting point, so I have decided about one-line segment. Now, I perform the same operation over here and suppose after I come to this location, this distance becomes greater than or equal to the threshold.

So, this becomes my next line segment. Then I start the same operation starting from this point, it continues like this. So, there are different criteria functions which can be used for polygonal approximation of an arbitrary boundary. So, following any of them what I get is, given an arbitrary boundary I get a polygonal approximation of the arbitrary boundary.

(Refer Slide Time: 47:03)



So, suppose I have a polygon something like this, which represents arbitrary boundary. Now, this polygon by itself is not much helpful for recognition purpose, so what I have is, I have an arbitrary boundary and I have been able to approximate that arbitrary boundary by linear segments, by piece wise linear segments and the combination or the collection of those linear piece wise linear segments give me a polygonal approximation of the boundary. But for recognition purpose, finally I have to find out the feature vector or a set of features, using which I have to go for recognition.

So, how do you generate that set of feature vectors, the set of features or the feature vector from here? So, the one approach to generate the feature vector is something like this, once I have this polygonal approximation, I can always find out what is the angle, inner angle subtended at each of the vertices of the polygon. So, suppose I take the n^{th} the angle

subtended at the nth vertex to be θ_n . So, once I decide this, I can have an auto regressive model, so that this n^{th} angle is represented by a linear combination of k number of previous angles or effectively I will have a linear regression equation which is given by this,

$$\theta_n = \sum_{i=1}^k \alpha_i \cdot \theta_{n-i}, \text{ where this } i \text{ will vary from 1 to some } k.$$

So, what I am doing? This angle is represented by some $\alpha_1 \cdot \theta_{n-1} + \alpha_2 \cdot \theta_{n-2} + \alpha_3 \cdot \theta_{n-3} + \alpha_4 \cdot \theta_{n-4}$ and so on. So, if I take k number of such previous angles to form this auto regressive model, this is what k^{th} order auto regression. Is that okay? So, for every such vertex I will have one such equation and the set of α these are the coefficients of auto regression. So, because I have k^{th} order auto regression, so there will be k number of α , α_1 , α_2 up to α_k and if there are say P number of vertices. If this polygon contains P number of vertices, then for each of the vertex I will have one such equation. So, I will have P number of linear equations like this.

So, now the problem becomes, that I have to find out a solution of those P number of simultaneous linear equations. Now, if P is exactly same as q, exactly same as k, then this model is well defined, because I know there are k numbers of variables α_1 to α_k and I have k number of equations, so I can get a unique solution. Whereas, if $P > k$ I will have number equations which is less than the number of variables, so the system is over specified. If $P < k$, that means the number of equations is less than number of variables, then I cannot find unique values of this set of coefficient α_1 to α_k .

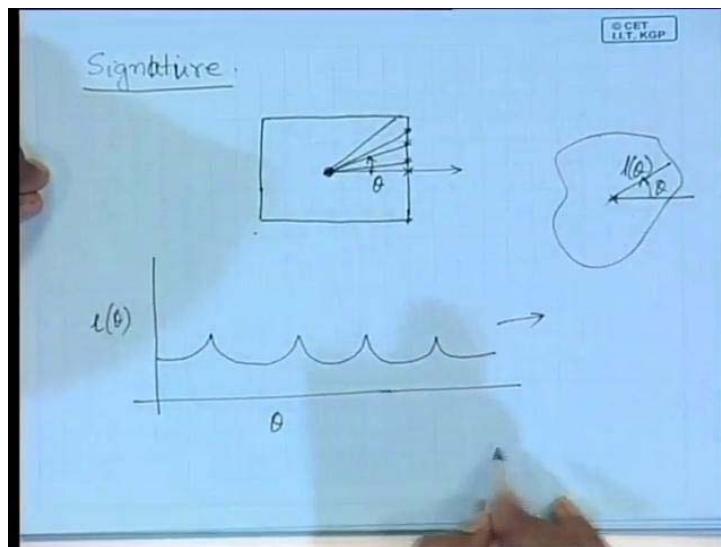
So, this model, order of the model, auto regressive model, I have to decide suitably depending upon how many vertices I have in the polygon. If value of $P > k$ as I said that the model is over specified, because I have more number of equations than the number of variables. So, in such case we can go for a solution approach which is mean square error solution, that is I have to find out the set of Alphas which gives minimum error.

So, those solution approaches we will talk later on in respect when I discuss about other linear classifiers and all that. But right now what is important to us that, we have given such an auto regressive model, we can find out a set of features α_1 to α_k and this set of α_1 to α_k can be used as feature vector, representing this particular polygon. Now, again you can find that, I have made use of only the angle I have not made use of the side lengths or the lengths of the edges. So, even if I elongate this polygon either horizontally or vertically or diagonally, I will get the same set of coefficients. So, that can be taken care of, that I can have an another

auto regressive model, along with this say if I consider the side length l_n and this l_n can also be represented can also be modelled by an auto regressive model something like this, $l_n = \sum_{i=1}^k \beta_i l_{n-i}$, where this i may vary from 1 to say some value P , so P^{th} order auto regressive model.

So, I get a set of coefficients α_1 to α_k considering the angle subtended at the vertices and I can also get another set of coefficients, P number of coefficients β_i, i varying from 1 to P considering the lengths of the edges. Now, if I concatenate these two, $\langle \alpha_1, \alpha_2, \dots, \alpha_k, \beta_1, \beta_2, \dots, \beta_p \rangle$, so I get a feature vector of dimension $k + P$ and this feature vector can be used for classification purpose or for recognition purpose. So, here we take care of both the angle values at the vertices and also the lengths of the edges of the polygon. So, that will take care of, if there is any deformation either the polygon is elongated in certain direction or the polygon is compressed from some other direction.

(Refer Slide Time: 55:18)



So, these are the feature vectors that can be generated based on polygonization approach, there are other approaches as well, one is to find out the signature. I just, I forgot to mention one thing that, here as I said that initially somehow I have decided these two points, but how to decide these two points, because selection of these initial points will determine, what is the accuracy of or how unique this polygonal approximation is? So, one way to decide this initial

split points is that, I think all of you have done K-L transformation, so in k l transformation it gives you the Eigen directions. So, I can decide the points along the first Eigen direction, because first Eigen direction gives you the direction along which your shape is maximally elongated.

So, I can take these points, these two points along the first Eigen direction, set of points between which your distance is maximum. Is that okay? And starting from there I can continue this algorithm iteratively to give me the polygonal approximation. Now, what is signature? The signature is something like this, that let us assume that we have a square or a rectangle, something like this. I am taking a simple example this is true for any type of arbitrary shapes and within this I have a reference direction, say this is my reference direction. What I do is, starting from the reference direction, I find out the distance of all the boundary points from the centroid of this particular shape.

That is, I find out the distance of the boundary point, over here, I find out the distance of the boundary point over here, I find out the distance of the boundary point over here and so on. That means, I am taking the boundary point along the rotational scan lines something like this, in different directions and this direction is at an angle θ from the reference line. Then what I do is, I plot this distance say $l(\theta)$ versus θ . So, if it is a square or a rectangle something like this, then you find that this plot will appear to be something like this, where these peaks correspond to the corners or the vertices of the square or rectangle.

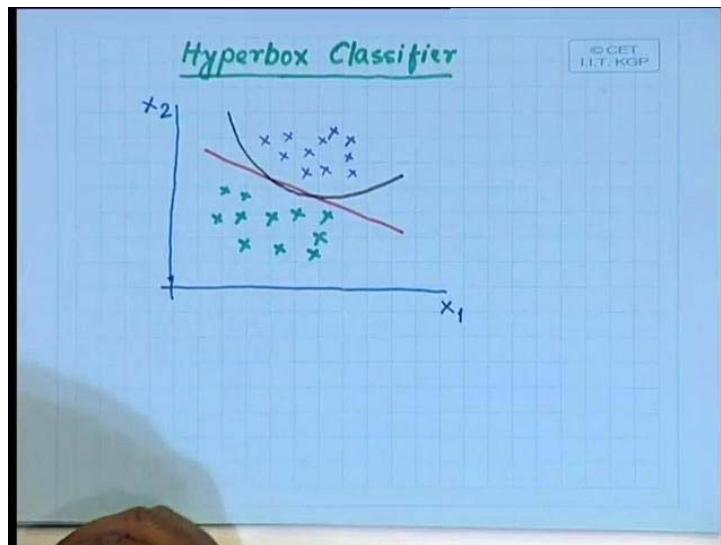
Now, this particular pattern itself can work as a feature or a feature vector or I can transform this pattern to get some transformation coefficients and those transformation coefficients can work as feature vector, which can be used for recognition purpose. So, find that if I have an arbitrary shape something like this, starting from its centroid and with respect to some reference. If I go, if I go on computing the distances, $l(\theta)$ at an angle θ , then the plot of $l(\theta)$ versus θ will give you some signature of this particular shape. The signature itself or a transformed version of it can be used as a feature for recognition purpose. So, let us stop here today.

Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 30
Hyperbox Classifier

Good morning. So, today what we are going to discuss about is the hyperbox classifier. Earlier, we had just introduced the problem in the sense that so far the kind of classifiers we have talked about it separates between 2 classes or the regions belonging to 2 classes, which are simply connected.

(Refer Slide Time: 00:41)



In the sense that I have a set of feature vectors belonging to two classes, which are given in this form. Again I am taking this example in 2 dimension having two feature components x_1 and x_2 . And if the feature vectors or the training vectors belonging to two classes say ω_1 and ω_2 , they form clusters like this, so this is one cluster. This is another cluster. Where one of the clusters is taken from say class ω_1 . The other cluster is formed by the points taken from another cluster say ω_2 .

Then the regions belonging to these two different classes; ω_1 and ω_2 , they are to be simply connected. I should be able to have a simple boundary, either a linear boundary or a non-linear boundary separating these two classes.

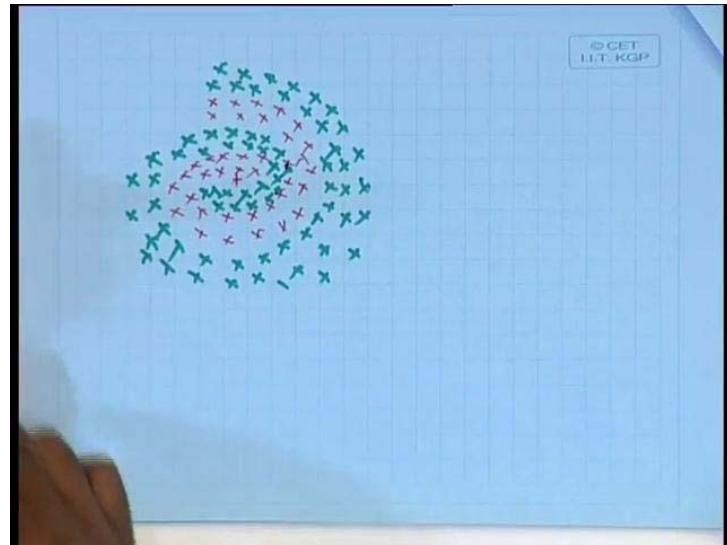
So, either I can have a boundary, linear boundary, something like this in which case the classes will be called linearly separable or I should be able to have a non-linear boundary something like this in which case the classes are linearly non separable. So, I should be able to have either a linear boundary or a non-linear boundary. So, accordingly we have talked about different types of classifiers starting from our Bayes classifier, whether it is Bayes minimum base classifier or Bayes minimum error classifier and so on. We have seen that in a particular case, when all the classes have the same covariance matrix or the points taken from all the classes, they have the same covariance matrix. At the same time, the feature components are statistically independent or different feature components are statistically independent.

Then, we can have a linear boundary separating the two classes, ω_1 and ω_2 . But in general, if different classes have arbitrary covariance matrices or the points belonging to different classes of arbitrary covariance matrices in that case, the classes are not linearly separable. So, you can have a nonlinear boundary. And mostly, what we have is a quadratic boundary. And then we have discussed about other approaches for designing of linear classifier like perceptron criteria or relaxation criteria, based approach, minimum square error based criteria based design approach.

In all these cases, we have talked about how we can design a linear classifier that separates two different regions by a linear boundary. And lastly, what we have talked about is the neural network approach. If I have a single layer perceptron or a single layer neural network having multiple number of outputs, then what we have is we separate the classes where the classes are separated with linear boundaries.

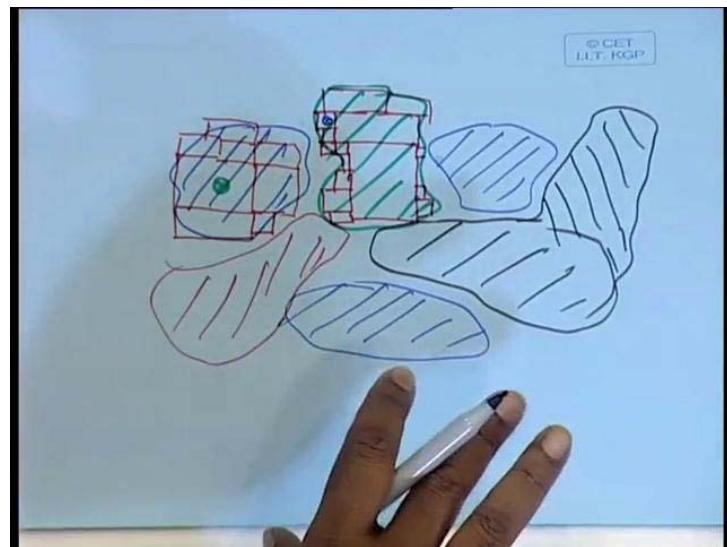
If the classes are not separable by linear boundaries in that case, we have to go for multilayer neural network or multilayer perceptron. In which case the non-linear boundaries are actually replaced by piecewise linear boundaries. So, this multilayer neural network what it does? It approximates a linear boundary by piece wise linear boundaries. So, in all these cases, the regions belonging to different classes, they have to be simply connected. But as we said earlier, we have a point of distribution belonging to different classes something like this.

(Refer Slide Time: 04:48)



So, these are the distribution points belonging to two different classes where you can assume that all the red points belong to one class, whereas the green points belong to another class. Not only that, I can have some arbitrary distribution something like this as well.

(Refer Slide Time: 06:34)



Say I have a set of points somewhere over here, which belong to one class. I can have set of points somewhere over here that belongs to the same class. I can have another set of points somewhere over here, which also belong to the same class. And in between, I can have,

so this is the set of points which belong to one class. This may be the set of points that belong to some other class. This may be the set of points, which belong to some other class or this set of points may even be extended like this.

So, if the pointer distribution is a complicated point distribution something like this, you find that neither linear boundary, obviously a linear boundary cannot separate among these classes. Even a nonlinear boundary is also unable to separate between different classes. So, how we can design a classifier that can even classify the point distributions or the class distributions, which are given like this? So, I just pointed out during our introductory lectures that we can go for a set of hyper-boxes. That is what I do is you divide this space into a number of hyper boxes something like this. Where different hyper boxes may be having different dimensions. Similarly, over here.

So in this two dimensional example, I can assume that those hyper boxes are represented by rectangles. So, I have a set of rectangles, which represent 1 particular class. I have another set of rectangles, which belong to another particular class. And if somehow we can assign the class label to this different rectangles, so I can say that this are the rectangles, which represent class say ω_1 . These are the rectangles, which represent class ω_2 . Similarly, I can have a set of rectangles over here that will represent class ω_3 . I will have a set of rectangles covering this region, which will belong to class ω_4 and so on.

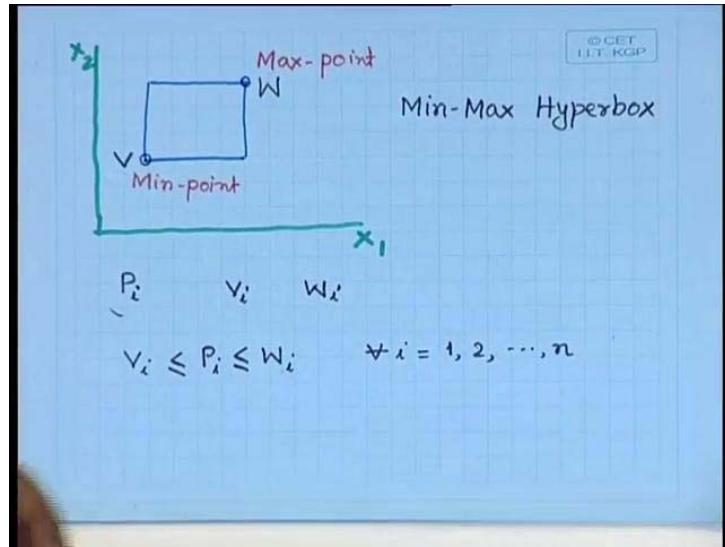
So, during the training process during the training of the classifier, we have to identify these set of rectangles or the set of hyper boxes. So, next time during the testing or when you actually go for classification of unknown feature vectors, I simply see that what is the position of the unknown feature vector with respect to these hyper boxes, which are already generated. So, if the unknown feature vector falls somewhere over here, so this is the location of the unknown feature vector. Then I can immediately say that because this unknown feature vector is lying within a rectangle, which is labelled as class ω_1 .

So, this unknown feature vector also belongs to class ω_1 whereas if the unknown feature vector falls somewhere over here. Then immediately I can say because of this unknown feature vector falls within a rectangle, which is labelled as class ω_2 , so this unknown feature vector also belongs to class ω_2 . So, that way, I can define or I can generate a set of hyper boxes based on the training samples, which I have given.

And those hyper boxes are given the class labels based on from which training classes belonging to which class that particular hyper box has been generated. So, that becomes the

class label at a particular hyper box. Once I get the class of the hyper boxes than while testing or while in actual operation, I just find out that in which of these hyper boxes that unknown feature vector is situated. So, this unknown feature vector gets the same class label as that of the hyper box. So, these hyper boxes are to be generated during the training operation of the classifier. Now, suppose I have a situation in something like this.

(Refer Slide Time: 11:47)



Say I have one hyper box somewhere over here. Now, what is the advantage of these hyper boxes or the advantage of the rectangles? Once I divide the space, the feature space into a number of hyper boxes or into a number of rectangles, you find that such rectangles or such hyper boxes can be defined using just two points. If I know what is the left bottom point and if I know what is the right top point that is sufficient to define the hyper box. So, one of the points, this point left, because if I plot it against our conventional two dimensional coordinate system, say this is my X_1 feature. This is the X_2 feature.

You find that corresponding to this left bottom point, both of the features X_1 and X_2 they are minimum. Similarly, corresponding to this top right point, both X_1 and X_2 feature values are maximum. So, accordingly this point is called a min point and this point is called a max point. So, once I define this rectangle or the hyper box using this min point and max point, such hyper boxes are called min max hyper boxes. And to see whether an unknown point is within this min max hyper box or not suppose this min point is represented by vector V , which will have X_1 coordinate and X_2 coordinate, which are minimum among all the points.

It is also represented by max point, which is W, which also has X_1 coordinate and X_2 coordinate which are maximum among all the points within this hyper box. Now, so similarly, if I have n dimensional feature space or the feature vectors are n dimensional feature vectors, then all the n coordinates for this min point will be minimum among all the points. Similarly, all the coordinates for the max point, all the n coordinates for the max point will be maximum among all the feature vectors, which are given. So, if I say that a point P, I want to check point P belongs to this hyper box or not.

So, let us consider the i^{th} dimension say, I have this unknown point P and I want to check whether this point P belongs to this hyper box or it does not belong to this hyper box. Suppose let me take that P_i to represent that i^{th} dimension of this feature vector P. It is i^{th} dimension, 1 out of the 10. Similarly, for the min point say, V_i represents the i^{th} min point that means the minimum value of the i^{th} coordinate. Similarly, W_i represents the maximum value of the i^{th} coordinate. So, this is i^{th} dimension, i^{th} component of the max point

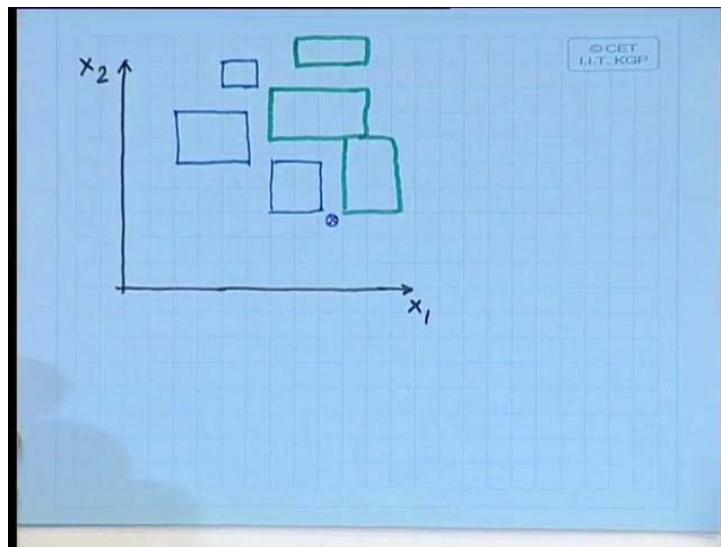
So, if this unknown feature vector P has to lie within this min max hyper box, then obviously I have to have the condition that $V_i \leq P_i \leq W_i$. and this has to be true for all i, where in from 1, 2 up to n. So, for all the components, all values of i, P_i has to lie in between V_i and W_i . So, if it is so, then we can say that this unknown feature vector P lies within this hyper box, or min max hyper box given by the min point V and the max point W.

So, this definition itself tells you that how to generate a min max hyper box that is given a set of points. I want to find out a min max hyper box, which encloses all the points within that set. So, for doing that, what I have to do is I have to find out every component, the minimum value of every component among all the feature vectors, which are there in the set. So, if I have 50 points within the set for each of these feature vectors, I have to find out the first component. So, I will have 50 first components. Minimum of that will give me the first coordinate of the min point.

Similarly, coming to the second dimension, I will get 50 components coming from 50 different feature vectors. I have to find out minimum of that. That gives me the second coordinate of the min point. Similarly, the maximum values will give me the corresponding coordinates of the max point. So, given a set of points, for every component, the minimum values will give me the min point. The maximum values will give me the max point. You find that that follows from this particular condition that if P lies within the min max hyper box.

Then, every component of P has to be within the minimum and maximum values of the corresponding component. So, that it is within min point and max point, or min component and max component of the corresponding hyper box. So, by checking this condition, I can easily find out with a point or an unknown point belongs to a given hyper box or not.

(Refer Slide Time: 18:54)



Now, I come to a situation that I suppose to have a situation something like this. This is my X_1 dimension and this is my X_2 dimension. The hyper boxes are generated from the training examples. And as the hyper boxes are generated from the 10 examples, it is not necessary that set of hyper boxes that you generate that will cover the entire space. It is quite possible that a part of the future space will not be covered by any of the hyper boxes because it depends upon the distribution of the training samples.

So, suppose I have a situation something like this that I have a hyper box somewhere over here, which belongs to class ω_1 . I have another hyper box somewhere over here, which also belongs to class ω_1 . I can have another hyper box somewhere over here, which also belongs to class ω_1 . I can have a hyper box over here, which may belong to class ω_2 . I may have a hyper box over here, which also belongs to class ω_2 . I may have a hyper box over here, which also belongs to class ω_2 .

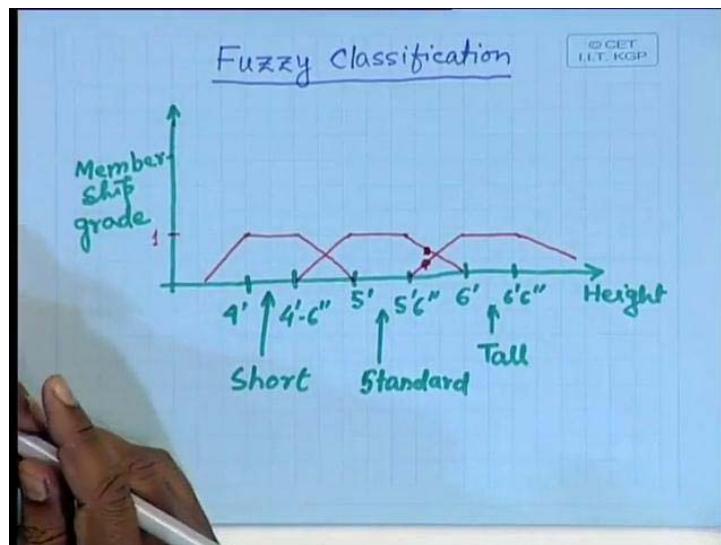
So, you find that given these hyper boxes, there is a lot of space in between the hyper boxes, which is not covered by any of the hyper box. So, if a feature vector unknown feature vector because these hyper boxes are created using the training samples. So, it is obvious that all the training samples, which have given, they are covered at least by one of the hyper box. But

given an unknown sample, if an unknown sample falls somewhere over here, say this is the position of the hyper unknown sample. This unknown sample does not belong to any of the hyper box. So, should we say that this unknown sample does not belong to any of the classes?

So, it is quite logical that if I can have some measure by which I can classify this unknown sample as well, so one of the approach is what we discussed about the minimum distance classifier. That is, you try to find out what is the distance of this unknown sample from defined hyper boxes, so by calculating the distance, you find out that which hyper box is nearest to this unknown sample.

And I can classify this unknown sample to the nearest hyper box or the class of the nearest hyper box. That is one of the way, where we go for minimum distance classification sort of thing combined with this hyper box classifier, the other approach is which appears to be more logical. This is because that is what in is in line with how we think or how we classify without actual measurement. So, that is what is called fuzzy membership or fuzzy classification.

(Refer Slide Time: 22:34)



I will not go into the details of fuzzy theory, but I will just introduce what is the concept of fuzziness. How this fuzziness can be combined along with our hyper box concept? So, that we can have a definition of fuzzy hyper box or fuzzy hyper box set. So, the concept of fuzziness is something like this. Let us take a simple example that given a set of persons, a large number of persons, we want to say that who are the persons who are tall, who are the persons who are short and who are the persons who are say medium height.

Now, while doing this, we never take a tape. Find out what is the height of a person. Accordingly, we classify whether the person is tall or the person is short or we also say that the person is very tall. We say that the person is tall, but not so tall. Similarly, short, but not so short. So, when we decide something or when we classify something, we do not really classify in the hard manner. It is not that if the height of the person is above 6 foot, 6 inches, he is tall. We never do that. But what we do is some sort of approximation or given a set of classes we want to find out what is the likelihood that a person belongs to one of the classes.

So, that is how we say that a person is very tall, the person is tall person is not so tall, he is very short, not so short or short things like that. That means when we are doing some sort of classifications, in our classification, we are incorporating some sort of fuzziness. So, if I put it something like this that, from the horizontal axis, I put the height of the person. And on the vertical axis, I will put something like membership grade.

So, our classification will be or the classification rule will be something like this. Say if the person is, his height is between say 6 feet and say 6 feet 6 inches, I will say that the person is tall. So, this is the region. When I say that the person is tall, roughly if it is between 4 feet to say 4 feet 6 inches, we say that the person is short and somewhere in between 5 feet to say 5 feet 6 inches, we say that he is medium or say standard.

Because in India, most of the persons are within this category 5 foot to 5 foot 6 inches. So, I can say this as a standard or medium. So, the membership grade can be put something like this that if the person is tall, I can say that these are 3 different sets. One set is tall, termed as tall, one set is termed as standard, the other set is termed as short. So, I just want to classify like this that whether the person belongs to this set tall or the person belongs to this set short or the person belongs to this set standard.

So, if a person has a height, which is within 6 feet to 6 feet 6 inches, then obviously the person is tall. So, the membership to this particular set tall has to be equal to 1. If the height is in between 4 feet and 4 feet 6 inches, the membership to this set has to be 1 that is maximum. If it is within this membership to this set has to be 1 that is maximum. You find I have a lot of space between 4 feet 6 inches to 5 feet. Similarly, it is 5 feet 6 inches to 6 feet. Similarly, beyond 6 feet 6 inches below 4 feet.

So, these are the different height regions, which are not marked as belonging to any of these sets. So, I will have a maximum value of say equal to 1, maximum membership grade equals

to 1. And when the height is between 4 feet to 4 feet 6 inches, the membership given to this particular set is 1. Between 5 feet to 5 feet 6 inches, the membership given to this set is 1. Between 6 feet and 6 feet 6 inches, the membership given to this set is 1 that is maximum. And the remaining region can be done like this. If the height is less than 6 feet, depending upon the difference from 6 feet, I can reduce the membership grade. So, I can have a membership grade something like this, so over here in this region.

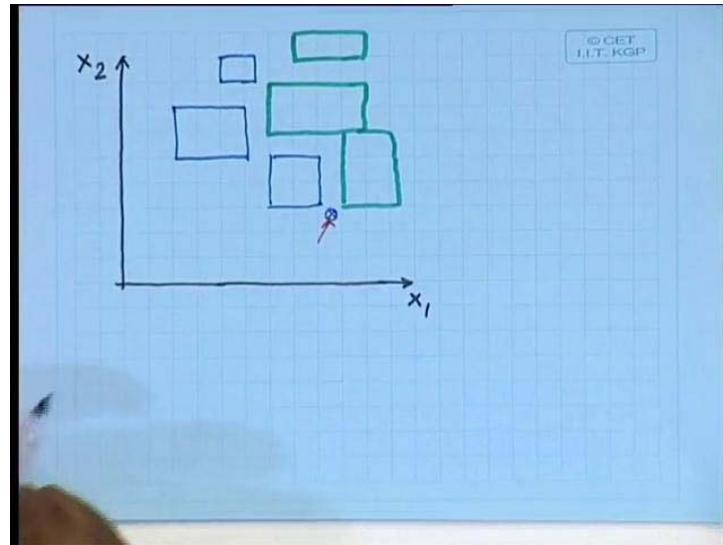
Let us consider 5 feet 6 inches to 6 feet. If the height is say around 5 feet 9 inches, so I will be somewhere, say somewhere over here at this position. So, in this location, you find that the person gets two membership values in two of the classes. So, the membership value to class tall is this to say set tall is this much membership value, say set short is this much. So, this itself consider as fuzzy classification because it tells me that what is the likelihood that the person is tall and what is the likelihood that the person is short.

This measure itself or that way I generate a vector actually. So, membership to class short falls to 0 at this point. At this point, membership to class short will be 0, membership to class standard will be this much, the membership to class tall will be this much. So, actually I get a 3 dimensional vector, 3 dimensional vectors of membership values to different classes. So, that is what fuzzy membership vector. So this fuzzy membership vector itself can be taken as the classification, which is called fuzzy classification.

If I want to make a hard decision that is decide about whether the person is stall or the person is short or the person is standard, then what I have to do is that I have to put a max operator. I have to find out what is the maximum of membership values. And the membership value to which ever class is maximum, I classify the person to that class. So, in fuzzy logic, taking this maximum operator is nothing but fuzzy union.

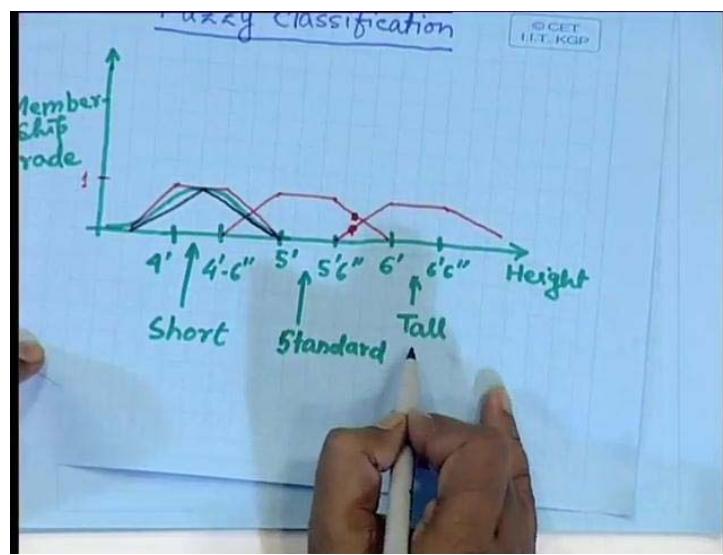
So, fuzzy union gives you the maximum of the fuzzy membership values. On the other hand, if I take the fuzzy intersection, it gives the minimum of the fuzzy membership values. So, I said I will not go into the details of this fuzzy logic or fuzzy algebra, but just for our application what is required, I am discussing that. Now, after giving introduction to this fuzzy membership values, let us see at how we can make use of this concept in our hyper box classification.

(Refer Slide Time: 31:57)



So, what I will do is given this unknown feature vector. I will try to find out that what its membership value to different hyper boxes. That membership value or the set membership values themselves can be used as the classification output. I can find out the maximum of these membership values. I can classify this unknown object to the class of the hyper box to which its membership value is maximum. Now, when I talked about this sort of membership profile, this membership profile is not unique. I can have other types of membership profiles as well. This is what is simplest the membership profile can be something like this.

(Refer Slide Time: 32:52)



It can be a triangular membership something like this. There are different types of membership profiles that can be used. This is triangular one. This is something like your

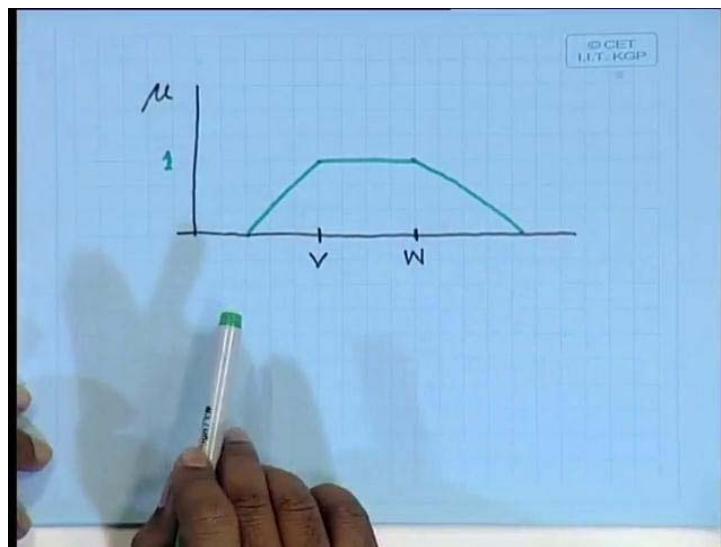
Gaussian curve. What is mostly used is this sort of membership that is trapezoidal membership profile. You will find that this is nothing but a trapezium. What is most commonly used is the trapezoidal membership.

Now, the question is how do you compute such membership functions. So, you can consider in our case, this point may represent our main point corresponding to this particular class tall. This point will represent max point. So, as we said, if the unknown feature vector falls within the hyper box that within the min point and max point, then the membership value to that particular class will be highest. That is equal to 1.

As it falls beyond min point or beyond max point, not within the hyper box, membership value has to be reduced depending upon its distance from the hyper box and the distance I can compute as distance from the min point. If it falls to the left of min point or I can

compute the distance to be distance from the max point. If it falls to the right of the max point, so the kind of situation that I have is like this.

(Refer Slide Time: 34:43)



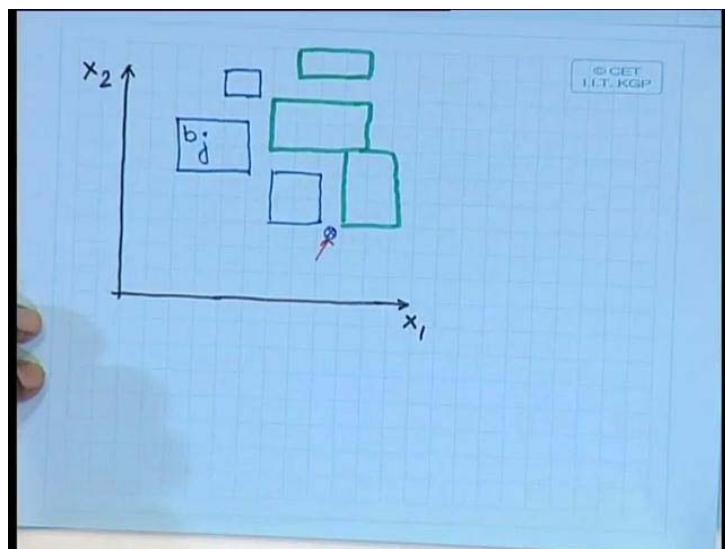
So, let us say that this is where I have the min point V. This is where I have the max point W. so, within the min point and max point, the membership value that will be computed that has to be maximum that is equal to one and beyond min point and max point, the membership value has to be reduced something like this.

So, this is my, on this axis. I have the membership value and the fuzzy membership value is usually represented by μ . That is what is conventionally used. I can use some other letter as

well. That is not a problem, but if I use something else, I have to define what I am using. But if I use this symbol, it is known to everybody that it represents fuzzy membership function. So, how do I compute this sort of membership function given the min point and the max point that is V and W?

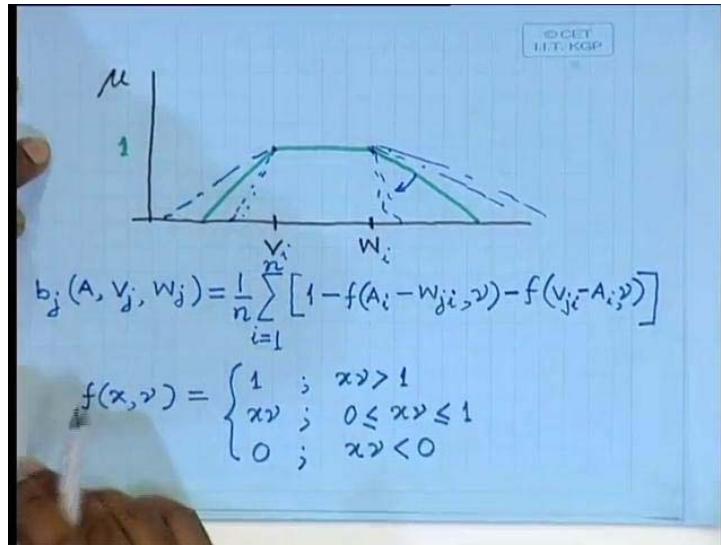
I am just writing in two dimensions. Actually, I have multi-dimensional feature vector. So, both of these min point and max point, V and W will be multi-dimensional vectors. So, if I define a function like this, so as I have multiple number of hyper boxes, so I have to identify which hyper boxes.

(Refer Slide Time: 36:36)



Suppose I consider a j^{th} hyper box. The membership function that will be computed by this j^{th} hyper box is represented by b_j . So, I have to find get an expression of this b_j and b_j will depend upon 3 terms or 3 of the variable, 3 arguments; one is the min point of b_j because b_j is a hyper box that is min point of b_j , which we will represent as V_j . The max point of the hyper box b_j will be represent as W_j . Our input feature vector, which is because depending upon the position of the input feature vector, I have to find out what is it is membership to the hyper box b_j .

(Refer Slide Time: 37:26)



So, I can have a functional representation something like these say b_j arguments are suppose my input feature vector A . Then V_j , which is min point and W_j , which is the max point. So these are the 3 arguments on which this b_j will compute the membership function.

And this membership function can be computed like this. If it is a n dimensional feature vector or the feature space that I have is n dimensional feature space, then I have to compute the membership function considering all the dimensions.

So, I define a function something like this $b_j(A, V_j, W_j) = \frac{1}{n} \left[\sum_{i=1}^n 1 - f(A_i - W_{ji}, \gamma) - f(V_{ji} - A_i, \gamma) \right]$

where W_{ji} is the i^{th} component of the max point W_j . This max point W_j is the max point of my j^{th} hyper box and V_{ji} is the i^{th} component of the min point V_j . This min point V_j is the min point of my j^{th} hyper box. Take the summation of this because I have n different dimensions. And this I am representing in i^{th} dimension. So, take the summation of this for $i = 1$ to n to consider all the dimensions and take the average.

But still this is not complete because I have not defined what these functions f , right? So, I can define these functions f like this that $f(x, \gamma) = 1$, if $x\gamma > 1$. $f(x, \gamma) = x\gamma$, if $0 < x\gamma < 1$.

And $f(x, \gamma) = 0$, if $x\gamma < 0$. This is how I define the function f . Now, you see what it implies with respect to this particular diagram. If the point A_i , I am considering only the i^{th} dimension that is true for all other dimensions. If I consider the vector A , which is the unknown feature

vector here falls within the hyper box, I am considering the i^{th} dimension. So, let me put this as V_i , this as W_i and A_i will be somewhere in between.

So, if A_i lies somewhere in between, that means $A_i > V_i$ but $A_i < W_i$. Now, come to this term $A_i < W_i$ so, $A_i - W_{ji} < 0$. Similarly, $V_{ji} - A_i$. So, $V_{ji} - A_i$ that will also be less than 0. And in this functional definition, this x is nothing but $A_i - W_{ji}$ or $V_{ji} - A_i$. So, both these terms being negative, my $x\gamma$ will also be negative. γ is a positive quantity. So, this term $x\gamma$ that term will also be negative.

So, I am in this region in this definition because $x\gamma < 0$. It is negative. So, $f(x, \gamma)$ becomes equal to 0. So, both these functional values $f(V_{ji} - A_i, \gamma)$ and $f(A_i - W_{ji}, \gamma)$, both of them are 0.

As both of them are 0s, this summation or single term in this summation determines a value 1, which is summed over $i = 1$ to n . So, I get value equal to n because this 1 is being added n number of times, take the average divide by n I get a value equal to 1.

So, that clearly shows if the point lies between min point and max point, then the membership value, which is computed will be equal to 1. Is that okay? Now, let us come to the other case. Suppose that the point lies to the other side to the left of V_i . If the point lies to the left of V_i , then $A_i < V_i$. At the same time, $A_i < W_i$. So, if $A_i < V_i$, then this term is positive because this is $V_{ji} - A_i$ and $A_i < V_i$. So, this term is positive coming to this term, $A_i < W_i$. So, $A_i - W_{ji} < 0$. Since that, this term is negative, so the value of this function is equal to 0 as per this definition.

Whereas value of this function will be either this or this depending upon what is the value of $(V_{ji} - A_i)\gamma$. So, if $(V_{ji} - A_i)\gamma > 1$, then the value of this $f(x, \gamma) = 1$. If this value is equal to 1, then what does this term give me? 1-1 is equal to 0. Whereas if $(V_{ji} - A_i)\gamma$ is in this region, it is between 0 and 1. Then the output given by this term is $1 - x\gamma$. So, $1 - x\gamma$ what is x ? x is nothing but the distance from the min point. So, that is my x .

So, it is $1 - x\gamma$ that means I get a straight line with a slope equal to γ . So that is this straight line. Similarly, if the point falls to the right of W_i , in that case that means $A_i - W_{ji} > 0$. A_i is also greater than W_{ji} , right? So, $A_i - W_{ji} > 0$, whereas $V_{ji} - A_i$, this term will be negative. As

this term is negative, so this function, functional value will be equal to 0. This functional value will be either this or this.

So, if the distance from W_{ji} of the point times γ is very large, it is greater than 1, then the output will be 0. This functional value the membership value will be equal to 0, whereas if the distance times γ is in the range of 0 to 1, then the output is $1 - f(x, \gamma)$. Again, I get a straight line with a slope, which is equal to minus γ . So, those are these two different straight lines.

Now, see what this γ do over here. Over here, γ is the slope of the straight line. Over here, minus γ is the slope of the straight line. That means the γ ; this particular parameter γ is indicating how steep this line is? If the γ is high, then the straight line will be something like this. The slope will be very high.

If the γ is small, then the straight line will be something like this. And that is what indicates the degree of fuzziness. When the straight line is bending in this direction, that is the slope is becoming more and more. That means we are moving more towards hard classification or crisp classification as it called. If the value of γ is less, then the crispness or hardness is reduced, at the same time, the fuzziness is increased. So, γ is a parameter, which controls the degree of fuzziness. If the value of γ is more, then the fuzziness is more. If the value of γ is less, then the fuzziness is also less, right?

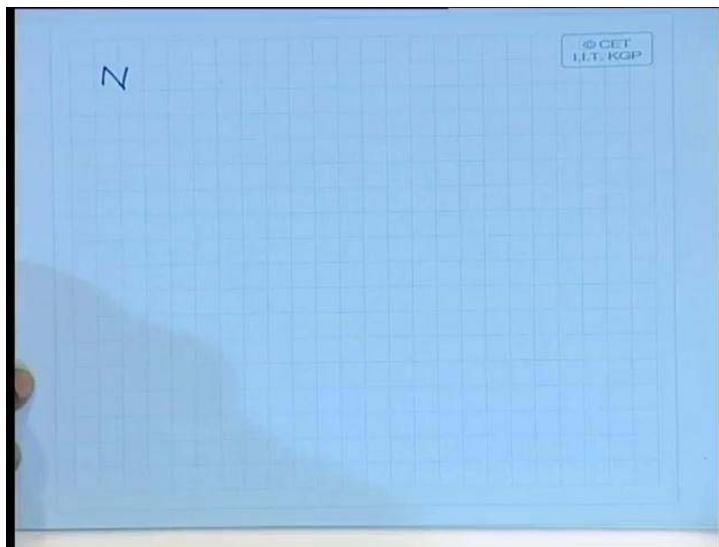
So, using this particular expression for different hyper boxes, I can compute the fuzzy membership function of this unknown feature vector to all the different hyper boxes. Now, to find out the fuzzy membership function of this unknown feature vector to a particular class, I will consider the membership functions computed by different hyper boxes belonging to that class.

So, here if I consider that this blue boxes represent class ω_1 and the green boxes represent class ω_2 , I will compute the fuzzy membership function of this unknown point from each of these two boxes and take the maximum of these membership functions. That gives me what is the membership value of this unknown feature point to the class ω_1 .

Similarly, I compute the fuzzy membership function, fuzzy membership value of this unknown feature point from, to each of these green boxes. Take the maximum of them that gives you what is the membership value of this point to this particular class. And once I get that, I get two membership values; 1 belonging to one for class ω_1 other one for class ω_2 .

So, these two membership values themselves as I said can be used as fuzzy classification. And if I want hard or crisp classification, then I have to find out what is the maximum of these two and whichever is maximum, I have to classify this unknown feature vector to that corresponding class. Is that okay? Now, how does this algorithm will actually work? I have a set of training samples.

(Refer Slide Time: 49:31)



Say I have capital N number of training samples let us assume, these training samples are coming from different samples. So, if I have two classes, some of these samples are taken from class ω_1 and some of the examples are taken from class ω_2 . So, what I will do is, I will consider this training samples one after another and while doing so, I will go on creating the hyper boxes, right?

So, when I consider the first training sample, I do not have any hyper box created till that time, right? So, what I will do is I will create the first hyper box. I have a single point. So, for the first, the first hyper box, I will put the min point and the max point to be the same. That means it is a point hyper box, which is same as the coordinates of that particular feature vector.

Then, I consider the second point, if the second point because I am considering the points arbitrarily. It is not necessary that second point will come from the same class. So, if it comes from the same class, then I have to go for expansion of the hyper box. So, I have an initial point. Then I have a second point. So, I have to go for expansion of the hyper box to include this second point.

If it belongs to another class to the second class, then instead of expansion of the previous hyper box because previous hyper box belongs to some other class, instead of expanding the previous hyper box, I have to create a second hyper box. Again, a point hyper box because this is the first point coming from the second class. So, I have to create a second hyper box, which is again a point hyper box for which the min point and max point is same. And I have to go on doing that wherever possible. I have to expand the existing hyper box to include the new point which is considered; otherwise I have to create a new hyper box.

Now, while I expand the hyper box, there is a danger, the danger in the sense if I come back to this figure see over here, no we should not. If it is already within that hyper box that means it is already covered. If it is not covered, suppose in this particular case, I go on expanding this particular hyper box. If I expand this hyper particular hyper box, you find that it covers points from another class. So, when I expand an existing hyper box, then there is a risk that the amount of error we will incorporate that will go increasing.

So, one extreme case is every point is considered as a point hyper box. The other extreme case is all the points irrespective of its class belongingness is put into the same hyper box. So, if every individual point is taken as a point hyper box, I do not have any error. In the other case, I have maximum error in between I have error varying from maximum and minimum. So, if I allow the hyper boxes to expand in unconstraint fashion, then that risk of incorporating error is more.

So, I have to put some constraint in the way the hyper boxes can be expanded. So, any time, I get a new point belonging to a class of an existing hyper box, I should not expand the hyper box, which is already existing in an unconstraint fashion in an uncontrolled fashion to include this new point. So, there should be some constraint up to which I can expand a hyper box; beyond which I cannot expand a hyper box. So, I will stop here today. We will continue with this discussion in the next class.

Thank you.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

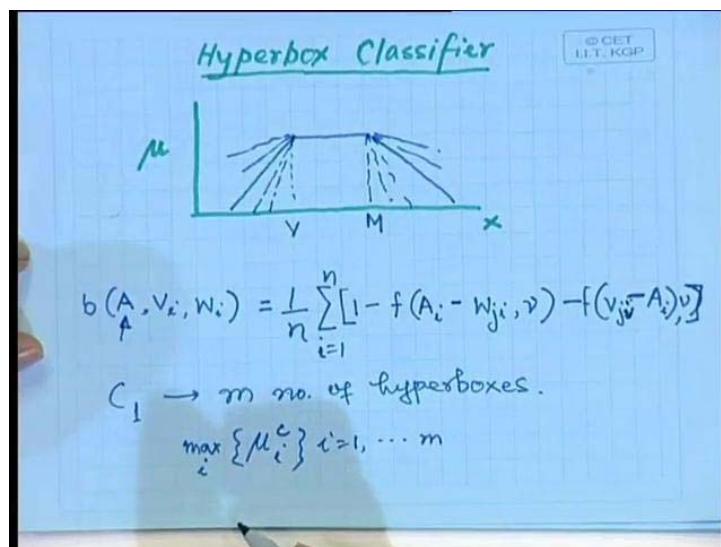
Lecture - 31
Hyper box Classifier (Contd.)

Good morning. So, today we will continue with our discussion on hyper box classification. So, in the last class, what we have said is if the feature samples the feature vectors belonging to different classes, they are mixed in an arbitrary manner. Then we cannot really design a linear classifier or a quadratic classifier to classify different samples to different classes.

So, in such kind of situation, the kind of classifier that we have talked about is hyper box classifier. Where you divide the feature space with the number of hyper boxes. And the different hyper boxes will be given different class labels. And then when an unknown sample comes if the unknown sample falls one of hyper box is well within hyper box, which belongs to particular class, then this unknown sample will also be classified as belonging to the same class which class label that has been given to that particular hyper box.

Whereas if the unknown sample falls outside the hyper boxes outside any of the box, then find out what is the distance of this unknown sample from the hyper boxes. And based on the distance, we can give a fuzzy membership grade. To whichever class the fuzzy membership function is maximum, we can classify the unknown sample to that particular task.

(Refer Slide Time: 01:51)



And for that, we have said yesterday that offers membership function or trapezoidal membership function that can be employed, this is the direction of the feature. On this side, I put the membership value, which is μ and the fuzzy membership function. If I define something like this, a trapezoidal fuzzy membership function where these are the min points and max points because as we said that the hyper box can be represented by a min point and a max point and accordingly it is called min max hyper box.

So, if I have a fuzzy membership function, a trapezoidal fuzzy membership function something like this, and we have given an analytical expression for the sort of fuzzy membership function which is given by, so you can define, we can give an analytical expression of the fuzzy membership function something like this where A is input sample vector V_i presents the min point in the i^{th} dimension and W_i represents the max point in the i^{th} dimension and γ is a parameter, which represents the degree of fuzziness. So, as we said if the value because the slope of these lines that depend upon the value of γ , so if the value of γ is more, then the fuzziness will be less.

It will be more towards hard classification because if I increase the value of γ , I will have these slanting straight lines something like this. Whereas if I reduce the value of γ , then I will have the straight lines something like this, which increases the fuzziness. So, this γ actually controls the degree of fuzziness. So, you can compute given a mini max hyper box where I know the min point V_i and the max point of W_i . This is what I have given on the membership calculation.

The membership calculation, membership computation only along the i^{th} dimensional. Such membership computation has become among all the dimensions added together and then normalize with respect to n . So, once I have this sort of analytical expression for computation of the membership, I can compute the membership of an unknown sample to different hyper boxes or because in this case I will have for every class a number of hyper boxes.

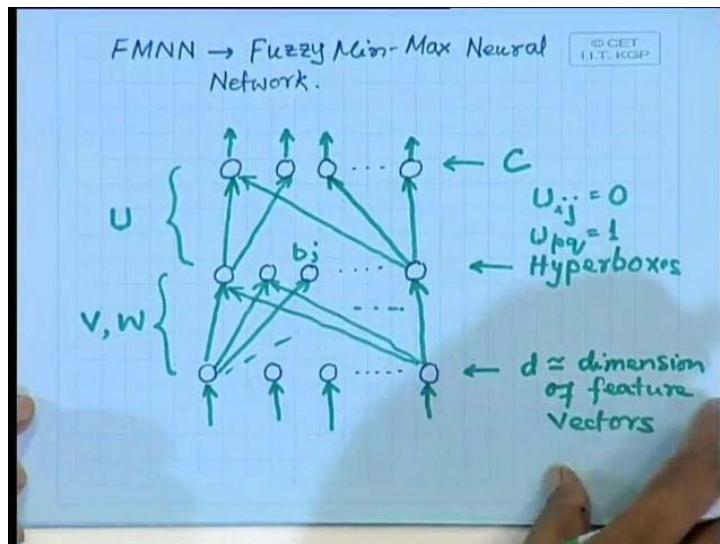
So, considering a particular class c_1 which has say m number of hyper boxes, so class c_1 has m number of hyper boxes. So, accordingly for each of this hyper box, I will have a membership function of this form. So, there will be μ_i^c for class c_1 . so let us, if I put it like this, it is μ_i^c for i varying from 1 to m . As I have m number of hyper boxes, so for each of these hyper box, I will get a membership function.

So, I will have m number of membership function like this and the maximum of this, maximum over i gives me what is the membership function of the hyper box belonging to class c , which is nearest to this unknown sample. Because obviously a hyper box, which is nearest to unknown sample that hyper box will get the maximum of membership value and the hyper box, which is farthest from this unknown sample that hyper box will give minimum membership value.

So, I take the maximum of these m number of membership values given by m number of hyper boxes belonging to class c . And this maximum value is the membership value of this unknown sample to class c_1 . So, what I have to do is I have to compute membership function with all the hyper boxes. Then particular class are to take a maximum of the membership functions and that gives the membership of value to that particular class.

So, that way when I get the membership values to all other classes, whichever class gives the maximum value, I classify this unknown sample x to that particular boxes. So, this is how this hyper box classifier has to work. Now, in the year 1992, Patrick Simpson said that this concept can be converted in the form of a neural network. So, accordingly neural network architecture is fuzzy min max neural network.

(Refer Slide Time: 07:56)



So, FMNN or fuzzy min max neural network for pattern classification was proposed by Patrick Simpson in the year 1992. So, what is this concept of fuzzy min max neural network?

The fuzzy min max neural network it has got one input layer and one output layer and one so called hidden layer. Now, this output layer is same as the number of classes that we have. So, this is equal to the number of nodes in output layer is same as the number of classes.

So, if I have c number of classes, there are c number of nodes in the output layer. Similarly, number of nodes of the input layer is same as before that is equal to the dimensionality of the feature vectors. So, this one number of nodes in the input layer will be d , which is nothing but dimensionality of features or feature vectors. And the hidden line nodes they actually represent hyper boxes. It is the responsibility of the hidden layer nodes to compute this membership function.

So, if the hidden layer nodes have to compute this membership function, then the hidden layer nodes or every hyper boxes as I am saying that these hidden layer nodes they represent the hyper boxes. They have to know what is the corresponding min point and max point. So the min point and max point is represented in the form of connections. So, the connection between the input layer nodes and the hidden layer nodes will be represented by two matrices corresponding to the min point V and max point W .

So, I will have all these connections from input layers to the middle, hidden layers something like where these connections actually represent the min points and the max points. This connection weights are represented in the form of two matrices; one matrix corresponded to the min point and one matrix corresponding to max point. And the connection from the hidden layer nodes to the output layer nodes represents different classes.

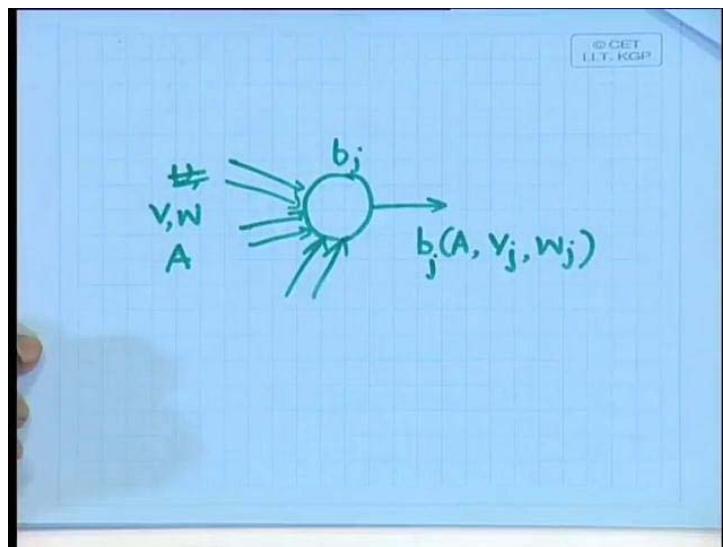
So, if hyper box say b_j is given the class label of c_i , then only I will have a connection from this hyper box b_j to the responding output nodes c_i . For a hyper box, which does not belong to a class of particular class, there will be no connection from that hyper box to the corresponding class. So, I can represent this connection from the hidden layer nodes to the output layer nodes by another matrix say u where the elements of the matrix u can be either 0 or 1.

If it is 0, say u_{ij} if an element $u_{ij} = 0$; that represents that I do not have a connection from the i^{th} node in the hidden layer to the n^{th} node in outer layer. That means the corresponding hyper box, i^{th} hyper box does not belong to that class c_j , where as if say up q that is equal to 1 that indicates

the hyper box p^{th} hyper box in the hidden layer belongs to the q^{th} class and the corresponding element in the matrices will be set to 1.

So, accordingly I will have connections from the hidden layer nodes to the nodes in the output layer. So, this being binary either I will have a connection or I will have no connection. So, this connection weights are either 0s or 1s. So, as before, you feed the input vector to the input layer node. So, this input vectors are passed on to the hidden layer nodes. The hidden layer nodes know what are the min point and max point of the corresponding hyper box through these matrices elements V and W . They compute the corresponding membership function, right? So, the architecture of a node in the hidden layer will be something like this.

(Refer Slide Time: 14:09)

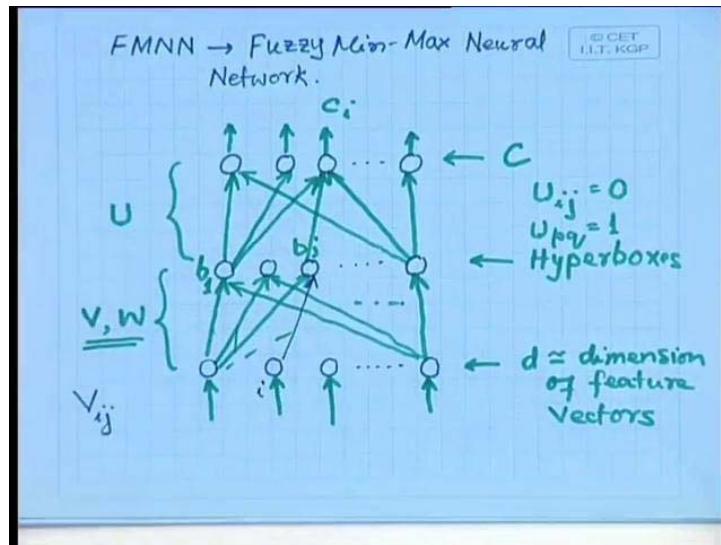


It will have the connections corresponding to V and W , which are the min points and the max points. It will also have the input vector unknown vector A , right? And it will give me an

$$\text{output, which is nothing but this function } b_j(A, V_j, W_j) = \frac{1}{n} \left[\sum_{i=1}^n 1 - f(A_i - W_{ji}, \gamma) - f(V_{ji} - A_i, \gamma) \right].$$

So, if this node this is the j^{th} hyper box nodes, let us represent it by b_j . So, in this architecture, each of these hidden layer nodes will compute the corresponding membership, right? So, when b_j say j^{th} node because all these hyper boxes are created during the training operation or learning operation.

(Refer Slide Time: 15:29)



So, if this b_j is connected to say an output class c_i , so I will have this corresponding connection with equal to 1, right? So, this output membership value is collected by this output node output and node c_i . Similarly, if I have another node says b_1 that might also be labeled as class c_i . So, this hyper box b_1 , it will also calculate the corresponding membership function. And that membership value will be passed onto this node c_i . So, this node c_i actually gets the membership values from different nodes, which are computing the membership values in the hidden layers.

Finally, the responsibility of this node c_i will be to make a fuzzy union of all this membership values that it gets. This fuzzy union is nothing but max operations. So, it finds out what is the maximum of the membership values. And that maximum is taken as the membership value of this unknown vector to this class c_i . So, that way, each of these output layer nodes will compute the corresponding membership value. And as we said earlier that outputs of this output layer node or the membership values computed by this output layer nodes that itself can be taken as fuzzy classification.

Or if I want to form a hard classification, I will take the maximum of this values and whichever output node gives the maximum membership value, I classify the unknown sample to that corresponding class. So, how the training process or the learning process will continue? Suppose that I have got m number of samples, training samples. The training samples are labeled as belonging to different classes. I take the samples one by one. So,

whenever a sample is taken for the first time, I have to create one hyper box node. That means I have to create one node in the middle layer.

So, initially what I have is I have this output layer nodes. I have this input layer nodes. The number of output layer nodes is fixed that is same as the number of classes that I consider. The number of input layer nodes that is also fixed. It is same as the dimensionality of the feature vectors. The nodes in the middle layer or the hyper box nodes they are created while training, that is in the form of connection weights. That is given in the form of connection weights and over here unlike in other neural networks like perceptron single layer perceptron or multilayer perceptron where we make a weighted summation of the input feature components that is a linear equation here is not a linear equation.

Here each of these nodes has to compute this membership function, this membership value. So, whether you call it as weights, connection weights or some information available to the hidden layer nodes, the information is saying is represented in the form of two matrices V and W . Where matrices, V will give you the min point information and W will give the max point information. Is that okay? So, a particular element v_{ij} represents what is the min point if I consider that this is the information going from the i^{th} node in the input layer to the j^{th} node in the middle layer.

If I consider this, two weights are associated with every node one corresponding to the min point one corresponding to the max point, so this hidden layer nodes, they only need to know this information min point max point along with the feature vector to be classified. Is that okay? So, while training as we are giving a number of labeled samples for which the class belongingness is already known. And initially, what I have is I have this output layer nodes. I have this input layer nodes. I have three empty matrices, right? One corresponding to u , one corresponding to V , one corresponding to W . These matrices are empty initially and this hidden layer that is also empty. There is no node in the hidden layer.

So, while training, I take the first training sample and because this is the first training sample that I am encountering, so we said in case of hyper box creation, that I have to create one node in the hidden layer, right? and corresponding to that, I have to fill up these V matrices and W matrices, right? What will be the V and W matrices? Initially, V and W will be identical. As I have only one hyper box corresponding to single point, so min point and max point will be same, right?

I also have to establish the connection from this created hyper box to output layer or one of the nodes in output layer. So, what I will do is if the first point has been taken that belong to class c_1 , I will make the corresponding entry in the U matrix equal to be 1. All other elements in the U matrices are zero.

That means I am making a connection from the hyper box, which is created in the middle layer to the class nodes c_1 in the output layer, is that okay? Whereas other connections from hidden layer to the output layer they are missing. All of them are represented by 0 in the matrix U, is that okay? Next time, I get the second point, second point either may belong to class c_1 or it may belong to another class.

If it belongs to another class, then what I have to do is I have to create another node in the middle layer corresponding to another hyper box. And that will be a point hyper box because I have not encountered any point belonging to that class earlier. So, that would be a point hyper box. Accordingly, I have to enter information in v matrices w matrices and U matrices. I get the third point assuming that this class of the third point I already have created the hyper box in the middle layer if I assume that one or more hyper boxes in the middle layer.

Then, I have to see that whether this third point for which at least one hyper box that is already been created whether third point is will be accommodated in those hyper boxes or not. If it falls within the hyper box, which is already created, I do not have any problem. I do not have to create anymore node, any new node for that. But if it is outside the hyper box and for that class one or more hyper boxes has already been created, then first what I have to check is whether this new point can be included in one of the hyper boxes or I have to create a new node for this new point.

So, the way this new point can be included in one of the hyper boxes is that you select a hyper box, which is nearest to that point and expand the hyper box whether if the expansion is permitted. So, if I have to have criteria to decide whether the expansion is permitted or expansion is not permitted, because as we said that when we try to stretch a hyper box, then there is a risk that samples belonging to other classes will also be included in the same hyper box. So, accordingly, I have to put some constraint or how much a hyper box can be expanded, is that okay?

(Refer Slide Time: 24:49)

$$n\theta \geq \sum_{i=1}^n [\max(W_{ji}, A_i) - \min(V_{ji}, A_i)]$$

$\theta \rightarrow \underline{\text{Expansion parameter.}}$

So, this expansion criterion for hyper boxes is given by this in θ , which $n\theta \geq \sum_{i=1}^n [\max(W_{ji}, A_i) - \min(V_{ji}, A_i)]$, i varying from 1 to n , when I have n number of dimensions, whereas before W_{ji} represents the i^{th} dimension of the max point of j^{th} hyper box. Similarly, this is the i^{th} dimension of the min point of the j^{th} hyper box. A_i is the i^{th} component of the feature vector, which is to be included in the j^{th} hyper box. So, I expand a hyper box. Then naturally its min point and max point, they are going to be disturbed, right?

And as we said that max point is nothing but the maximum coordinate value of all the feature vectors within that hyper box. So, the new max point in the i^{th} dimension will be maximum of these two. Assuming that this max point will be going to shift to A_i , so that maximum these two will represent the new max point and minimum of these two will represent the new min point.

And the difference between the min point and max point that is going to tell what is the spread or width along the i^{th} dimension. So, sum of all these widths in different dimensions in all the n dimensions that must be less than $n\theta$. As long as this condition is satisfied, we are allowed to stretch or expand the hyper boxes. The moment I find that this value exceeds $n\theta$, then expansion of hyper box is no more permitted, where this term θ , it is called the expansion parameter, is that okay? So, while trying to incorporate a feature vector in a class for which some hyper boxes has already being created I have to first try to see whether the

hyper boxes can be expanded to include this feature, if the expansion is permitted, then I simply expand the hyper box. Over here, expansion of hyper box means I have to modify the min point and the max point.

So, I do not have to create any more, any new hyper box. What I have to do is I have to change the min point and max point of an existing hyper box. That means I have to change the values in the V matrices and W matrices. That is the only thing I have to do. I do not have to do anything else. If I have to create a new hyper box, then I have to make a corresponding entry in the V matrices, W matrices. I also have made an entry in the U matrices.

But in this case, U matrices entry already exists because one of the existing hyper box, I am going to expand. Only the min points and max points are to be modified. So, that modification, I will do in the V matrices and W matrices. So, that is how the learning will continue. But while doing learning as there is a provision of expansion of hyper boxes, other problems may come in, that is it is possible that a hyper box and expanding that overlaps with another hyper box. So, let us just take an example to see that how this overlapping or expansion may affect our performance of the fuzzy min max neural network. So, let us take few points belonging to say 2 different classes class 1 and class 2.

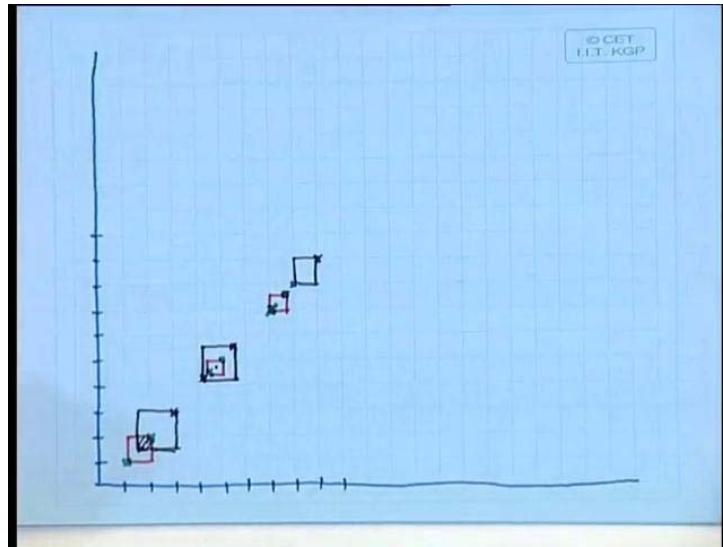
(Refer Slide Time: 29:37)

<small>©CET I.I.T. KGP</small>	
$A(0.7, 0.7) \rightarrow 1$ $B(0.75, 0.75) \rightarrow 1$ $C(0.3, 0.9) \rightarrow 2$ $D(0.8, 0.8) \rightarrow 2$ $E(0.1, 0.1) \rightarrow 1$ $F(0.2, 0.2) \rightarrow 1$ $G(0.3, 0.3) \rightarrow 2$ $H(0.15, 0.15) \rightarrow 2$ $I(0.45, 0.45) \rightarrow 1$ $J(0.5, 0.5) \rightarrow 1$	$K(0.42, 0.42) \rightarrow 2$ $L(0.55, 0.55) \rightarrow 2$

So, I take a point A whose coordinates are say $(0.7, 0.7)$. Suppose that this point A belongs to class 1. I take a point B whose coordinates are $(0.75, 0.75)$. Suppose that this also belongs to class 1. Say these are the points of feature vectors 2 dimensional feature vectors, for which

these of the class belonging that is specified. So, let us assume that these are the training vectors using, which I have to generate the hyper boxes. So, let us take these points one by one. So, I take this 2 dimensional feature space.

(Refer Slide Time: 31:51)



So, let me consider this 2 dimensional features space. So, you find that the first point that I have that is A whose coordinates is $(0.7, 0.7)$, this sample belongs to class 1. So, I take 0.7 over here. So this is $(0.7, 0.7)$. This sample belongs to class 1. The next point is $(0.75, 0.75)$ that also become to class 1. This $(0.75, 0.75)$ is over here and this point belongs to class 1. So, initially, I will have 1 hyper box. Where the min point and max point will be same as $(0.75, 0.75)$ next time another sample belonging to same class having points as $(0.75, 0.75)$ because it belongs to the same class. So, I have to try that whether this hyper box and expanded to include this particular point.

Now, assuming that depending up on the value of θ , this hyper box expansion is permitted. So I shall have a new hyper box, which is given by this. So, this will be a hyper box responding to class 1. Next, I have a point given by $(0.9, 0.9)$, which belongs to class 2.

So I have this point this one $(0.9, 0.9)$, which belongs to class 2 a point over here. The next point is $(0.8, 0.8)$; that also belongs to class 2. This one. So, $(0.8, 0.8)$ belongs to class 2. So, initially I will have one hyper box for this. Next, I have to see that whether for this 1 hyper box and expanded to include this point. And let us assume that this expansion is also permitted.

So, I will have this box corresponding to class 2. Then I will have this point for class 1 that is $(0.1, 0.1)$, $(0.1, 0.1)$ is somewhere here this belongs to class one. Then I have this point $(0.2, 0.2)$. I have this, you find that I have already an existing hyper box, this one, which also belongs to class 1, so I try to include there is by expanding this hyper box. You find that the hyper box size will be quite big.

So, here this expansion may not be permitted by our expansion criteria. So if this expansion is not permitted then I have to create a new hyper box for this point. So, assuming that this is new hyper box is created, so the next point that I take is $(0.2, 0.2)$, which also belongs to the same class 1. So, I have $(0.2, 0.2)$ over somewhere over here. Now, I have 1 hyper box responding to class 1 and another hyper box corresponding to class 1. So, I have to try to see which one can be expanded to accommodate this.

So, here I find that this is the nearest hyper box and assuming that this expansion is permitted, I get hyper box of this point. Then I have to take other training samples. So, I have this $(0.3, 0.3)$, which belongs to class 2. So, $(0.3, 0.3)$ somewhere over here, which belongs to class 2. I have $(0.15, 0.15)$, which also belongs to class 2. So, $(0.15, 0.15)$ is somewhere over here.

So, when I have created this hyper box, this hyper box out of class 2 already existed. Again supposing that this expansion is not permitted, this was not expanded, so I have to create a new hyper box, yet another hyper box, if this expansion is not permitted. Now, let us assume that this expansion is permitted.

So, I have the new hyper box, which is something like this. So, here you find that what I have is an overlap because this hyper box belongs to class 1, whereas this is hyper box belongs to class 2 and these two hyper boxes overlap. So we will see that how to tackle such overlap situation? So, let us take other training samples. So, again I have this point $(0.45, 0.45)$, so $(0.45, 0.45)$ somewhere over here, which belongs to class 1. Then I have $(0.5, 0.5)$ that is somewhere here that also belongs to class 1. Suppose that these two formed 1 hyper box. Next, I have this $(0.42, 0.42)$. Which is, this was 0.45. So, $(0.42, 0.42)$ will be somewhere over here, right? And then I have $(0.5, 0.5)$, which is somewhere over here. Assuming that these two form a single hyper box, I have a situation like this.

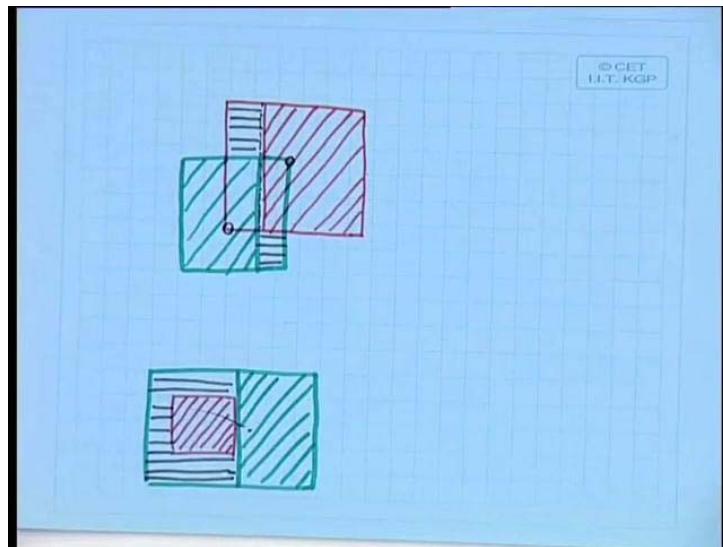
So, here to find that 1 hyper box belonging to class 1 that is completely enclosed within another hyper box belonging to class 2. So, as the expansion of the hyper box are permitted, such types of overlap are likely to occur. So, the question is how to tackle such overlaps. So

this what is called an overlap and this is what is called containment because 1 hyper box belonging to 1 class is fully contained in a hyper box class belonging to another.

If I expand this definition that it may not be necessary that the hyper box is completely contained within another hyper box, it is also called containment. At least 1dimensional of hyper box is completely contained within similar to the same dimension of another hyper box. So, here you find that if I have this overlap or containment with in hyper boxes belonging to the same class that is permitted because hyper boxes belong to the same class. But if the hyper boxes belonging to different classes either there is containment or there is an overlap, then I have a problem in computing the membership function in this overlap region over this region. I cannot compute the membership function because over this region if the sample falls, get to the membership function of 1 for class 1. I will also get the membership function of 1 for class 2.

So, over this entire region, the classification cannot be performed. Similarly, over here, if a sample falls within this to which class we should say whether the sample falls in belongs to class c_1 or the sample belongs to c_2 . So, I have similar kind of problem, if such types of overlap or containment occurs while training this fuzzy min max neural network, so I have to have ways of dealing with this sort of problem. So, what we have seen so far is that?

(Refer Slide Time: 41:09)



I can have either an overlap of this form that is I have a hyper box of 1 class. I have a hyper box of another class. And these 2 hyper boxes overlap, or I can have a situation something like this. I have a hyper box of 1 class, which is fully contained within the hyper box of another class. So, this sort of situation so when this sort of situation occurs; then Simson said that you go for contraction of hyper boxes. Because if hyper boxes belong to different classes they overlap or one contains another, such a kind of containment of overlapping cannot be permitted.

So, you go for contraction of the hyper boxes. When you go for contraction, you have to go for the dimension of min overlap, so that the disturbance incorporated in the hyper boxes, which is already created that is minimum. So, when you contract, then what have to do is I have to break it somewhere in the middle over here assuming that this is the dimension of minimum overlap.

So, I get new hyper boxes, one hyper box will be this and the other hyper box will be this. So, this is one hyper box after contraction and this is another hyper box after contraction. And for the contraction, what I have to do is I have to simply modify min points and max points nothing else. So, by this contraction, you find that the overlap between the two hyper boxes belonging to two different classes that has been removed.

But at the same time, I am incorporating some amount of error at the time of training itself. This is because this region for which information was already gathered one is this region and another is this region. For these two regions, information was already given. So, by this contraction process that information is and not only that when I have created this min much hyper box, this was the min point of this hyper box and this was the max point of this hyper box. By this contraction, you find that this min box point of class 1 has been put into class 2, right?

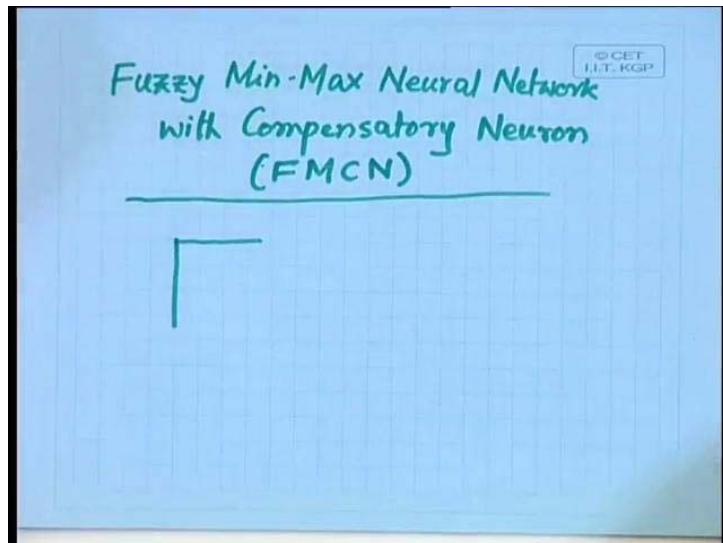
I am not saying first feature along the feature along, which the overlap is minimum, it may be first feature or it may be second feature, may be third feature. It may be any feature. It is along the feature dimension along which the overlap is minimum. So, you find that this min point of class 1 that has been put into the hyper box of the second class and the max point of class 2 has been put into the hyper box of the first class.

So, while training it, I am introducing some amount of error. So, if I introduce amount of error while training, it is quite natural that while testing or when, I get unknown samples the

performance will be poor because I cannot maintain the accuracy during training process itself. Similarly, over here also, I have to go for contractions.

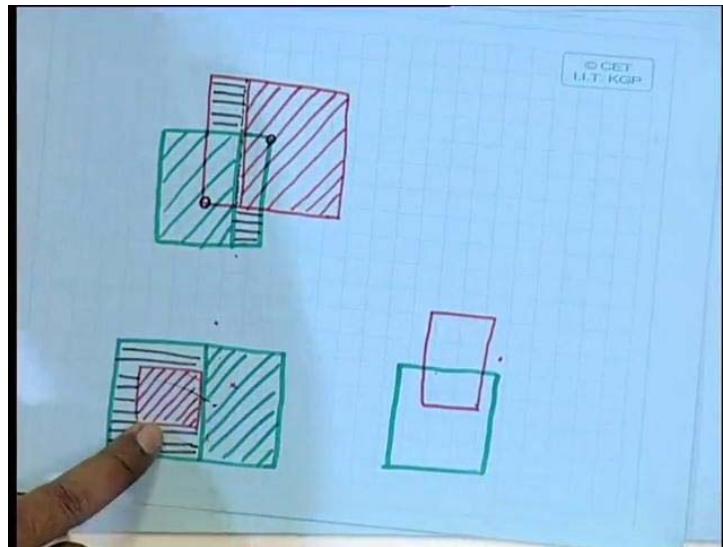
So, after contraction, the hyper box may be something like this. So, the new hyper box will be this one for 1 class and will be this one for another class. So, we find that the information, which is already gathered for this region, this information is lost. So, as this information is lost, so obviously I am incorporating error while training operation. And as I am incorporating error training operation, obviously during the testing or classification of unknown samples, the performance will not be that good so to solve this problem, we had suggested an architecture, which is called fuzzy min max neural network with compensatory neuron.

(Refer Slide Time: 47:05)



So, that is fuzzy min max neural network with compensatory neural. In short, it is FMCN. So, in case of FMCN, what you have said is if I have similar kind of overlap and when I said this sort of containment, it is not necessary that one hyper box has to be fully contained in a hyper box. It is also possible that along one dimension contained with another hyper box. So, I can have a situation like, so this also possible.

(Refer Slide Time: 48:12)



So, containment is only on one dimension, not it is fully contained. So, in case of FMCN, what is suggested as whenever there is an overlap of this form or whenever there is a containment of this form, do not go for contraction of the hyper boxes to avoid this overlap, avoid this containment. But what you do is you try to re compute the values within overlap region, whether it is here or it is here, the logic is as the hyper boxes created using the information of min point and max point.

So, when you try to re compute this membership value, you try to retain maximum membership value along the, around the min point and max point. As you move away from the min point and max point, you try to reduce the membership value. So, in the graded region, the membership value will actually be upgraded membership at this point, which is the max point of hyper box belonging to class 1. The membership value of a sample near to this point will be more to hyper box 1, more to class 1 than towards class 2. Similarly, an unknown point, which is near to this max point, this is max point of the hyper box belonging to class 2.

So, if an unknown point is more towards the max point of this hyper box, its membership value to class 2 will be more than its membership value to class 1. So, putting this logic, what we have suggested is we have a graded membership in this overlap region where the membership value near this min point will be maximum to class 1 and minimum to class 2. And the membership value of an unknown point, which is near to this the max point

of class 2 of this hyper box in class 2, it will have maximum membership to class 2 and minimum membership to class 1. And in between, the membership value will vary linearly, right? And to do this.

Similarly, over here, if an unknown sample falls within this region, what you do is, you do not disturb the structure of the hyper box. The structure remains the same. If an unknown point falls over here, it is within this hyper box belonging to class 2 though it is also within hyper box belonging to class 1. But the nearest hyper box is this 1, the red 1. So, it will have a membership value 1 to this red hyper box. The membership value will be 0 to this green hyper box.

If an unknown point falls over here, it is not within this hyper box. It will have membership value 1 within this green hyper box. I am not saying it will have a membership value 0 within the red hyper box. It will have a membership value to this red hyper box depending upon its distance from the red hyper box, but obviously it will be less than that. So, by max operator because this 1 will have membership value to 1 to green hyper box, which will obviously, be greater than the membership value to red hyper box.

So, max operator obviously, this will be filtered out. And the class membership will be given as the class belonging. This will be given to the green class. When it falls outside this somewhere over here again, its membership to this will be more value than its membership value of this. Naturally, this will be classified to this. Similarly, over here, if the hyper box falls in this region, in the overlap region or in the contained region, it will have a membership value of 1 to this red class, membership value of 0 to the green class.

If the point falls somewhere over here depending upon its distance from these 2 different classes, its membership value will be computed. It will be put into that class for its membership is maximum. So, what we are suggesting is that do not disturb the structure of hyper boxes. This is because disturbance of structure bar of boxes introduces more amount of error. By retaining the structure of the hyper box, you try to modify the membership value computation. So, we will have upgraded membership within the overlap region as well as with the contained region. But, in the contained region, it will not be graded membership value of either 0 or 1. But in the overlap region, it will have upgraded membership. So, accordingly, when we suggest this min max neural network with compensatory neuron, this is actually inspired by the reflex action of our human brain.

What is reflex action? The moment you suddenly touch a hot object or suppose you put your foot on a nail usually, if you have a stimulus that stimulus information sends to your brain. The brain instructs what to do next, but in such sudden event, the instruction does not come from the brain. So you sense the signal instead of going to the brain, there is a bypass. So, immediately, there is a short circuit thing. You get an instruction to your muscle, what to do next? So, it is an instant action or a reflex action. This reflex goes on decreasing with your age. As you become older, the reflex action becomes weaker.

So, it is a similar type of concept that whenever you meet such a kind of situation, you take an immediate action. Such a kind of concept has been introduced in this FMCN architecture that if an unknown sample falls within this overlap region or within the contained region, we consider these to be such a kind of sudden situations. So, immediately you have to take action without going for much of computation. So, to do this for every such overlap region or every such contained region, you introduce a new hyper box or a new node in the hidden level, which are called compensatory neurons.

So, whenever an unknown sample falls within this type of regions in the overlap region or the contained region, then only that compensatory neuron will be active. It will try to act as a modifier of the membership computation, fuzzy membership computation of the output layout over here. So, I will have a set of compensatory neurons. These neurons, these nodes or neurons are actually computing the membership value to a particular class whenever a sample falls within the overlap region or within the contained region. Then these compensatory neurons will modify these values. And the modified value will be taken as membership function to different classes. So, we will discuss about this FMCN architecture in our next class.

Thank you.

Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 32
Fuzzy Min Max Neutral Network for Pattern Recognition

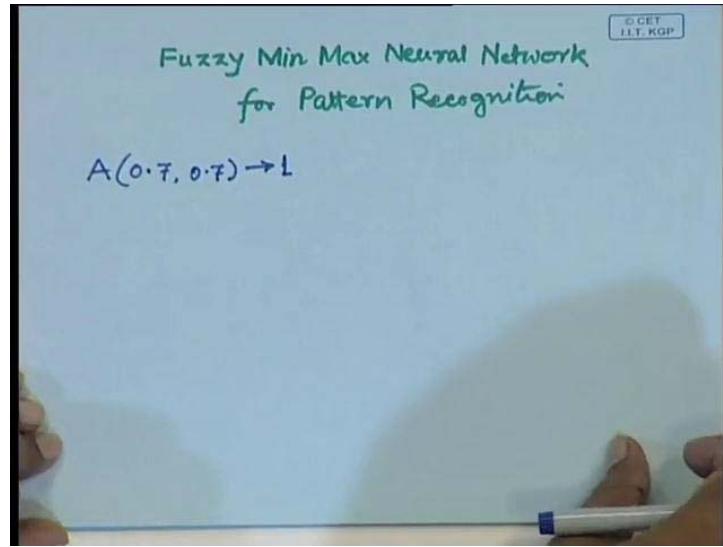
So, we were discussing about the fuzzy neural network, fuzzy min, max neutral network for pattern recognition. And in the last class we have said that how the fuzzy, how the hyper box classifiers can be extended to fuzzy min max neural network, where every middle layer node in the fuzzy min max neural network will compute the fuzzy membership function of an unknown point to a particular hyper box.

Then outputs of all the hyper boxes belonging to the same class they are gathered together in one of the output layer nodes or class nodes. But the class node decides about what is the maximum membership given by the different hyper boxes belonging to different classes. And the membership function to that class is this maximum membership of different hyper boxes belonging to a same class.

So, if I have c number of output neurons pertaining to c different classes, then every c neuron will give me a membership function and this membership function is the membership of the unknown feature vector to the corresponding class. And we said that this itself can be taken as fuzzy classification because this is nothing but a membership vector or if we want this classification or hard classification then whichever neuron gives the maximum membership value we can put the unknown feature vector into the corresponding class.

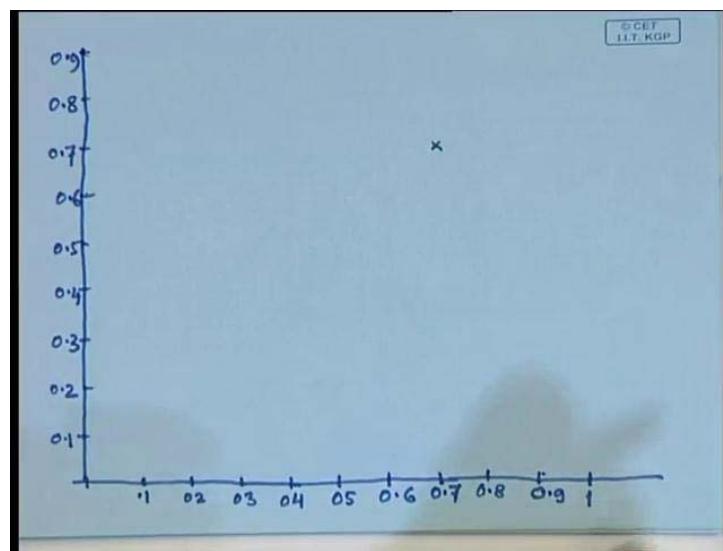
So, we can have a max operator which will give us a binary vector where only one of the components of a binary vector will be equal to 1. And that obviously corresponds to a maximum membership and all other components of the binary vector will be equal to 0. And that is equivalent to our hard classification. So, today what we will do is we will take the same example that we have taken for designing the hyper boxes. And we will take the same example and we will re draw the hyper boxes and simultaneously we will try to design a fuzzy min max neutral network. So, this is nothing but learning or training of the fuzzy min max neutral network. So, the set of points that we had taken in the last class is something like this.

(Refer Slide Time: 02:39)



We had taken a point having the components $(0.7 \ 0.7)$ and we said that this belongs to class 1. And let us simultaneously draw the hyper boxes that we will get out of these points that we have taken, okay?

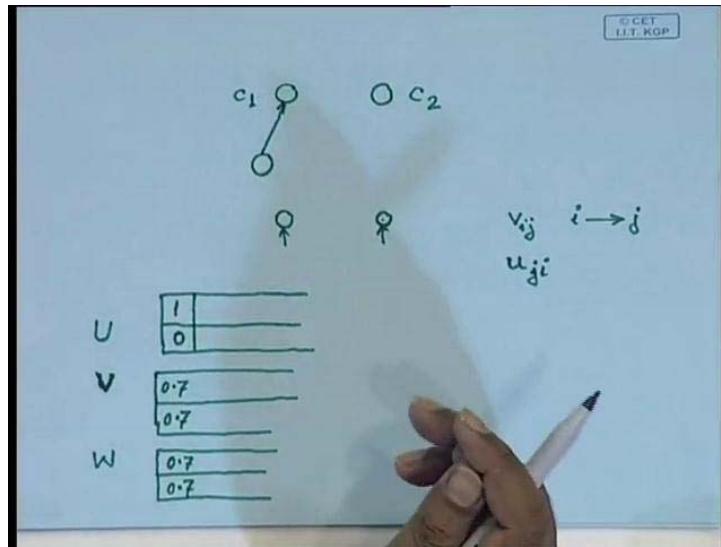
(Refer Slide Time: 03:01)



So, we will do it like this. The lines are very light so I will put it like this is $0.1, 0.2, 0.3, 0.4$. So, when we said that this point, this feature vector $(0.7 \ 0.7)$ this belongs to class 1. So, immediately we said that there will be a hyper box for which the min point and max points are same. So, I will have a hyper box somewhere over here $(0.7 \ 0.7)$. So, at this

location I will have a point hyper box for which min point and max point is same, that is (0.7 0.7). And for this hyper box let us draw, try to generate this fuzzy min max neutral network. So, as we said that I assume that there are two classes, class 1 and class 2.

(Refer Slide Time: 05:24)



So, obviously in the output layer I will have two nodes, one corresponding to class 1 and other one corresponding to class 2. So the first node corresponds to class 1 and the second node corresponds to class 2. Similarly, at the input layer also I we will have two nodes because I am considering feature vectors of dimension 2. So, at the input layer also we have two nodes. So, I have node 1 and input node and this will have the input feature vectors.

And while designing this fuzzy min max neutral network what we have to do is we have to generate three matrices because from the middle layer to the output layer, the connections are actually binary connections, either 0 or 1. 1 means there is a connection from a middle layer node to that corresponding output layer node. 0 means there is no connection from the corresponding middle layer node to the corresponding output layer node. And I will have two matrices, which represents the connections from the input layer nodes to the middle layer nodes. And those two matrices actually represented the min point and max point.

So, for min point I will have a matrix V. For max point I will have a matrix W and for the middle layer node to the output layer node I will have, sorry I will have matrix V and W, not even W. V corresponding to the min points and W corresponding to the max points. And I will have a matrix U which represents the connections from the middle layer nodes to the output layer nodes.

So, if I use this convention that for every node in the input layer matrix V will have two rows corresponding to the min points and the number of columns will actually be variable because till now I do not know what the number of nodes in the middle layer is. So, the number of nodes in the middle layer that will actually represent that how many columns I need in this matrices V and W .

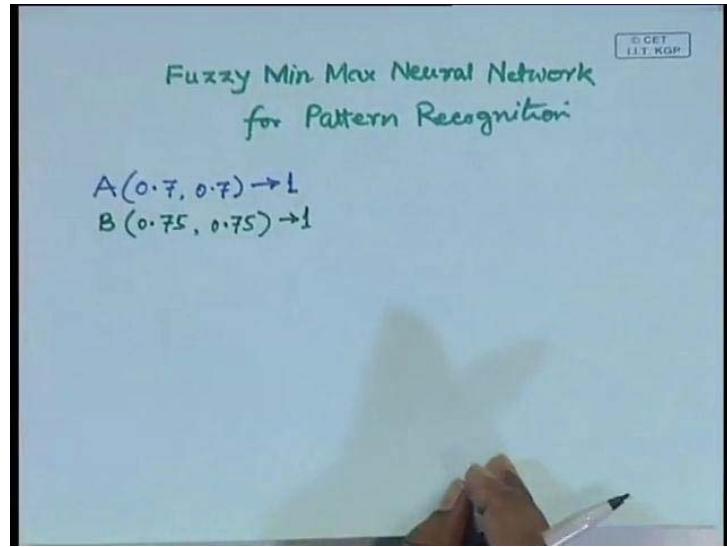
So, matrix V will have two rows. Similarly, matrix W will also have two rows. Similarly, for matrix U also let us assume that we will have two nodes. It depends upon which way I represent the matrix. So, here in matrix U our row will correspond to one of the nodes in the output layer node. The other row we correspond to the other node in the output layer node.

So, from input layer to middle layer a connection V_{ij} will represent a connection from i^{th} node in the input layer to the j^{th} node in the middle layer. Whereas, in matrix U the node, the entry U_{ji} will represent the connection from the j^{th} node in the middle layer to the i^{th} node sorry, a connection from U_{ji} will represent a connection from the i^{th} node in the middle layer to the j^{th} node in the output layer. I just said the reverse, j^{th} node in the middle layer to the i^{th} node in the output layer. So accordingly for every node in output layer I will have a row.

So, in this matrices what I have to go on adding is the number of columns. So, for the first point that you considered, this point is $(0.7, 0.7)$. So, min point and max point that is there same, right? So, accordingly in matrix V , I will have to introduce $(0.7, 0.7)$ because that is the min point. Similarly, matrix W , I have to introduce $(0.7, 0.7)$ because that is also the max point.

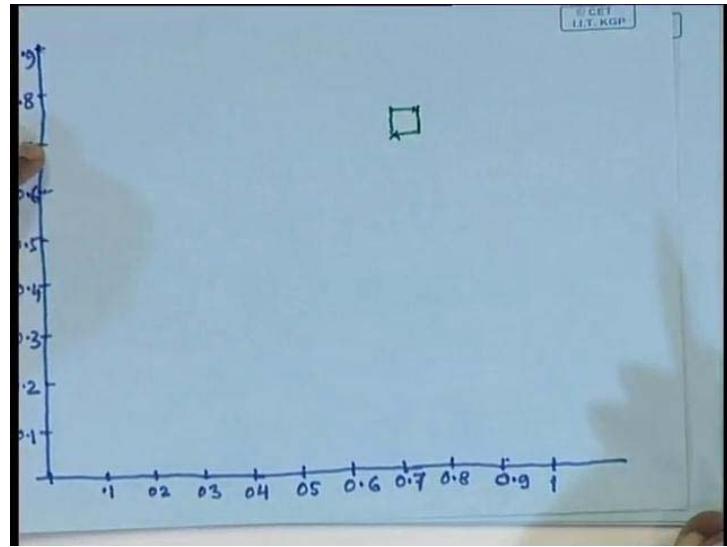
And I have to introduce one node in the middle layer corresponding to this particular feature vector $(0.7, 0.7)$. So, the corresponding entries in matrix V and W , I already established. It is known that this feature vector belongs to class 1. So, as this node belongs to class 1 so the corresponding column in matrix U will be $1 \ 0$ indicating that this node is connected to c1 only. This node is not connected to c2. Similarly, when I consider the second point.

(Refer Slide Time: 11:10)



The second point that I have is point B which is $(0.75, 0.75)$ and this belongs to again class 1, right?

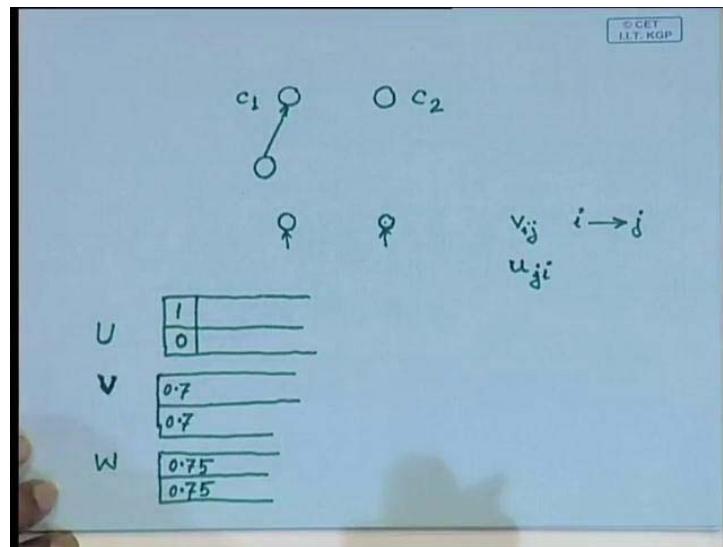
(Refer Slide Time: 11:31)



So, coming over here $(0.75, 0.75)$ is this point. This also belongs to class 1. So, I generate a hyper box. Assuming that this expansion is permitted, I generate a single hyper box containing these two points. So now, find that the earlier min point and max point that we have generated, this min point and max point gets changed and I am not introducing a new node in the middle layer. Only the min and max points I have to change and because I am not introducing any node in the middle layer so the matrix U will remain unchanged, okay?

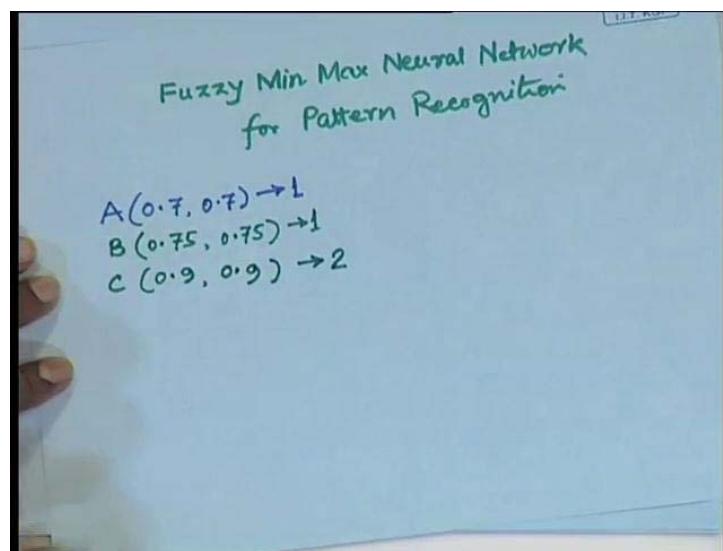
So, what I have to do is over here, the min point is now minimum of these two. In the first dimension the minimum of 0.7 and minimum of 0.75, it remains 0.7. And similarly, minimum of 0.7 and minimum of 0.75, it remains 0.7. Whereas, the max matrix that will be maximum of these two. So, it will be (0.75, 0.75), right? So, the matrix W though initially we had set it (0.7, 0.7).

(Refer Slide Time: 12:41)



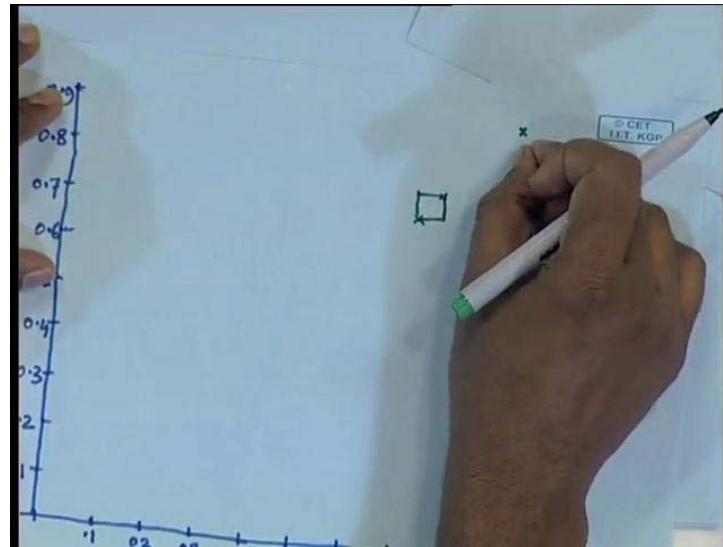
Now, it will be 0.75 and 0.75. So, any time the min point and max point gets changed. I have to change the corresponding entries in matrix V and matrix W. Now, comes the third point.

(Refer Slide Time: 13:02)



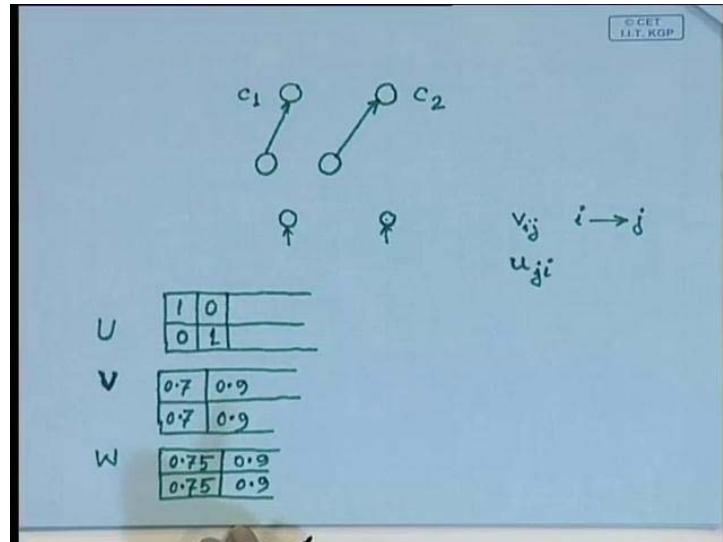
The third point that we considered was point C which is (0.9, 0.9) and we said that this point belongs to class 2. So, I will take (0.9, 0.9) over here. So, 0.9 and 0.9 is this one.

(Refer Slide Time: 13:27)



And this belongs to class c2. So, obviously I cannot expand any of them to include this particular point. And this is the first point, the first feature vector that is considered from class 2. So, naturally I have to introduce a new node in the middle layer, right?

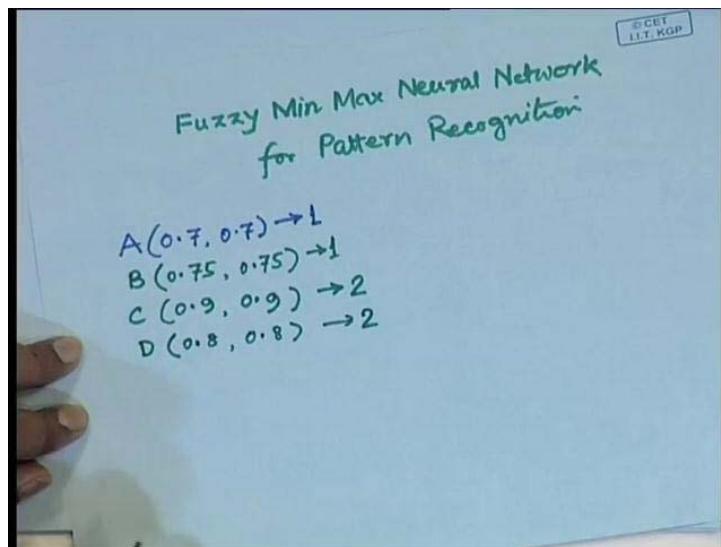
(Refer Slide Time: 13:49)



So, I introduce a new node in the middle layer. And as I know that this hyper box will belong to class c2. So, this will only have a connection from here to c2. It will not have any connection up to c1. So, accordingly the corresponding entity in matrix U will be (0, 1).

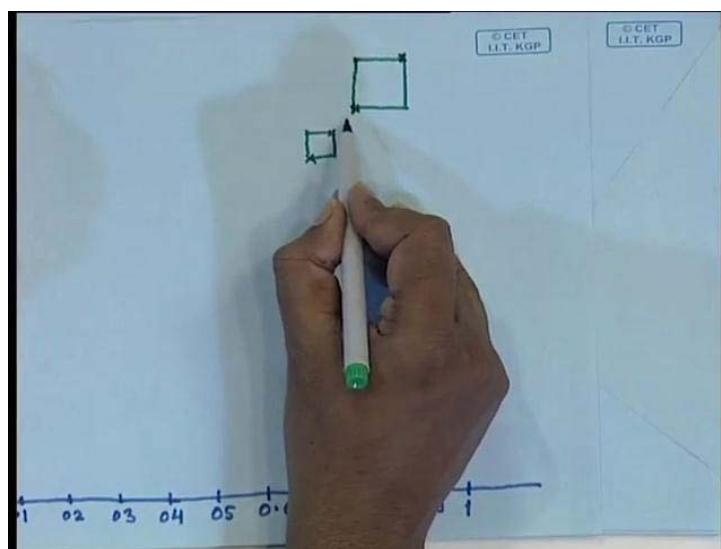
Now, I have to set the entries in the other two matrices, that is V and W. And you find that this is the only feature vector considered so far belonging to class c2. So, naturally its min point and max point will be the same. And min point max point both of them will be (0.9,0.9). Max point will also be (0.9, 0.9).

(Refer Slide Time: 14:54)



Now, I consider the third point where the third point is D and it is (0.8, 0.8) and this also belongs to class 2, right?

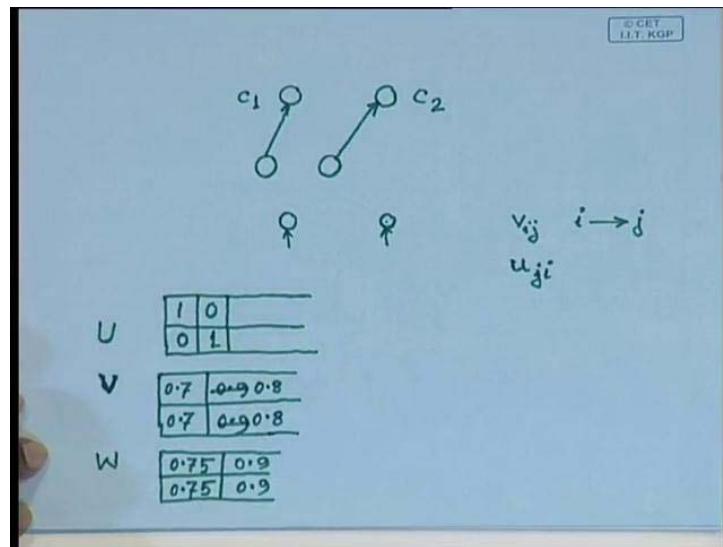
(Refer Slide Time: 15:11)



So, corresponding to this I have (0.8, 0.8) somewhere over here. And suppose I can expand this hyper box to include this also. So the new hyper box that I will get is this one. Now, in

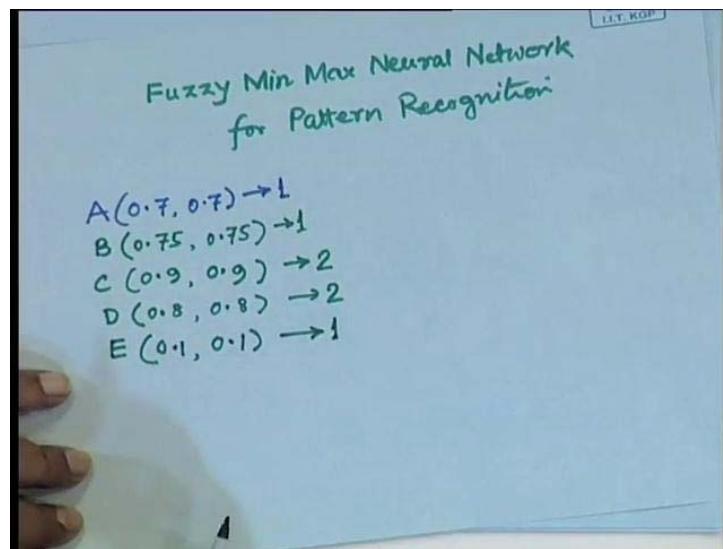
this hyper box we find that the max point will remain the same as the previous hyper box whereas, the min point has to change. So, earlier the min point was (0.9, 0.9). Now, the min point has to be minimum of these two. So, it will now be (0.8, 0.8).

(Refer Slide Time: 15:54)



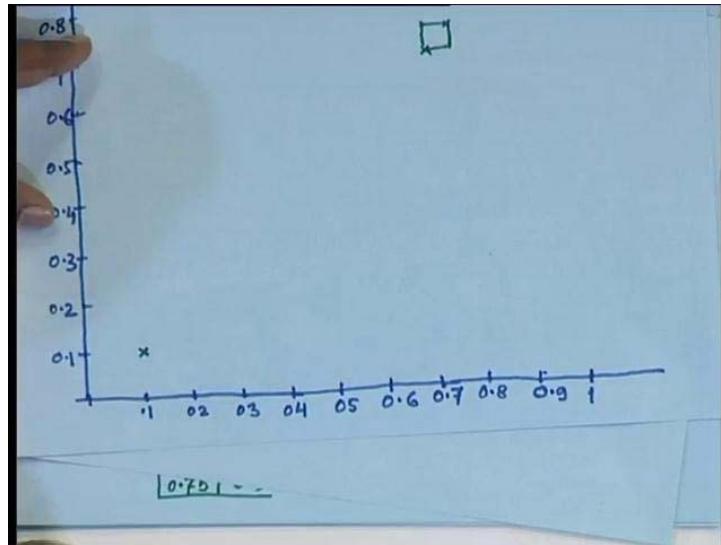
So, earlier this 0.9 0.9, which we had put, this min point is to be 0.8 0.8, okay?

(Refer Slide Time: 16:08)



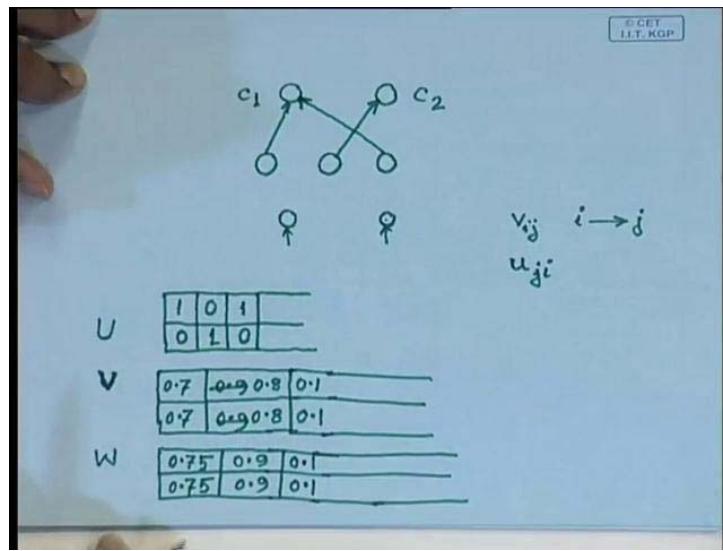
So, now let us take the fifth point which is E and the vector is (0.1, 0.1), and we said that this feature vector belongs to class 1, okay?

(Refer Slide Time: 16:30)



So, for this I will have the hyper box at (0.1, 0.1), right? And suppose that this hyper box belonging to class 1, I cannot expand to include this. So, I have to generate a new node for this hyper box and that new node have to generate over here.

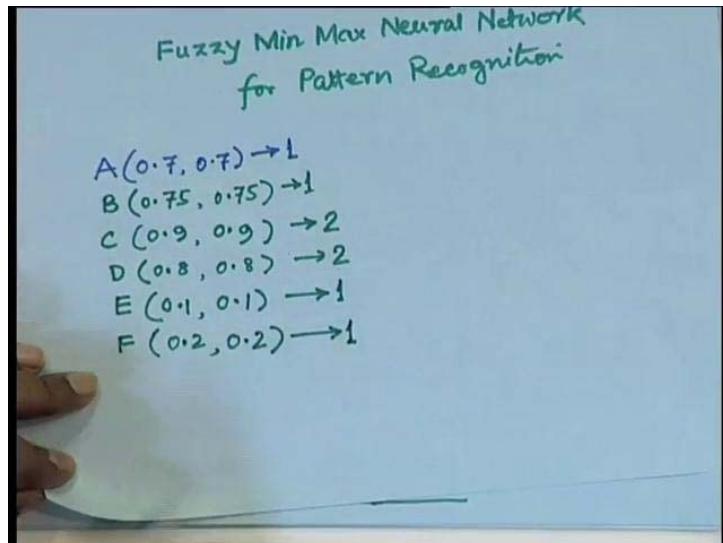
(Refer Slide Time: 16:50)



And as I know that this belongs to class c1. So, the corresponding entry in my matrix U has to be (1, 0) indicating that this node belongs to class c1, right? And then I have to make an entry into the matrix V. I have also have to make the entry into the matrix W. And here again because this is a point hyper box so the min point and max point they will remain the same,

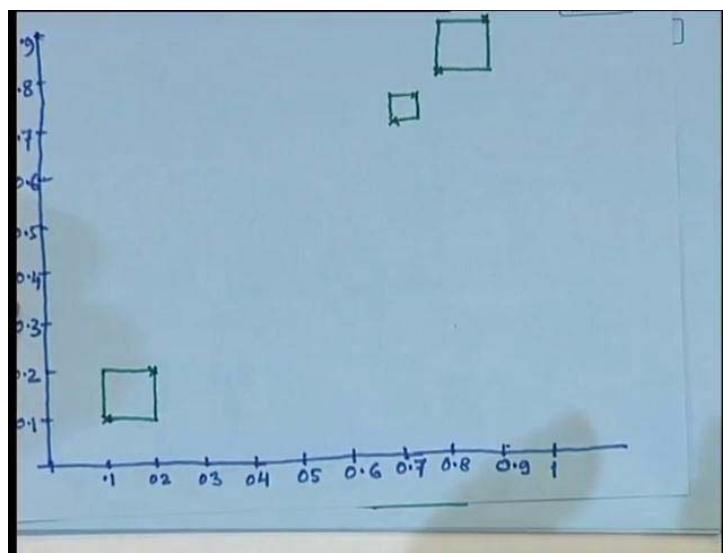
that is $(0.1, 0.1)$. Here also it is $(0.1, 0.1)$. Is that okay? So, then let us consider the other point that is point F.

(Refer Slide Time: 17:56)



Whose coordinates are $(0.2, 0.2)$ and this point also belongs to class 1, right?

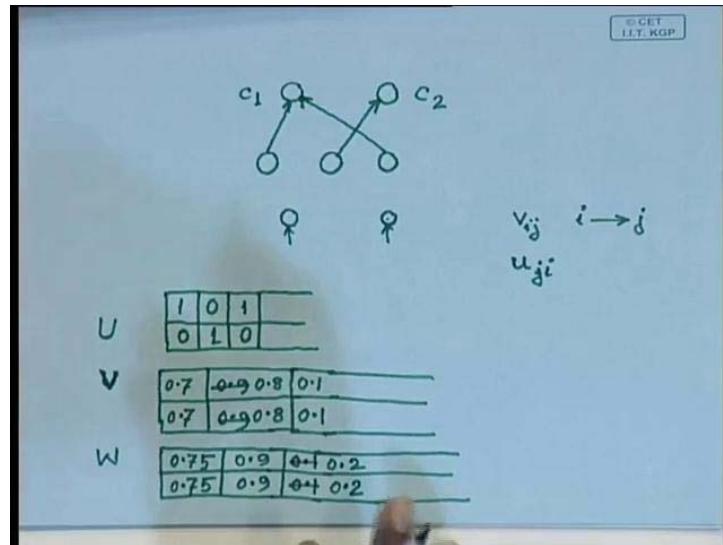
(Refer Slide Time: 18:17)



So, over here this corresponds to $(0.2, 0.2)$ a point somewhere over here. And assuming that this $(0.1, 0.1)$ can be expanded to include this point. I get a single hyper box, and this single hyper box is going to class c2 and because I am not adding any new hyper box so I do not have to add any node. The previous node that I have only in that node I have to change the min point max point. So, accordingly I do not have to make any entry into matrix U. In

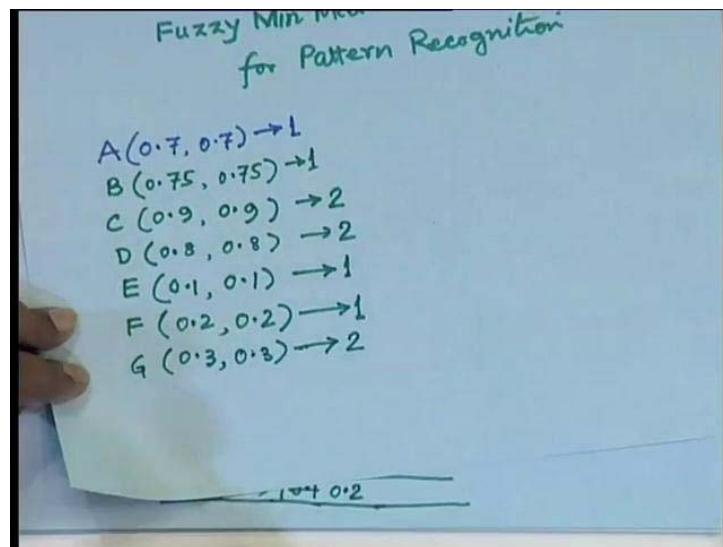
matrix V and W change the min point and max point. So, the new min point that remains minimum of these two. So, the min point does not change but max point has to be maximum of these two.

(Refer Slide Time: 19:18)



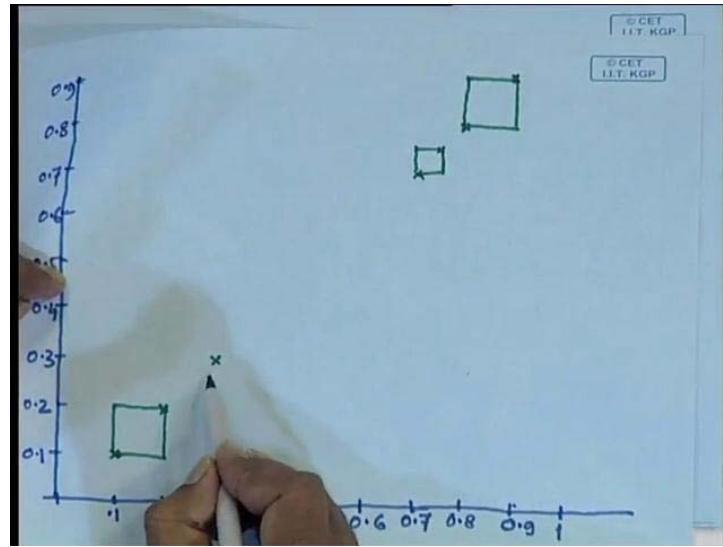
So, the max point has to be changed to $(0.2, 0.2)$ from $(0.1, 0.1)$. So, the new max point that I get, earlier it was $(0.1, 0.1)$ now I have to make it $(0.2, 0.2)$. The other matrices remain the same.

(Refer Slide Time: 19:35)



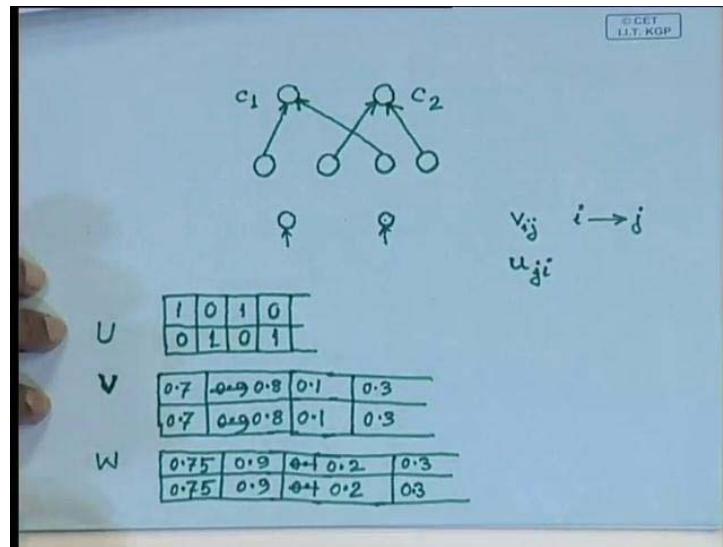
Then let us take the next point, that is G where the vector is $(0.3, 0.3)$ and this belongs to class 2.

(Refer Slide Time: 19:52)



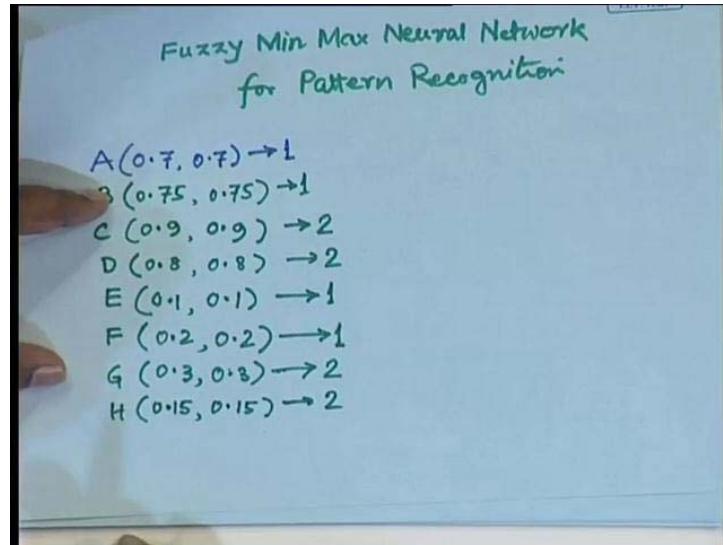
So, (0.3, 0.3) if I put over here (0.3, 0.3) comes at this location. So, you find that this belongs to class c2. The earlier hyper box for c2 was this. Assuming that this cannot be expanded into this, so I have to add a new node in the middle layer, okay?

(Refer Slide Time: 20:13)



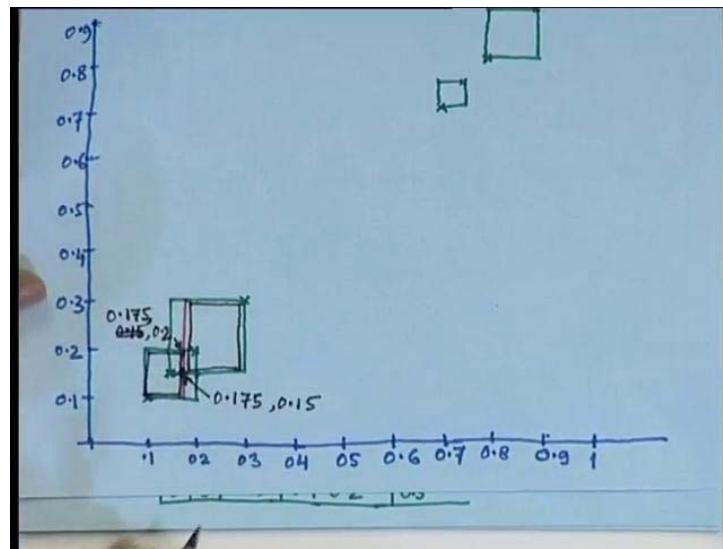
So, the new node in the middle layer comes somewhere over here. And I know that this belongs to class c2. So, it will only have a connection to c2. The classifying neurons c2 and accordingly the corresponding entity in the U matrix will be (0, 1). So, this is U matrix. Now, this is a new node and a point hyper box. So, I will have the min point and max point which is same as (0.3, 0.3). Max point we will also be (0.3, 0.3), the same min point max point.

(Refer Slide Time: 21:07)



Then you come to the next point which is $(0.15, 0.15)$ and this also belongs to class 2.

(Refer Slide Time: 21:23)



coming over here $(0.15, 0.15)$ is this, right? And assuming that I can expand this to include this one. So, the new hyper box that I will generate is this one. So, this becomes my new hyper box. And now we find that, this one is hyper box belonging to class c2. This is hyper box belonging to class c1.

So, whenever I have such a situation, because there is an overlap I have to break this two hyper boxes. And to break these two hyper boxes I have to find out that in which

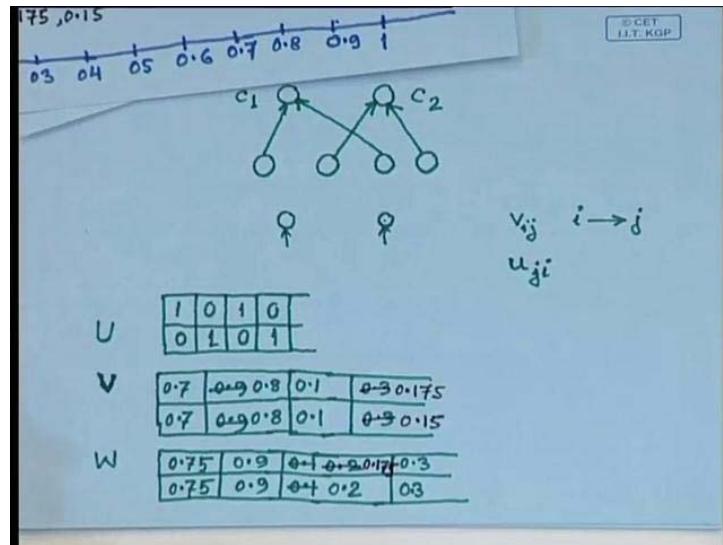
dimension the overlapping is minimum. So, here you find that along the horizontal dimension the overlap is, this is 0.15 and this is 0.2. So, the overlapping is 0.05. In this dimension it is 0.15. This is 0.2. So, the overlapping is 0.05.

So, in both the dimensions the amount of overlap is same. So, I can break this hyper box in any of the two dimensions. Let us assume that we break the hyper box along the middle of the first dimension over here. So, when I break this I get two hyper boxes. One hyper box is this one which belongs to class c1 and the other hyper boxes is this one which belongs to class c2, right?

Now, over here you find that when I am breaking these two hyper boxes, I am not generating any new hyper box. There was a hyper box corresponding to this and there was a point hyper box corresponding to this. So, it is the point hyper box which has been expanded to include the second point. And while it was expanded there was an overlap because of this overlap I had to contact this hyper box. So, what I have to do is I have to change the min point max point accordingly for this hyper box which was a point hyper box earlier. I have to change the min point and max point for this which hyper box was already existing, right?

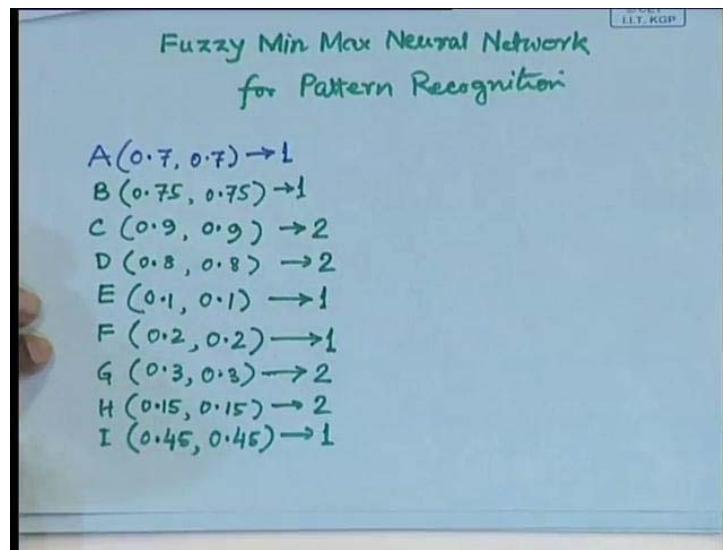
So, what will be the min point max point? For this hyper box the min point remains the same, right? What is the max point? Max point is this coordinate, right? And what is this coordinate? This coordinate we be 0.15 and 0.2, sorry max point, max point for this hyper box will be (0.15, 0.2), right? So, this is (0.15, 0.2) and that is the max hyper box. Max point of this hyper box, the modified max point, similarly the min point of this hyper box which is nothing but this location. This can be 0.15 and 0.15. Earlier it was 0.15 now it will be 0.175, 0.175 and this will remain same as 0.15, right? And this is 0.175 and 0.2, 0.175 and 0.2. So, the max point of this hyper box is going to change and for this hyper box earlier I had a point hyper box for which the min point and max point was same. Now, the new min point will be this. Max point will remain unchanged. Is it okay? So, coming to our V matrix and W matrix you find that for the earlier one where the min point was (0.1, 0.1) and max point was (0.2, 0.2). Now, the new max point is (0.175, 0.2).

(Refer Slide Time: 26:07)



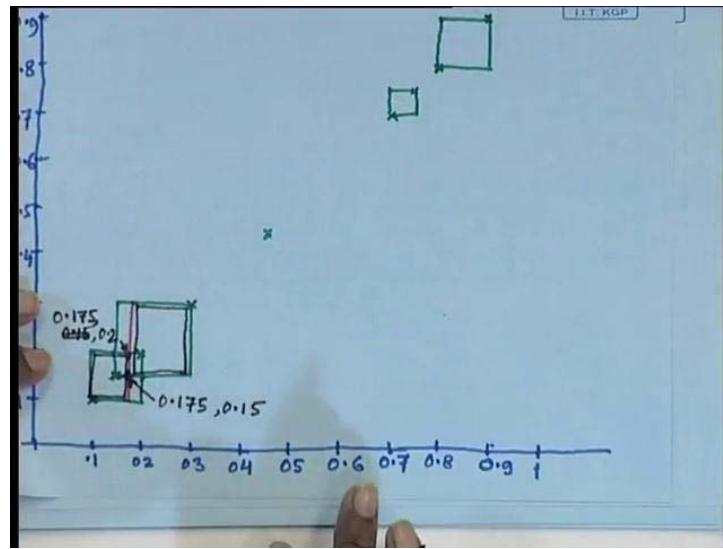
So, this 0.2 now it gets replaced by 0.175. The other point, the other dimensions remains as 0.2, right? For this one (0.3, 0.3) which was a point hyper box. For this point hyper box the max point remains the same. But the min point becomes 0.175 and 0.15, is that okay? So, the min point will be for this will be changed to 0.175 and 0.15, is it okay?

(Refer Slide Time: 27:05)



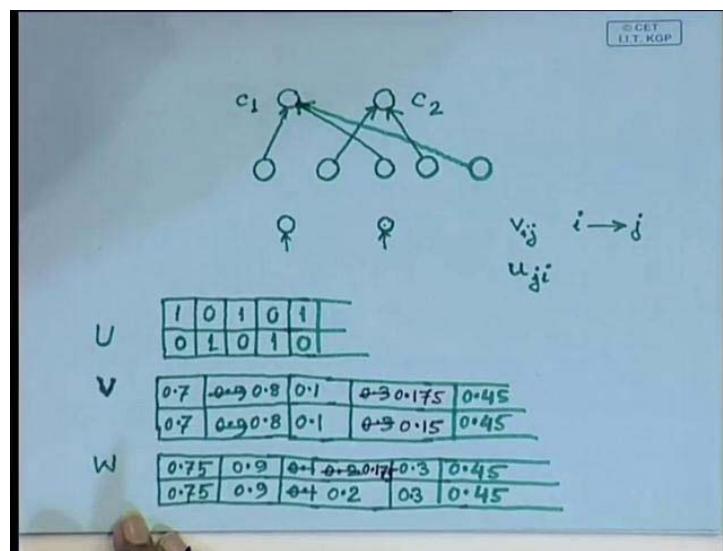
Now, let us take the next point. The next point is I for which vector is (0.45, 0.45) and this belongs to class 1, okay?

(Refer Slide Time: 27:27)



So, for this I have 0.45 and 0.45. This belongs to class 1, and suppose this cannot be included in any of the hyper boxes after expansion. So, I have to make a new node for this particular point whose corresponding vector is (0.45, 0.45).

(Refer Slide Time: 27:48)



So, I generate a new node and I know that this node belongs to class c1. So, accordingly I have to make a connection from this node to class c1 and correspondingly matrix U will be modified as (1, 0). And I have to enter the min point and max points in this two matrices. And because it is a point hyper box so the min point and max point will remain the same. So, both of them are (0.45, 0.45). Here also it is 0.45, here also it is 0.45, okay?

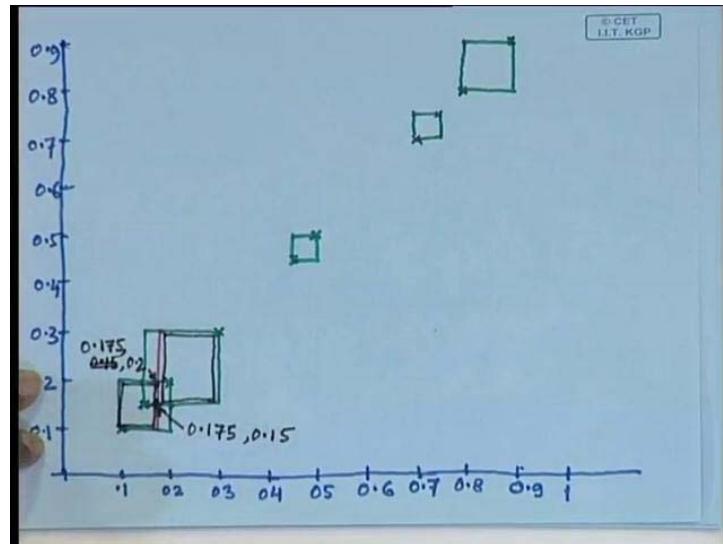
(Refer Slide Time: 28:50)

for Pattern Rec.	
A	(0.7, 0.7) \rightarrow 1
B	(0.75, 0.75) \rightarrow 1
C	(0.9, 0.9) \rightarrow 2
D	(0.8, 0.8) \rightarrow 2
E	(0.1, 0.1) \rightarrow 1
F	(0.2, 0.2) \rightarrow 1
G	(0.3, 0.3) \rightarrow 2
H	(0.15, 0.15) \rightarrow 2
I	(0.45, 0.45) \rightarrow 1
J	(0.5, 0.5) \rightarrow 1

0.75	0.9	0.1	0.175	0.3	0.45
0.75	0.9	0.1	0.2	0.3	0.45

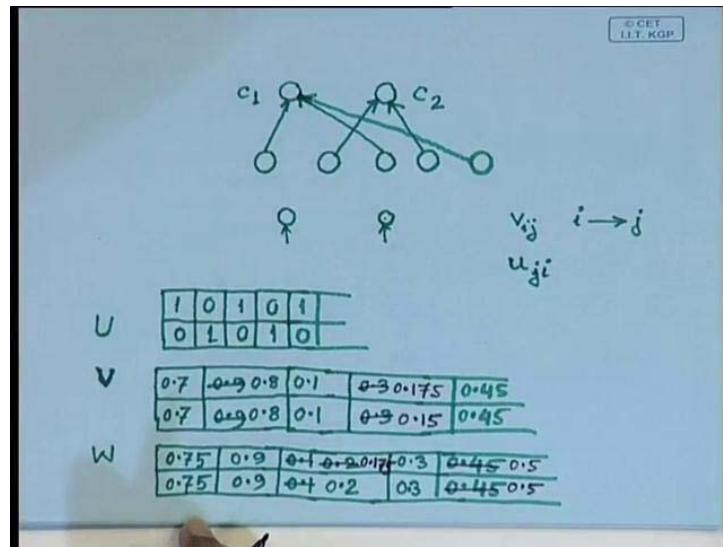
So, when I consider the next point that is point J corresponding to this the feature vector is (0.5, 0.5) and this also belongs to class 1. If I come over here (0.5, 0.5), that is this, okay?

(Refer Slide Time: 29:09)



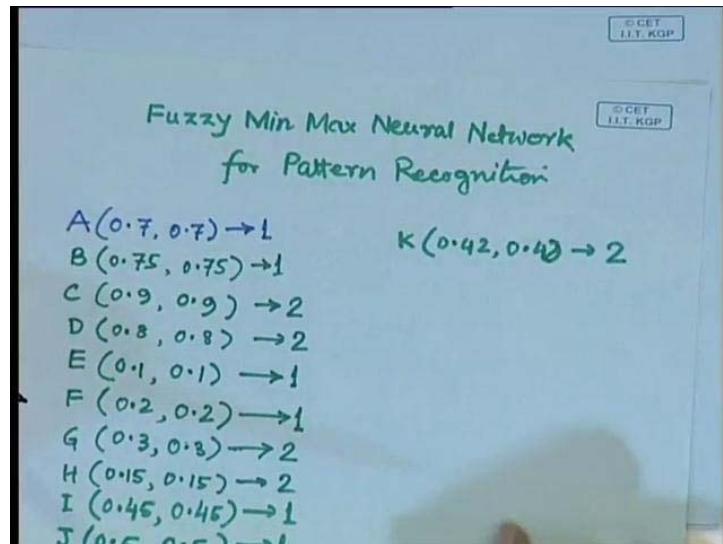
And assuming that this point hyper box can be expanded to include this. So, I generate a new hyper box and because I am not adding any new node because the previous point hyper box has been expanded to include this. So, U matrix will remain the same. However, I have to look at the V matrix and W matrix. And here you find that the max point that is being changed. So, earlier max point was (0.45, 0.45). Now, the new max point is (0.5, 0.5), okay?

(Refer Slide Time: 29:48)



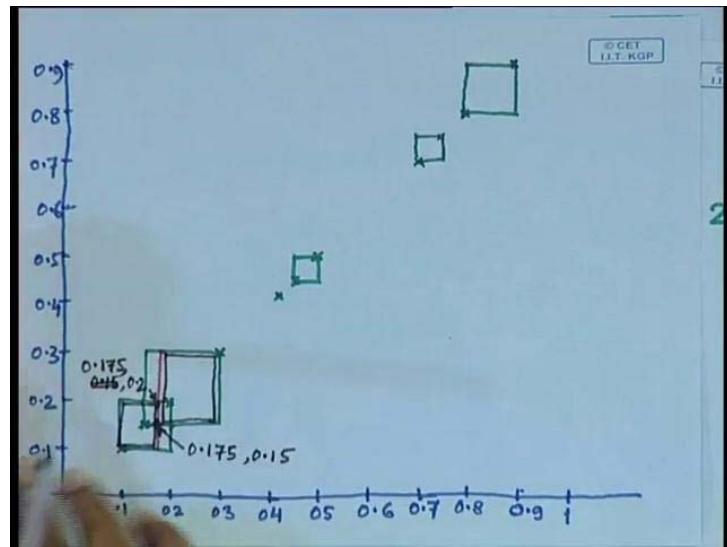
So, I have to change this corresponding entry in matrix W. So, it will be (0.5, 0.5), okay?
Then let us consider the next point, the next point is K.

(Refer Slide Time: 30:07)



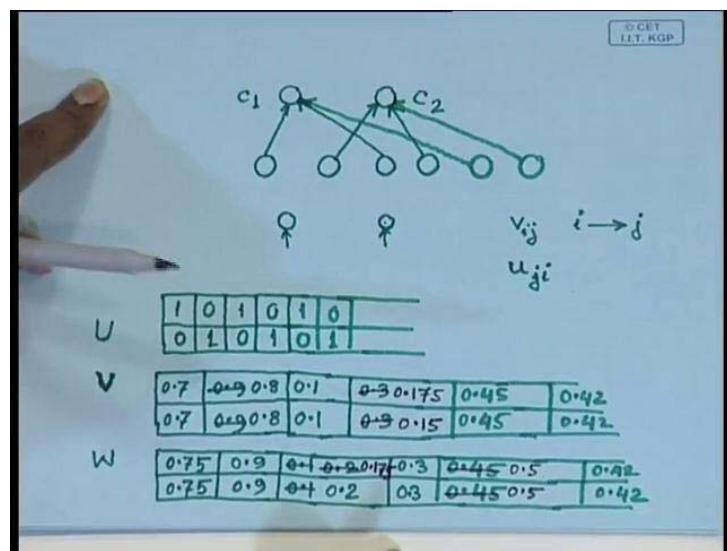
The feature vector is (0.42, 0.42) and this belongs to class 2. So, I come over here. This will be somewhere over here.

(Refer Slide Time: 30:35)



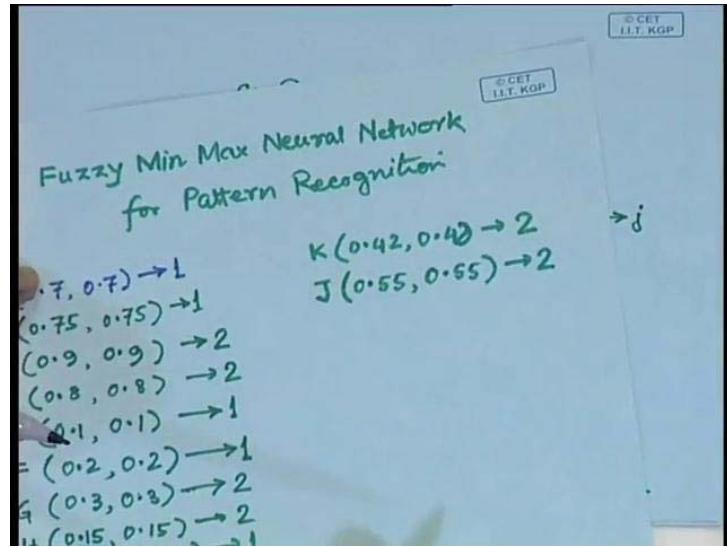
Somewhere over here (0.42, 0.42) and assuming that this has to be point hyper box I have to have a corresponding node in the middle layer, okay?

(Refer Slide Time: 30:51)



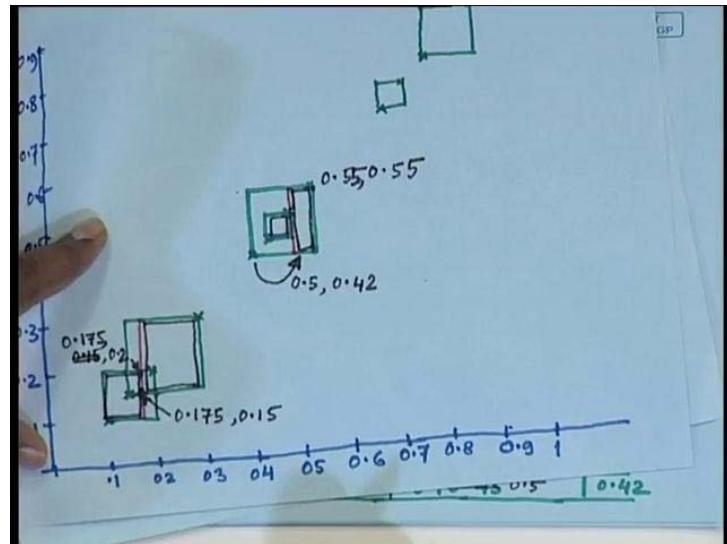
So, have to have a node in the middle layer and as I know that this node belongs to class c_2 . So, I have to change the corresponding U matrix. So, I have to make a column, add a column in the U matrix which will be 0 1 because this belongs to class c_2 . And in the U matrix, V matrix I have to generate, I have to add a column corresponding to this particular point, so this 0.42, 0.42. Here also it is 0.42, 0.42. Then I take the next point.

(Refer Slide Time: 31:43)



The next point is $(0.55, 0.55)$ and this belongs to class 2. So, according to this I have over here $(0.55, 0.55)$ somewhere over here.

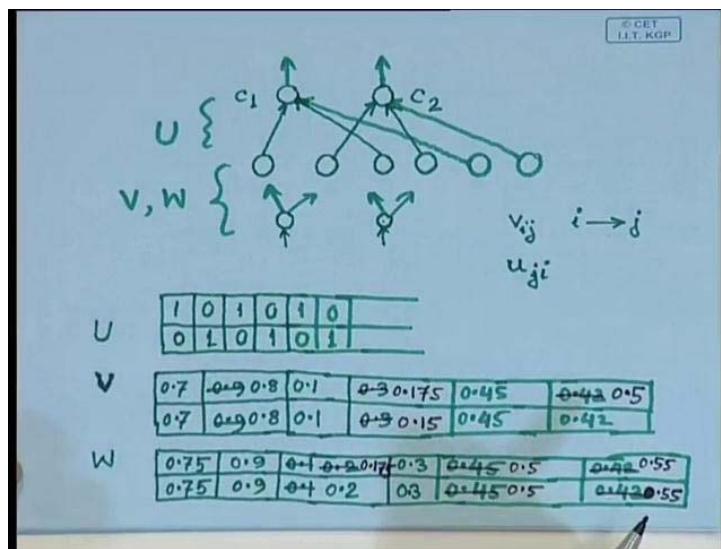
(Refer Slide Time: 32:02)



And assuming that this can be expanded to include this I have this new hyper box, the modified hyper box which comes like this and now you find that there is a containment. So, because there is a containment I had to contract this hyper box. So, this new hyper box will be contracted. Let us assume that we contract it from this location. So, in this side I have one hyper box belonging to class c 1 and on this side I have another hyper box belonging to class c 2. So, over here you find that this hyper box was already existing and for this hyper box and min max point has not been changed. So, accordingly the earlier

entry in our matrix V and W matrix that remains the same, that is $(0.45, 0.45), (0.5, 0.5)$. This remains unchanged, right? Whereas, for this new hyper box I had earlier the min point, which is $(0.42, 0.42)$ that corresponds to this particular point. And when I have expanded this to include the second point and because of this expansion there was a containment and due to this containment I had to contract this hyper box. So, as I contracted this hyper box the min point has now changed from here to here, right? And this is my max point. So, what is this new min point? The new point is $(0.5, 0.42)$ that is the new min point. And the max point is $(0.55, 0.55)$. So, accordingly I had to make the modification over here.

(Refer Slide Time: 34:40)



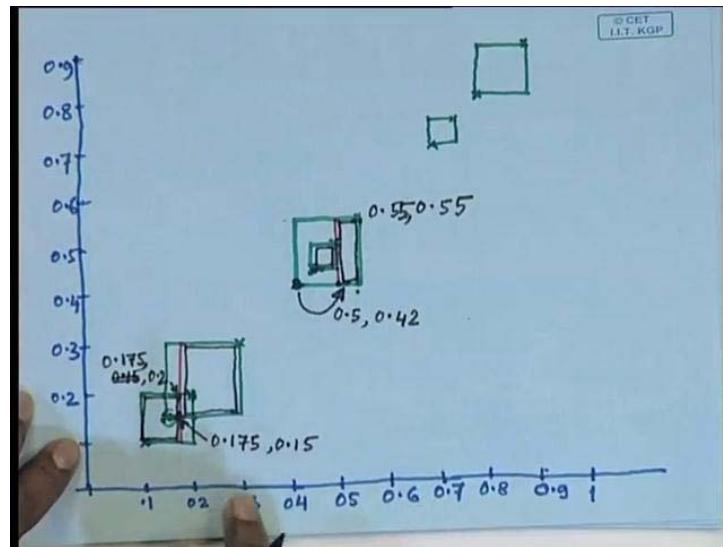
Now, the min point becomes $(0.5, 0.42)$. So, this will be $(0.5, 0.42)$ as it is and the max point which earlier was $(0.42, 0.42)$ because it was a point hyper box. Now the new max point will be 0.55 and 0.55 , that is this one. So, this max point, new max point becomes 0.55 and 0.55 . So, with this I have included all the training points that were given. So, at the end of the training, the three matrices that I have generated are like this. So, I have this U matrix which represents connections from the middle layer nodes to the output layer nodes. And I have two matrices V and W corresponding to min points and max points of the nodes which are generated in the middle layer. So, over here these are represented by the matrices V and W. And over here the connection is presented by matrix U and I have the final memberships computed to the two different classes c_1 and c_2 which comes from the output layer nodes, is that okay?

So, we find that your network has been trained in a single pass unlike in case of multi-layer perceptron or single layer perceptron. What we have done is, we have changed the

weight vector or the weight matrices iteratively every time a given feature vector becomes misclassified then following the gradient descent approach or following the back propagation learning we had to change the weight matrices for different layers. And once it is changed I have to re-check with all the samples all the feature vectors which was used to train the neural network before this. That means I have to come to conclusion that the multilayer perceptron or single layer perceptron has been trained properly only when in a single pass all the samples are correctly classified, all the training samples are correctly classified or the error that you get is less than certain specified value or less than the tolerable value so the training algorithm of the neural network in such cases was an iterative algorithm. Whereas, in this case the neural network is trained in a single pass because every time.

And what I am doing is if there is an overlap immediately contacting the existing hyper box, which takes care of the fact that there is no overlap or no ambiguity during classification or during testing, however as we said before that the problem with this fuzzy min max neutral network is that as we are going for contraction.

(Refer Slide Time: 38:07)



So, here to find that say this point which we know that these belongs to class c2. But because of overlap and contraction finally this point has been classified into class c1. Because now if I after this contraction operation if I present the same point to this neural network, the neural network will say that this point belongs to class c1, though I know that this point belongs to class c2. Similarly, for this I know that this point belongs to class c1. But after creation of this hyper boxes and contraction of these hyper boxes, if I present the same point to the

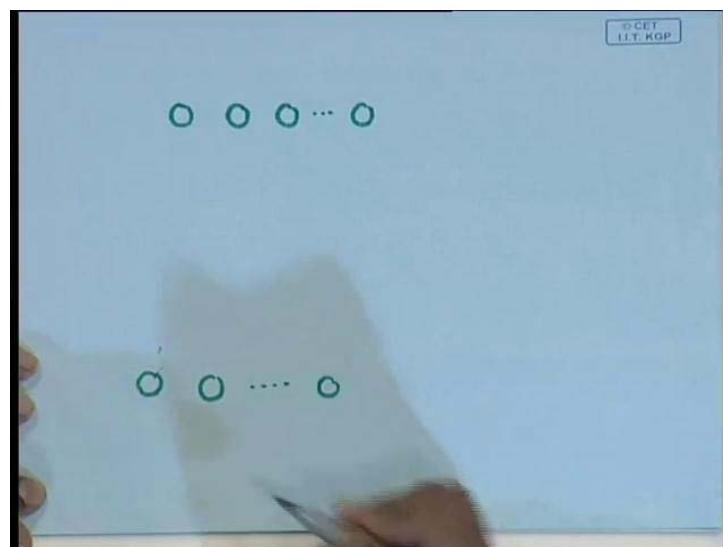
neural network it will say that this black point belongs to c2. Though I know that this point belongs to class c1, okay?

So, that is the disadvantages of this kind of neural network. And such a kind of problem you get both over here and over here. Because earlier we know that this point belongs to class c1 but after creation of this hyper boxes and contraction. Now, if I present this point to the neural network, what it will do? For this point, it will try to calculate the fuzzy membership function to this hyper box and also the fuzzy membership function to this hyper box. And now you find that this point is nearer to this hyper box, then this hyper box. So, the fuzzy membership function for this hyper box will be more than the fuzzy membership function for this hyper box.

So, naturally if I go for hard classification. This point will be classified to class c1, though you know that it actually belongs to class c2. So, after this expansion and contraction process you say that the training samples which are used to train the classified, train the neural network those samples themselves will be misclassified or some of those samples themselves will be misclassified.

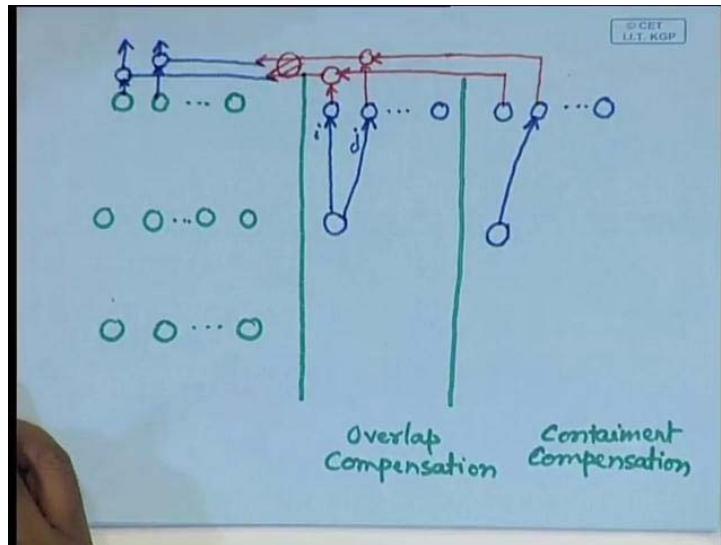
So, naturally due to this contraction we are introducing some amount of error in our learning process or in training process. So we have suggested some modification to this kind of algorithm, that is by introducing the concept of compensatory neurons. So, what we have done is the basic structure or the classified section of this neural network called reflex neural network remains the same as what has been suggested by Patrick Simson, that is fuzzy min max neural network, okay?

(Refer Slide Time: 40:44)



So, in the new architecture I have as before a number of classifying neurons in the output layer. A number of input layer neurons. Let me draw in another paper.

(Refer Slide Time: 41:12)



So, I have a number of classification neurons. I have a number of input layer neurons. Output layer nodes is same as the number of classes we are considering. The number of input layer neurons is same as the dimensionality of the feature vector that we are considering. So, the major part of this reflex fuzzy neural network is same as the fuzzy min max neural network which has been suggested by Simson.

And we have this middle layer nodes which are basically hyper boxes. And as we have just seen that these number of nodes in the output layer and the number of nodes in the input layer they are fixed by the problem. Whereas, number of nodes in the middle layer that we create during the training process and accordingly we generate matrix U, matrix V and matrix W.

Now, what we have suggested is, we have suggested two more sections in this neural network which is called the compensation section, and in compensation section again as we have just seen that we can have two types of situations. One is overlap situation and the other one is containment situation and the kind of compensation that you have to impart is different in case of overlap compensation we have to give some sort of compensation.

In case of containment we have to give some other type of compensation. So, accordingly this compensation section that we have generated, this compensation section is again divided into two parts. One of them we are calling as overlap compensation, the

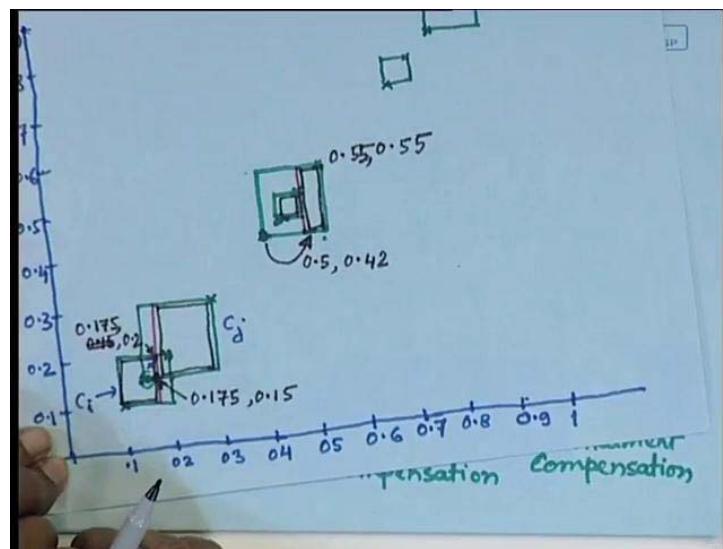
other one is called containment compensation. So, I will have some additional nodes. One in the overlap compensation section. This we call as overlap compensation and the other one is containment compensation.

So, what is done in this overlap compensation and containment compensation? Whenever there is an overlap like here while training if we find that there is overlap in the previous case, what we have done is we have contacted the hyper boxes to remove that problem. Now, we do not go for contraction. But we go for the gradation of the membership function within that overlap region. So, whenever there is overlap one new node is introduced in the overlap sections.

So, over here in the overlap section we will introduce a new node and this new node will give one compensation to the classifying neurons. Corresponding to this hyper box and another compensation to the classifying neurons. Corresponding to this hyper box because I cannot grade the membership of only one hyper box. I have to grade the membership of both the hyper boxes. Because if a point falls within this region, the overlap region normally our classifying neurons. In this section, classification section both of them will give membership function to this class and membership function of one to this class, okay?

I have to modify both of them or compensate both of them. So, this new neuron that we are introducing in the overlap compensation section that will actually generate two outputs. One output is meant for compensation of one class and the other output is for compensation of the other class.

(Refer Slide Time: 45:36)



So, if I have class c_i and class c_j , the hyper boxes belonging to c_i and the hyper boxes belonging to class c_j there they overlap. Then output of the compensation neuron that will give compensation to both c_i and c_j and however the amount of compensation to c_i and the amount of compensation to c_j will be different. The reason is if this point is nearer to this point then the membership function to c_i to c_j should be more. And if this point is towards this, then the membership function to plus c_i should be more, okay?

So, the amount of compensation that you give to plus c_i and the amount of compensation that you give to plus c_2 or c_j , they cannot be same. They have to be different and while calculating this compensation values you have to take care that, to which of the points your unknown point is nearer. So, what you are doing is, in this section also I will have a number of classifying neurons or compensation neurons which is same as the number of classes that I have. This will get input from the input layer neurons and output of this will actually go to two of the output layer neurons because I have to compensate for i^{th} class as well as the j^{th} , okay?

Similarly, in the containment section also I will also have a number of output layer neurons which is same as the number of classes I have. Here again whenever there is a containment like this I will put a containment compensation neuron. Now, why I said that the amount that the type of compensation that have to give in this case is different from the type of compensation that you have to give in this case is that, here I said that I have to compensate the outputs, the membership functions of both class c_i and class c_j , right?

Whereas, in this case if any feature point lies over here we are saying that it should get a membership value of 1 to this class. Whereas, the membership value to this class has to be 0. And my hyper box in the classifying section that itself will compute membership value of 1 to this class. It will also get a membership value 1 to this class.

I have to make the membership value to this class equal to 0. And the membership value to this class should remain unchanged. That means I have to only compensate the membership value of one class which is containing the hyper box of another class. So, the hyper box that is contained, the membership value of that should not be changed. However, the hyper box which contains the other hyper box, the membership value of the class of the hyper box that contains that should be made equal to 0. Is that okay? So, accordingly output of a compensation neuron in the containment section should go to only one class, that is the class of this particular hyper box.

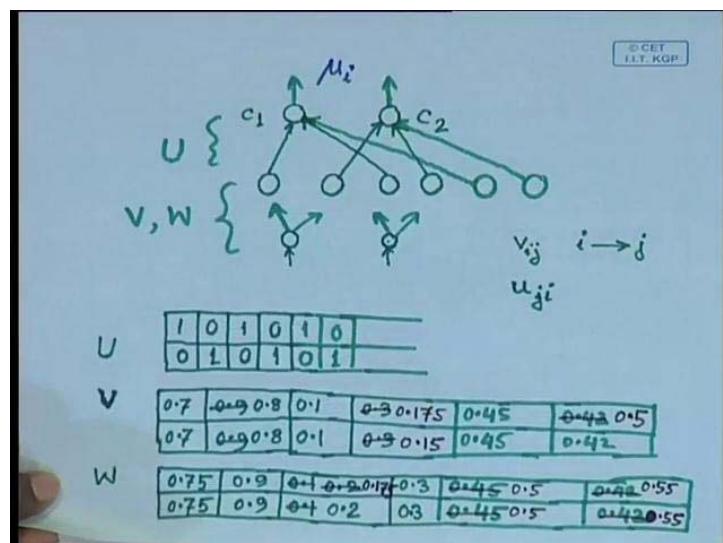
It should not go to the membership function. It should not compensate the membership function of this class. So, here in the overlap compensation section the output of this compensating neuron goes to two classes. Whereas, in this case the output of the containment compensation neuron will go to only one class, right? So, I get two types of compensation; one from the overlap compensation neurons and the other one for the containment compensation neurons. Now, I have to combine these two compensations.

So, I have to have a combiner, right? So, over here, that belongs to the class of smaller hyper box, because that defines its class membership more vigorously than the class membership of the other one. So, I will have to combine the outputs of this compensation neurons. So, I have to have a number of combiner, okay?

So, I will combine the output of this and the output of this. So, these are outputs of the output layer neurons of the compensation class belonging to the same class. So, this is class c1. This is class c1 you combine these two. Similarly, class c2, class c2 you combine these two. Finally, after combination combining these outputs I know that what is the overall compensation that I had to give to the compared to the membership values generated by these neurons. These neurons are generating the membership value as if there is no containment or there is no overlap. We are trying to modify these outputs, these membership values as computed by this compensation values, okay?

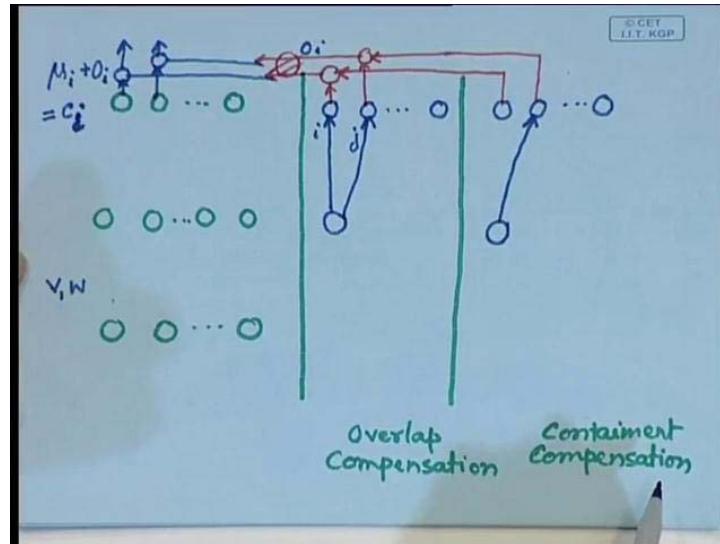
So, what I have to do is, I have to add these values to these membership values. So, this will come over here, this will come over here and then finally I get the final membership value. So, we find that in case of this neural network without any compensation section if the membership value to a particular class say i^{th} class was μ_i .

(Refer Slide Time: 52:29)



In this case the membership value after this compensation to i^{th} class will be μ_i plus let us put it as o_i , where o_i is the combination of the outputs of the overlap compensation neuron and outputs of the compensation neuron.

(Refer Slide Time: 52:43)



So, over here I will have this output as o_i , okay? So, the final membership value will be μ_i . μ_i is what is computed by these neurons. I have to modify this with the outputs of the compensation neurons so the final membership value becomes $\mu_i + o_i$. And that I can take as the final membership value say C_i .

Two class to the i^{th} class for any of the unknown feature vectors which have been put up in the neuron. So, when these compensation section neurons whether it is overlap compensation or containment compensation, they compute the compensation values they will make use of the same V and W matrices, the min points and max points that we have generated for these hyper boxes.

So, they will make use of the same min point and max point to compute what should be the compensation given. And this composite compensation computed only when an unknown feature vector falls within either the overlap region or the contract region. This will not be computed if the unknown feature vector falls outside the overlap region or falls outside the containment region. And when it gets, the final membership value your classification will depend upon this final membership value. So, the classification instead of doing based on μ_i it will be based on c_i , okay?

So, in the next class we will see the details of the membership computation or the compensation computation as done by this overlap compensation neurons and the complete containment compensation neurons. So, we will take the same example to design our reflex for the min max neural networks with, I will take the same example because I have the case of overlap I also have a case of containment. So, with that we will illustrate that how this neural network with compensation neurons that can also be generated. So, we will stop here today. Thank you.

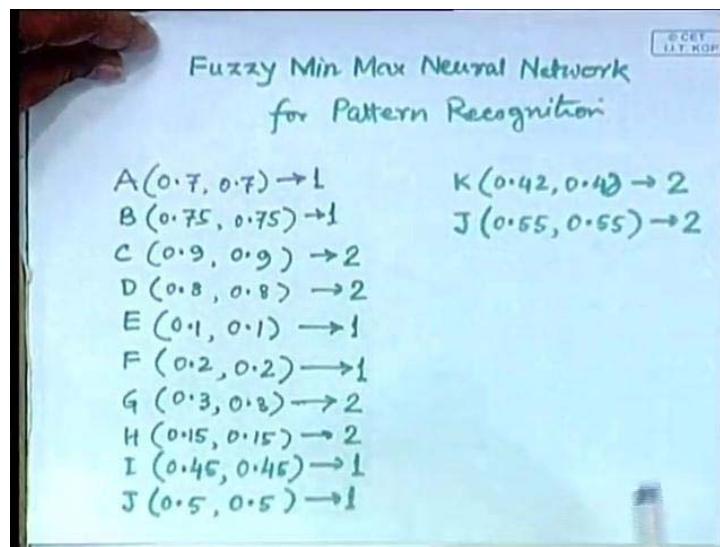
Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 33
Reflex Fuzzy Min Max Neural Network

Today, we are going to discuss about the construction of reflex fuzzy min max neural network, which makes use of the compensatory neurons. So, in the last class we have said or we have seen that how to construct a fuzzy min max network. where there is no concept of the reflex mechanism or there is no concept of compensation to the membership functions computed by the different nodes or the different neurons, which represent the different hyper boxes.

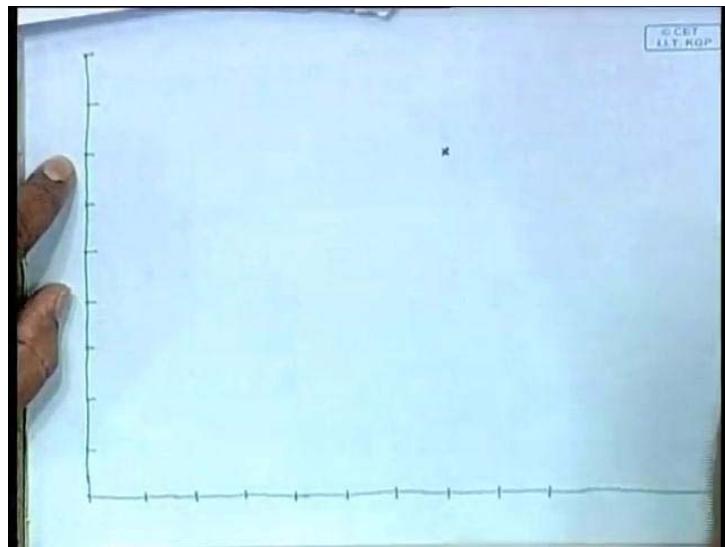
So, I will take the same example as in that example we had two cases; one of the overlap of hyper boxes and one of the containment of hyper boxes belonging to different classes. And see that without modifying the min max points of the classifying neurons, how we can add the different types of compensation neurons to have a reflex fuzzy min max neural networks. So, I will take the same example that we have taken in the last class.

(Refer Slide Time: 01:24)



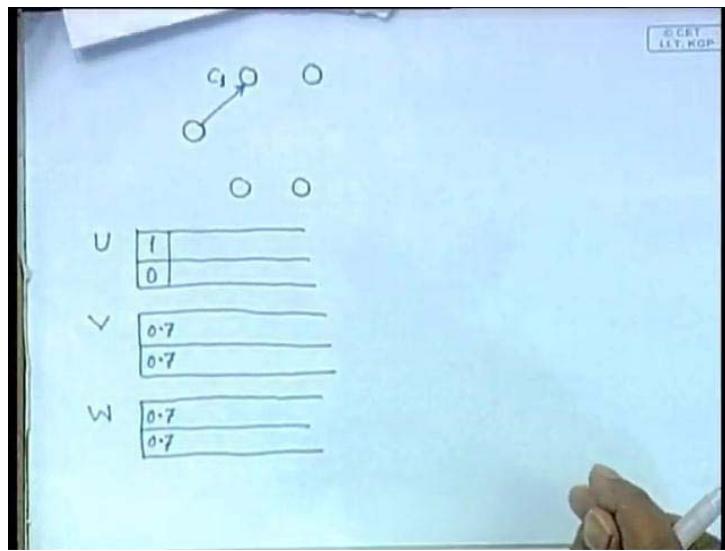
That is, we have taken a set of points as given over here. So, if I take the first point, that is (0.7, 0.7).

(Refer Slide Time: 01:34)



So, $(0.7, 0.7)$ as we said comes somewhere over here, that is this point. So, whenever we have this training sample or feature vector as we said that we construct a neural network or we construct one neuron.

(Refer Slide Time: 02:07)



In the middle layer of all fuzzy neural min max network as we said that we have the number of input nodes and the number of output nodes which are fixed. So, as we are considering two

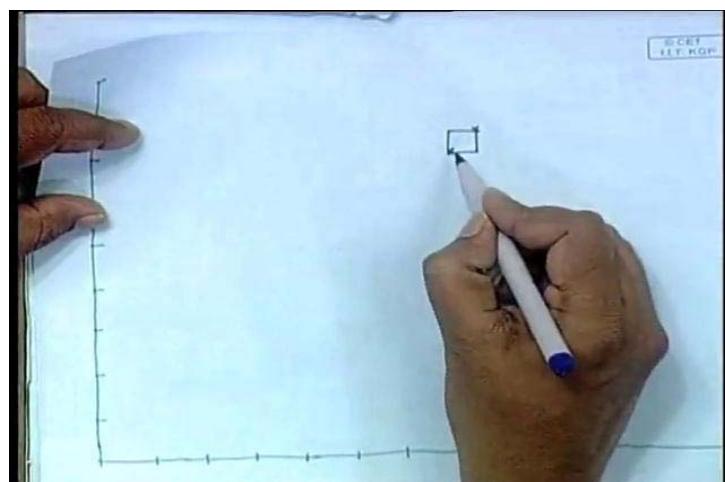
dimensional feature vector, so number of nodes in the input layer will be two and as we are considering just two classes, C1 C2.

So, number of nodes in the output layer will also be fixed. So, earlier what we tried to see is we try to construct three metrics, one metrics is U and the other metrics V and the third metrics W, okay? Where U metrics represent a connection from the middle layer neurons to the output layer neurons. And V metrics and W metrics they represent the min max points of the hyper boxes represented by different neurons.

So, as we said that each of these matrices will have two rows and the entries in the matrices will be made, while the neural network is trained. So, once I get this first training sample this represents a point hyper box for which the min point and max point will be the same. And as we know that this point belongs to class 1. So, in the U metrics I will make an entry (1, 0) indicating that this neuron is connected to the output layer neuron, representing class C1. And max points will be same as this feature vector, that is (0.7, 0.7).

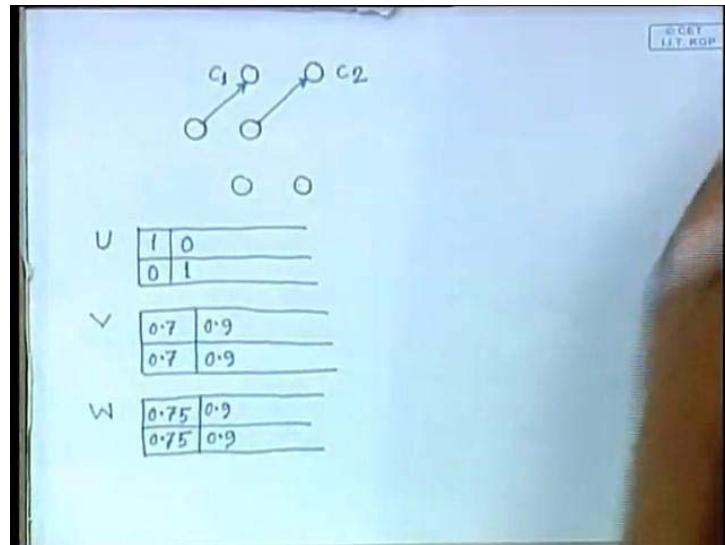
So, we put 0.7 over here and 0.7 over here. Here also you put 0.7, here also you put 0.7 and then when I took, we take the second point, that is (0.75, 0.75) which also belongs to the same class, class 1. So, the hyper box configuration, it will come over here, and I can expand the previous hyper box to include these points. So, I take a hyper box something like this and as we said that, as you are not creating any new hyper box so there is no neuron added in the middle layer, and as such the U metrics remain the same. Now, what happens is the min point and the max point.

(Refer Slide Time: 04:53)



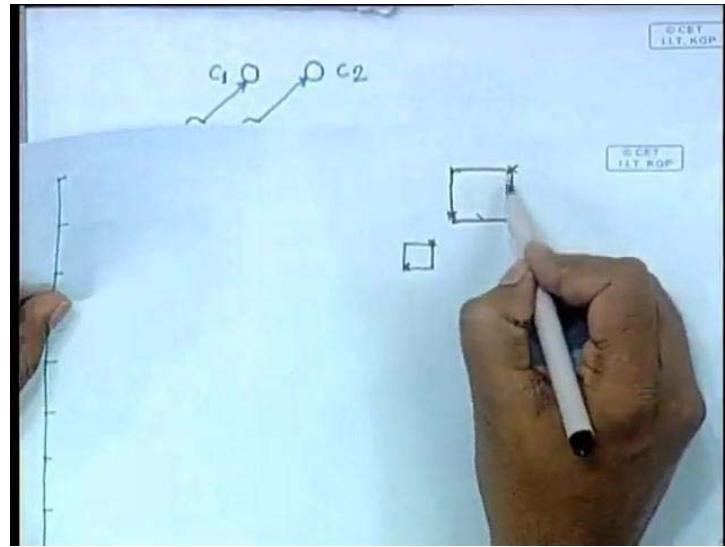
So, you find that the min point that remains the same as (0.7, 0.7) whereas, the new max point becomes (0.75, 0.75). So, I make the corresponding change in metrics W representing the max point, then I take the third point training sample which is (0.9, 0.9) at and this can example belongs to class. So, obviously have to make a new hyper box and over here it will be again a point hyper box representing (0.9, 0.9), and that will be somewhere over here, and because this belongs to class 2.

(Refer Slide Time: 05:18)



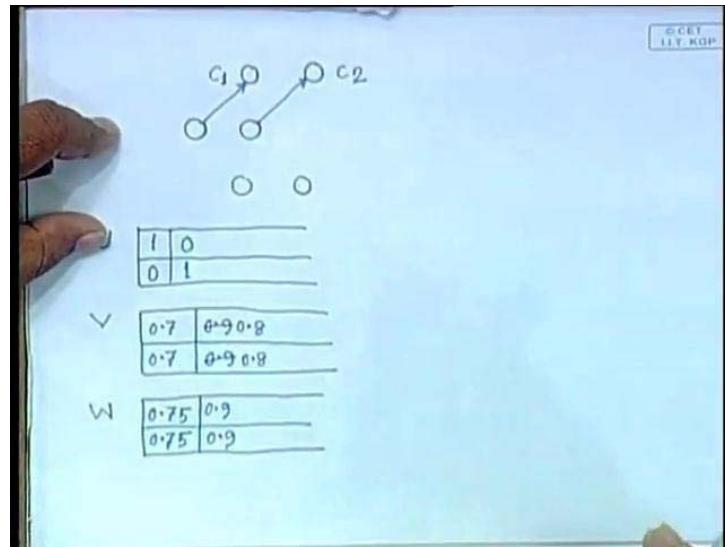
So, I have to make a new neuron in the middle layer and as this neuron belongs to class C2 so I will have the connection from here to here, where this output layer neuron represents class C2. Accordingly, in the new metrics I have represent, I have the entry 0 1, the min point and max point of this new hyper box, because it is a point hyper box. Again it will be 0.9 and 0.9. Here also it is 0.9 and 0.9, okay? So, then I take the next point which is 0.8, 0.8 and this also belongs to class C2.

(Refer Slide Time: 06:31)



So, I take this next point which comes here 0.8, 0.8, this point. And assuming that this can be expanded to include this point I expand this hyper box to include this point.

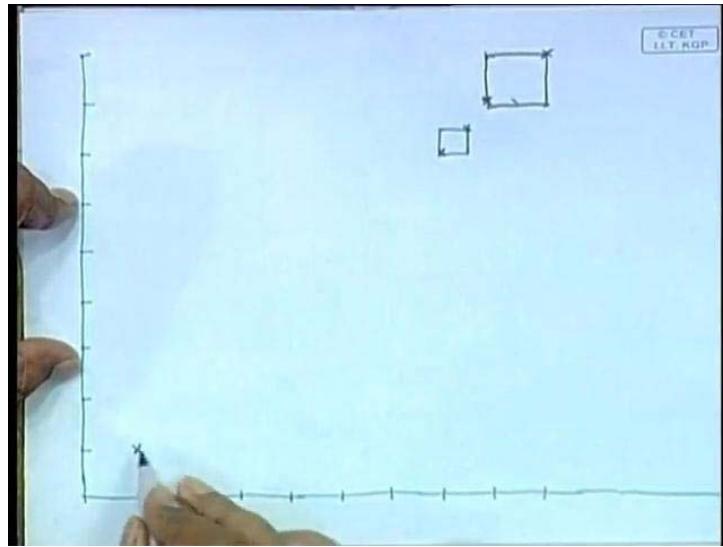
(Refer Slide Time: 06:58)



Accordingly, there will not be any change in the middle layer neuron and there will not be any change in the U metrics, but the min point and max point we have to change and

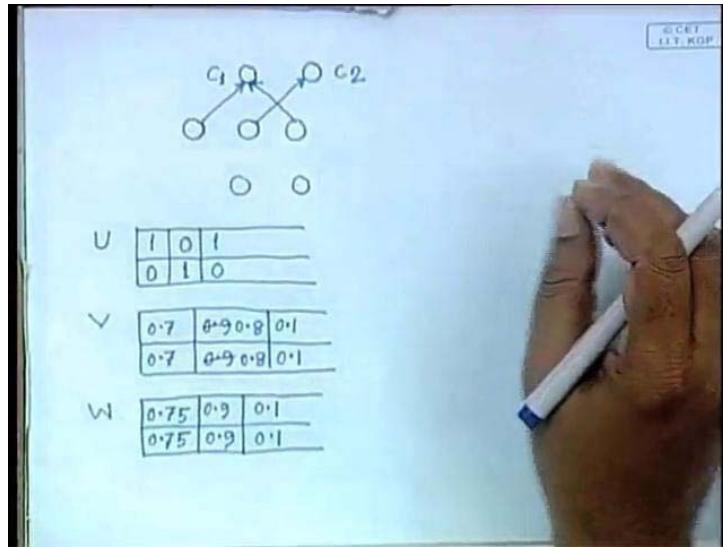
over here the max point remains the same, that is (0.9, 0.9) and the min point that becomes (0.8, 0.8). So, I change the min point to 0.8 and 0.8 and then I take the other point, that is (0.1, 0.1).

(Refer Slide Time: 07:36)



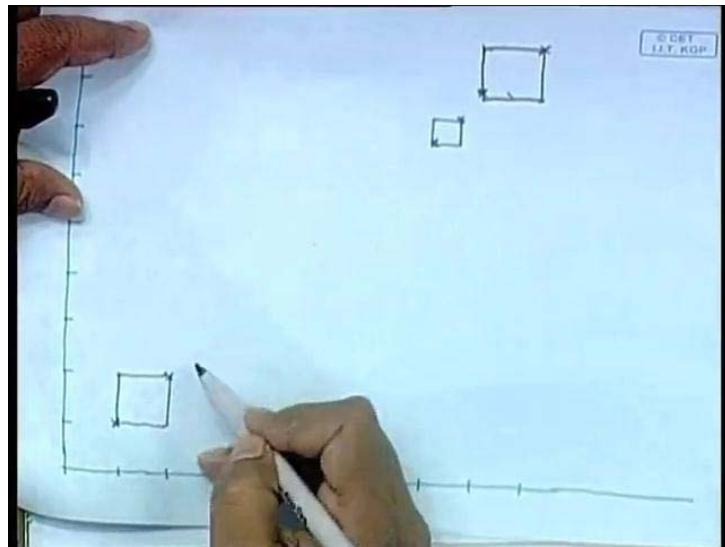
So enter in this is $(0.1, 0.1)$ and here as this cannot be included within the first point because this also belongs to class C1. And this also belongs to class C1, but I cannot expand this to include this. So, I have to enter a new node in the middle layer.

(Refer Slide Time: 07:55)



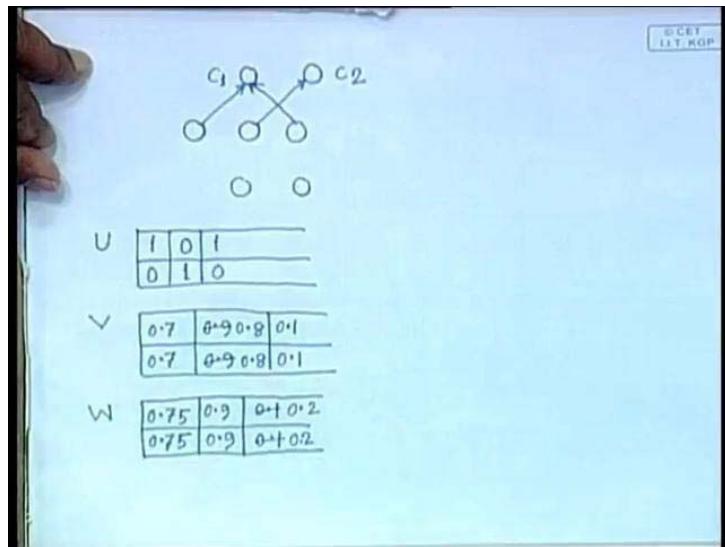
And this node belongs to class C1. So, accordingly I have to enter in the U matrix, $(1, 0)$ and the new entry in the min point and max points will be $(0, 1)$. Similarly, here it is $(0, 1)$. $(0, 1)$. then come to the next point, which is $(0.2, 0.2)$ again belonging to the same class C1.

(Refer Slide Time: 08:35)



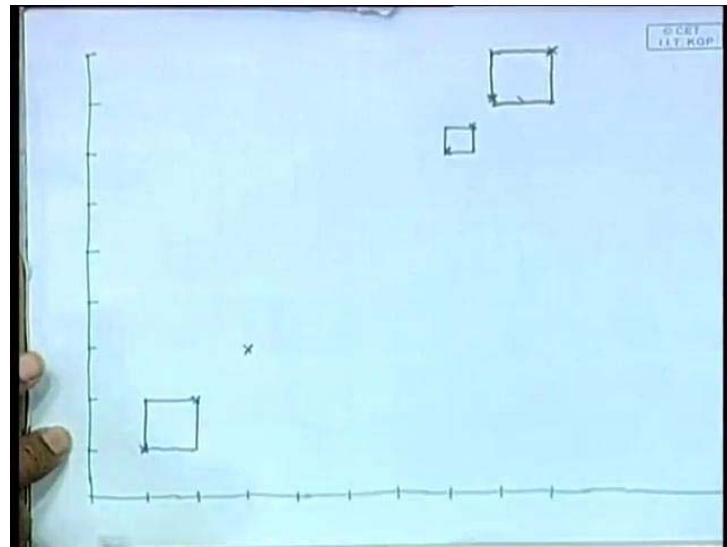
So, $(0.2, 0.2)$ is over here, I can expand this point hyper box to include this $(0.2, 0.2)$. So, I am not adding any new hyper box.

(Refer Slide Time: 08:51)



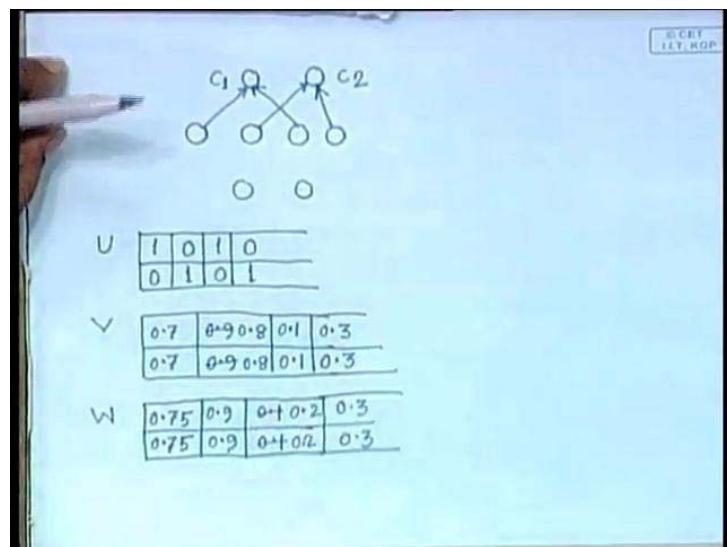
So, accordingly there will not be any change or any node added in the middle layer and there will not be any change in the U matrix.

(Refer Slide Time: 09:01)



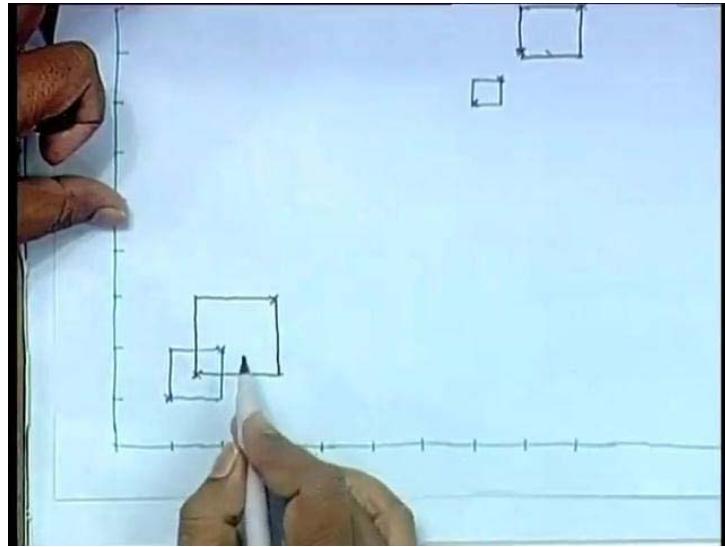
However, the min point and max point of the previous hyper box which has been expanded that will change.

(Refer Slide Time: 09:06)



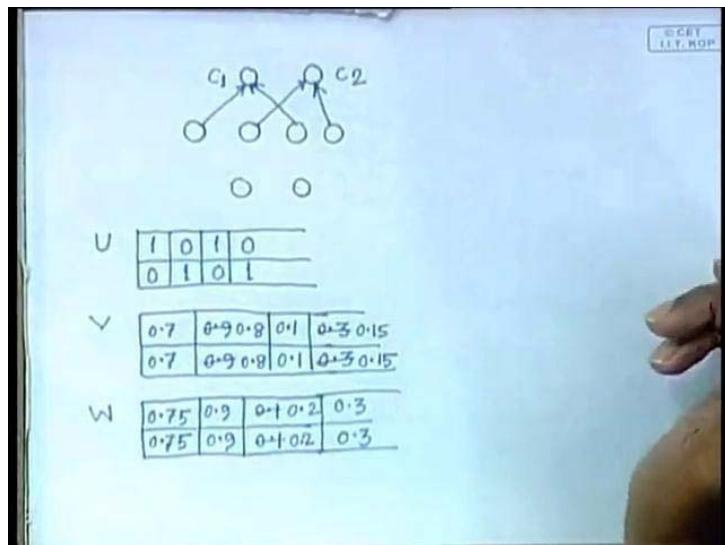
And here the max point will be $(0.2, 0.2)$, okay? Then take the next point which belongs to class C2 and the vector is $(0.3, 0.3)$.

(Refer Slide Time: 09:27)



So, I take this $(0.3, 0.3)$ which is over here and this belongs to class C2. So, I have to add new neuron in the middle layer.

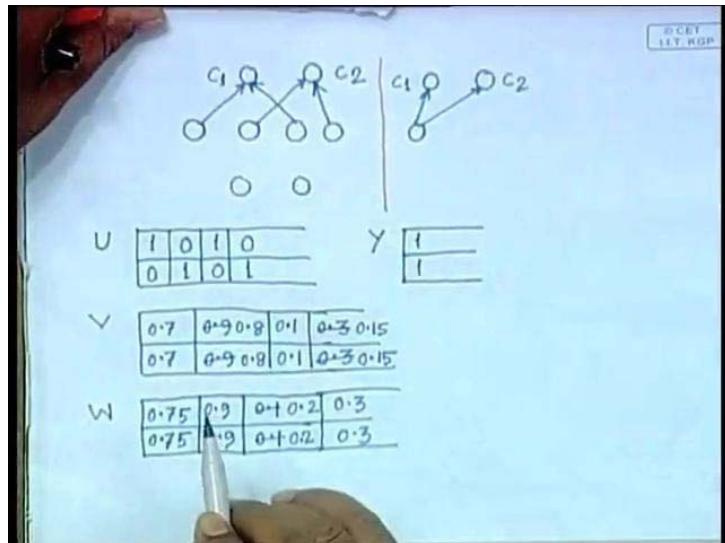
(Refer Slide Time: 09:39)



So, I have to add a neuron over here and these belongs to class C2. So, I have to change. I have to make an entry in the U matrix which is $(0, 1)$ and for this one I have to enter the min point and max point. And this being a point hyper box, both of them will be 0.3 and 0.3, okay? Then I consider the next one which is $(0.15, 0.15)$ belonging to the same class C2. So you enter $(0.15, 0.15)$ expand this point hyper box to include this $(0.15, 0.15)$. So, when I expand this what I have to do is, because I am not adding any new node so in the middle layer the structure will

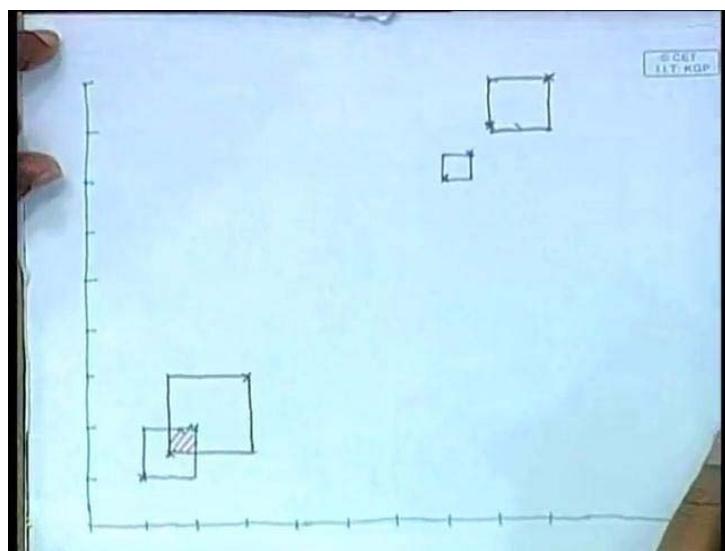
remain the same. In matrix U I do not have to make any more entry, only the min point and max point of this two will be different.

(Refer Slide Time: 11:01)



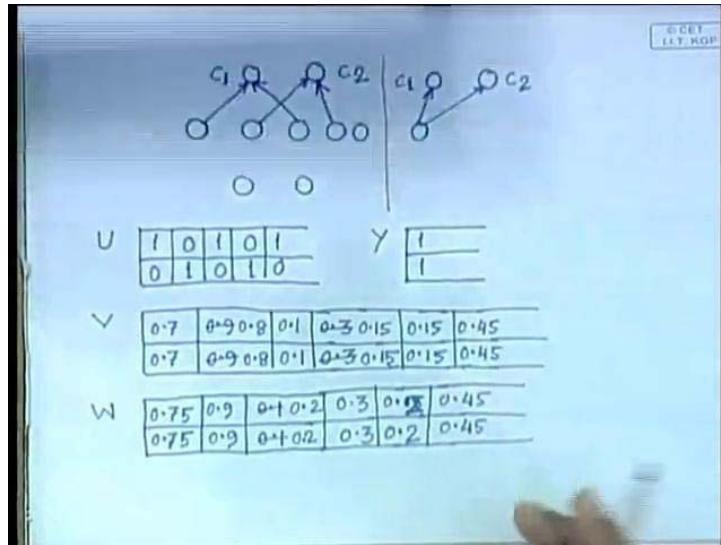
So, the min point, new min point of the previous hyper box will be 0.15 and here it is 0.15. And now, there will be a difference between the previous neural network that we were constructing and this new neural min max network we are making, because in this case you will find that there is an overlap, I have a overlap in this particular region.

(Refer Slide Time: 11:32)



So, what I have to do is in the compensatory section I have to add a new neuron which represents a hyper box of this overlap region. So, I will add a new neuron in the compensation section or in the reflex section. So, I will put over here in the overlap compensation section, I will add a new neuron.

(Refer Slide Time: 12:00)



And in the overlap compensation section also I will have two output layer nodes, one corresponding to class C1, other one corresponding to class C2. And the function of this overlap compensation neuron will be to calculate, what is the amount of compensation that has to be provided while calculating the membership function. And that compensation has to be added to the outputs of the neurons as given in the classification section. So, it has to get compensation to both C1 and C2. So, it has to give compensation to this class, also it has to give compensation to this class.

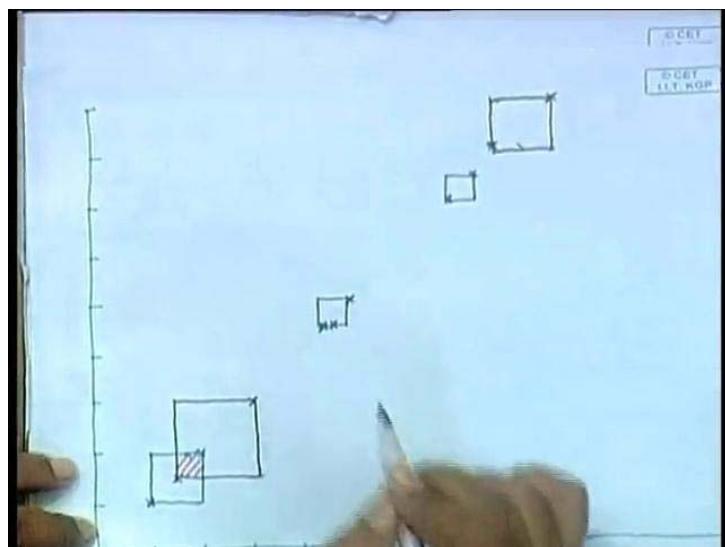
So, this is an output layer neuron in the reflex action responding to class C1 and this is output layer neuron in the reflex section corresponding to class C2. However, so accordingly as we have proposed the matrix U over here, let us assume that for this overlap compensation section I have another matrix, say Y. And in this matrix Y, because this neuron is feeding the output to both class C1 and C2. So, both of them will be made equal to high. However, if I have three classes, three such classes, one of the class will be equal to 0.

Because every node in the middle layer in the compensation section or overlap compensation section provides compensation only to two classes corresponding to the two hyper boxes, belonging to classes which overlap. So, every neuron in the middle layer over the overlap compensation section will be connected to two output layer neurons in the overlap compensation section.

So, here it is

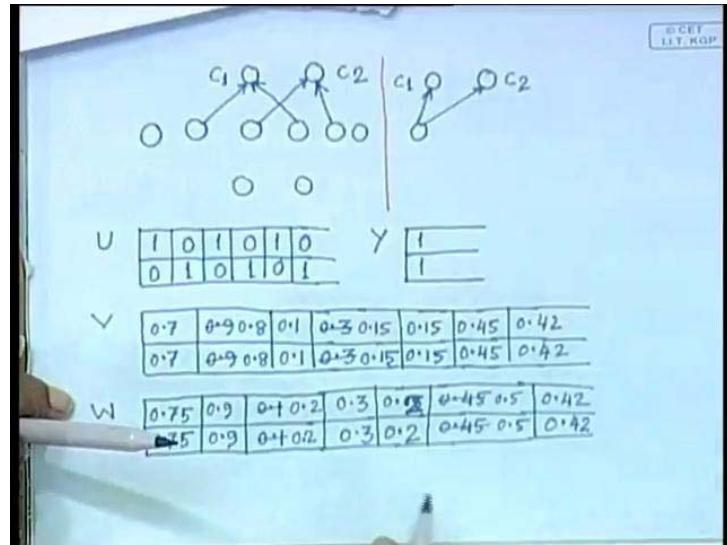
this Y matrix will be $(1, 1)$ and the min point and max point of this particular hyper box will be the min max point of this region.

(Refer Slide Time: 14:21)



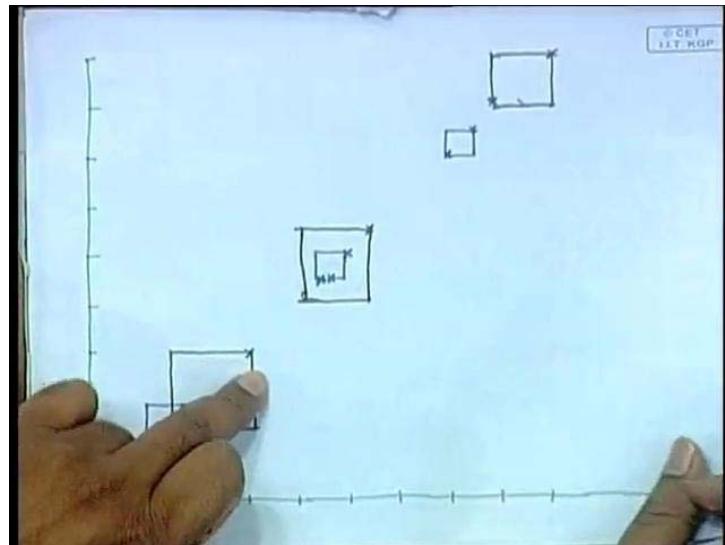
So, here you find that the max point is given by $(0.2, 0.2)$ and min point is given by $(0.15, 0.15)$, okay? So, accordingly more than two hyper boxes overlap in the same region is unlikely, because whenever by expansion hyper box gets overlapped with another hyper box the same hyper box will not be expanded any more. However even then if there is such a possibility then I will have pair wise compensation class 1 class 2, class 1 class 3, class 2 class 3 like that, but it is quite unlikely because once the hyper box overlaps, the same hyper box will not be expanded further. So, here the min point is $(0.15, 0.15)$ and the max point is $(0.2, 0.2)$.

(Refer Slide Time: 15:24)



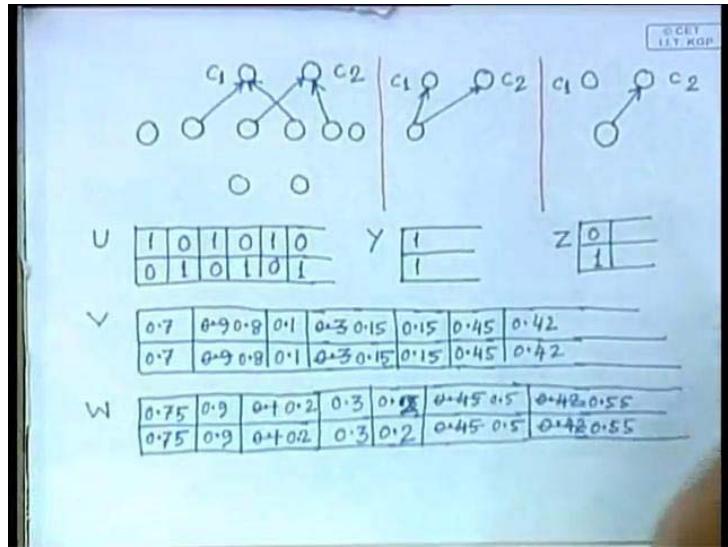
So, corresponding to this particular neuron I will enter in V matrix and the W matrix. The min point and max point as $(0.15, 0.15)$, and here it will be the max point will be 0.12 , sorry 0.2 and 0.2 . So, these are the min point and max point of the hyper box or the node in the overlap compensation region, but you notice that unlike in the previous case where we had broken these two hyper boxes to avoid such overlap, accordingly, these min max point are changing, but here we are not disturbing the hyper boxes which are already created in the classification section. So, that ensures that the points which are already learned, that knowledge which is already acquired, this knowledge will not be disturbed. And after that when I take the next point, that is $(0.45, 0.45)$ which belongs to class 1.

(Refer Slide Time: 16:56)



So, (0.45, 0.45) that will come somewhere over here and this belongs to class 1.

(Refer Slide Time: 17:04)

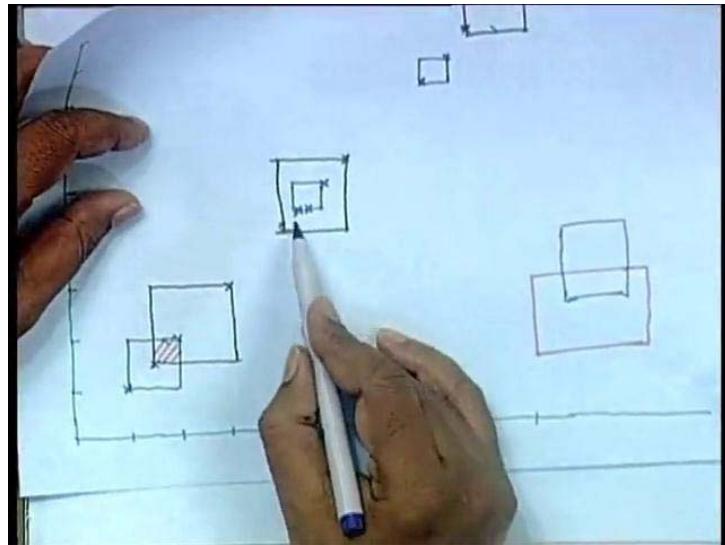


Again for this I have to enter one node in the classification section and this node belongs to class C1. So, I have to make in the U metrics an entry which is 1 0 and min point and max point of that is (0.45, 0.45) max point will also be 0.45 and 0.45, okay? Then I take the next point which is (0.5, 0.5) that is this one. Put over here, the previous one is (0.45, 0.45) so somewhere over here, (0.5, 0.5) that points come over here.

So, I have this hyper box and I do not have to add any node, because the node is already added. So, accordingly I do not have any modification in the U matrix but V and W matrices will be changed, but the min point is the same, but the max point now becomes 0.5 and 0.5.

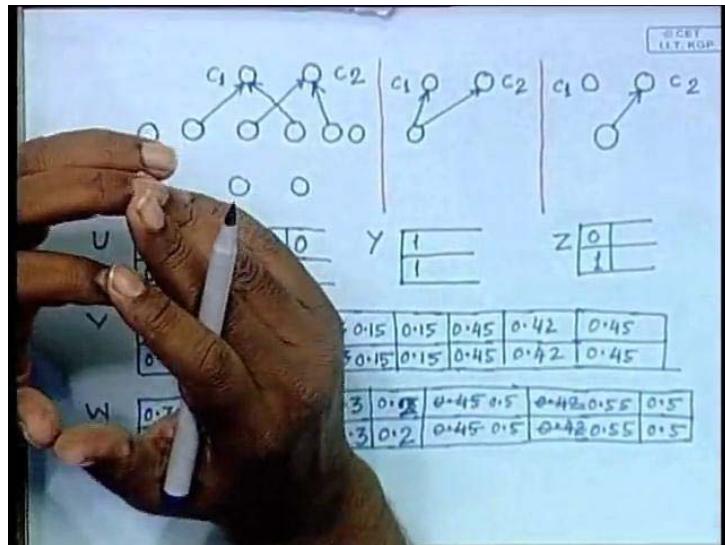
Then I take the next point which is (0.42, 0.42), that will come somewhere over here. This belongs to class C2. So, for this I have to add one more node, does not matter if I put on this side. And for this the min point and max point will be (0.42, 0.42) and because new node is added so I have to make an entry in the matrix U and this belongs to class C 2. So, the entry matrix U would be (0, 1) and for this my min point and max points will be (0.42, 0.42), (0.42, 0.42). Then I take the next point which is (0.55, 0.55) put in this (0.55, 0.55) this is 0.55, right? yes this was (0.55, 0.55) comes over here.

(Refer Slide Time: 19:58)



I expand the previous hyper box. So, accordingly because it is expansion of the previous hyper box so I do not have to make an entry in the middle of the classification section. So, the classification section neurons remain the same. U metrics also remains the same, but the change I have to make is in the max point of this new neuron, that is new that is being added.

(Refer Slide Time: 20:33)



So, when I change this max point it becomes instead of (0.42, 0.42) it becomes (0.55, 0.55) but while doing so you find that I have a containment. So, because there is containment I have put

one compensation neuron in the containment compensation section. So, in one more section that is containment compensation, so I have one more section here. Again I will have two neurons in the output layer corresponding to class C1 and corresponding to class C2. I have to add one compensation neuron containment compensation neuron.

So, to and this containment compensation neuron will give the compensation output to one of the classes, not to both. So, it will compensate the membership function of the class which contains the hyper box of the other class, right? So, accordingly the output of this neuron will be connected to class C2 and it will not be connected not to C1. Because the membership function of class C1 is not to be compensated where the membership function of class C2 is to be compensated. So, accordingly I have to have it, pardon container has to be compensated, not the one which is contained.

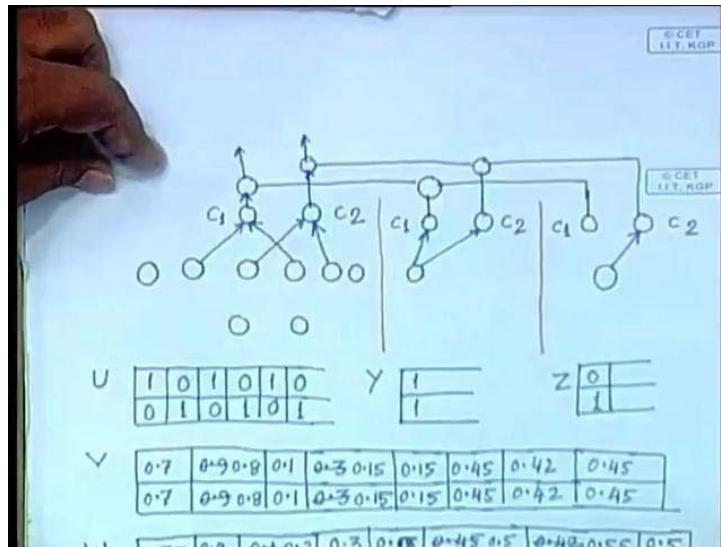
So, because here it is hyper box no need to class C2. So, the output of this neuron will be fed to the classifying neuron C2 in the containment compensation section. And to have this connectivity there is a code that will maintain another matrix, which is Z matrix, and here the entry will be 0 1, right? And you find that unlike in case of overlap where both the min point and max points were changed. In this particular case, I have the same hyper box belonging to class 1 whose min point and max point will be the min point and max point of this particular neuron. However, that is not always the case because I can have a situation where I will have partial containment, as we have already said that I can have a containment of this form.

Suppose, I have one hyper box like this belonging one class and I can have another hyper box like this, which belongs to another class, where this hyper box is not fully contained in this hyper box. It is partially containment. But, still it is containment because one of the dimension is content in the respective dimension of the other hyper box. So, to take care of such situations I have to make entries in the U matrix and the W matrix, those indicate that what will be the min point and max point of this particular neuron. And here the min point and max point of this neuron, this neuron is nothing but the min point and max point of the neuron, which is previously added. So, for that the min point max point was $(0.45, 0.45), (0.5, 0.5)$, right? So, I will have $(0.45, 0.45)$ and here it is 0.5 and 0.5.

So, you find that these two matrices V and W, they give you the min point and max point of all the hyper box nodes in the middle layer, whether that node is in the classified section falling overlap compensation section or in the containment section, matrix U to give you the

information of the connectivity from the middle layer nodes to the output layer nodes in the classifying section. Matrix Y gives you the information about connectivity between the middle layer nodes to the output layer nodes in the overlap compensation section. And this matrix Z that gives you the information of connectivity between the middle layer nodes and output layer nodes in the containment compensation section. And after this is done what is required is that I have to combine the corresponding outputs of the corresponding classes.

(Refer Slide Time: 25:37)



So, I have to combine the compensation output of class C1 over here with the compensation output of compensation class C1 over here. Similarly, I have to combine the output of compensation neuron C2 over here and output of this. So, that gives you the overall compensation that has to be made and finally, this is to be added to the membership function which has been computed by the classified neuron section. So, it comes over here. Similarly, over here and so final membership output I get from this point. So, these are the memberships which of the compensated membership functions of the sample through different classes.

So, then what we said is same case of overlap, I want that if a sample is somewhere over here that means it is near the min point and max point of one of the hyper box. If this distance is greater than the distance of the same point from the min point and max point of the other class, then its membership to the previous class should be more. Similarly, if the point is somewhere over here, which is nearer to min point and max point of this class then the min point and max

point of this class, then the membership function of this point to this class has to be more, okay?

Similarly, in case of containment if the point belongs inside then its membership to this contained class has to be 1. Its membership to the container class has to be 0. So, the membership functions or the compensation that has to be computed in case of overlap and the compensation that has to be computed in case of comp containment, they are different.

So, accordingly the functions, the compensation values which are to be computed by the middle layer neurons in the overlap compensation section. And the compensation values to be computed by the middle layer neurons in the containment compensation section, they will be different. So, how do they differ if I take a neuron, input connections to the middle layers are given by the matrices V and W. It is the same matrix which is used for these connections, used for this connections, used for this connections because what I need over here is for this neuron is what is important point for this neuron? What is the min point max point?

For each of these neurons what are the min points and max points? So, the inputs connections to all these neurons in the middle layer, whether it belongs to classifying section or it belongs to the compensation section, whether overlap compensation or containment compensation, all of them will come from the corresponding entries in the V matrix and U matrix. So, considering this that the compensation that has to be given to different classes are different compensation that has to be computed by the overlap compensation neurons, and containment compensation neurons are different. So, the kind of compensation that we can compute something like this.

(Refer Slide Time: 29:34)

$$d_{jl} = U(b_j(A, v, w) - 1) \times \left[-1 + \frac{1}{n} \sum_{i=1}^n \max \left[\frac{A_i}{w_{ji}}, \frac{v_{ji}}{A_i} \right] \right]$$

$\ell = p \text{ or } q$

So, we said that if we consider a neuron in the middle layer of the overlap compensation section, okay? So, to compute this what are the inputs that it may, that it needs? Obviously, it needs the information of the input vector A because that only decides, I mean location of this A will tell whether this is in the overlap section or not. In the overlap section, it also needs the information of the min points and max points of the hyper boxes which are overlapped. That will be used to calculate the amount of compensation that you want to give.

And it also needs the information of min point and max point of this node itself, because that will be used to check whether this point A is within the overlap region or it is not within the overlap region. And this neuron will give me two outputs corresponding to the two hyper boxes that are overlapped. So I can put it like this, if this neuron in the overlap region I represent this by say d_j .

So, this will give me two outputs, one corresponding to say the d_{jp} and one corresponding to d_{jq} . Indicating that the hyper boxes labelled as p and hyper boxes labelled as q, they have overlapped. And accordingly this output will be connected to the output layer neuron, and this output will be connected to input of the q^{th} neuron and the output function can be put like this

$$\text{is } d_{jl} = U(b_j(A, V, W) - 1) \left[-1 + \frac{1}{n} \sum_{i=1}^n \max \left[\frac{A_i}{W_{li}}, \frac{V_{li}}{A_i} \right] \right], \text{ where this summation will be } i \text{ varying}$$

from 1 to n. If I have n dimensional feature vector and this l can be p or q.

So, what does this expression give you? This $b_j(A, V, W)$, this V and W these are min points and max points of this neuron, that is the compensation neuron. Whereas, this V_l and W_l they are the min points and max points of the hyper boxes which have overlapped. So, this V and W there are the min points and max points of the compensation neuron. Whereas, V_l and W_l they are the min points and max points of the hyper boxes which have overlapped.

So, you find that this function $b_j(A, V, W)$, this is the same membership competition function of the classified section neurons. So, there we said that if a point falls within the hyper box then its membership function, membership value will be equal to 1. So, this V and W these being the min point and max point of the overlapped region. So, if a point falls within the overlap region then this $b_j(A, V, W)$ will be equal to 1.

If it does not fall within the overlap region, then the value of $b_j(A, V, W)$ will be less than 1 and its value will depend upon how far it is from that hyper boxes. So, you find that if the point falls within the hyper box then $U(b_j(A, V, W) - 1)$, If it falls out outside the hyper box, the overlap region in that case is this value is negative and if I use this U to be the unit step function.

(Refer Slide Time: 35:39)

GATE
IIT-KGP

A

d_{jP}

d_{jV}

$U \rightarrow \text{unit step fun}$

$U(x) = 1 \quad x \geq 0$

$0 \quad x < 0$

$$d_{jl} = U(b_j(A, V, W) - 1) \times \left[-1 + \frac{1}{n} \sum_{i=1}^n \max \left[\frac{A_i}{w_{li}}, \frac{v_{li}}{A_i} \right] \right]$$

So, here what is U? U is the unit step function. So, if it is unit step function, in that case $U(x) = 1$ for $x \geq 0$, and this is equal to 0 for $x < 0$, okay? So, this clearly says that if the point falls within the hyper box then this $b_j(A, V, W) - 1 = 0$ and the value of this function $U(b_j(A, V, W) - 1) = 1$. If it falls outside the hyper box then $b_j(A, V, W) - 1 < 0$. So, the value of $U(b_j(A, V, W) - 1) = 0$ and if the value of $U(b_j(A, V, W) - 1) = 0$ then this entire product is equal to 0.

And that is the amount of compensation. And if this U is 1 then the amount of computation depends upon what is of value computed over here. Now, here you find that the computation is $-1 + \frac{1}{n} \sum_{i=1}^n \max \left[\frac{A_i}{w_{li}}, \frac{v_{li}}{A_i} \right]$, w_{li} is the i^{th} component of the max point of w_l . Similarly, v_{li} is the i^{th} component of the min point of V_i . So, forgetting about n dimension, n dimensional feature vector. So, if I consider only one dimension, okay?

So, what effectively I will have is, $\max\left[\frac{A}{W_1}, \frac{V_1}{A}\right]$. Considering only one dimension, here you

find that if A and W_1 they are same, that means this unknown feature vector coincides with the max point then $\frac{A}{W_1} = 1$ is

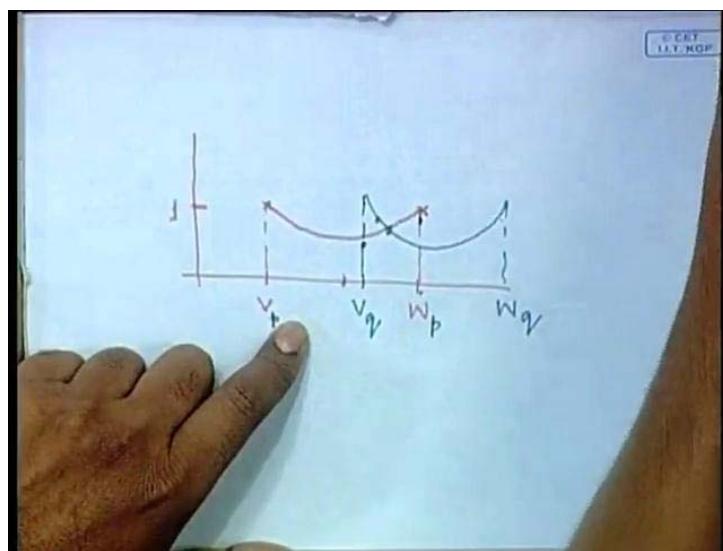
equal to 1. Or if $\frac{V_1}{A}$ they are also same, that is unknown feature vector coinciding with the min point, they are same. This is 1 if unknown feature vector coincides with the max point. Then

obviously $\frac{V_1}{A} < 1$ if unknown feature vector coincides with the min point. Then obviously

$$\frac{A}{W_1} < 1, \text{ okay?}$$

So, here you find that if the unknown point coincides with either the min point or max point then this max operator will return you value 1. And depending upon the distance of the point from the min point or max point, this max operator will return a value whichever is maximum. And if you compute this value only between the min point and max point of the same hyper box then the kind of function that was going to get is something like this.

(Refer Slide Time: 38:50)



Suppose, I have V over here and W over here of the same hyper box and depending upon the position of the point with respect to V and W , if it coincides with V or if it coincides with W

then the value returned by this max operator, that is equal to 1. As I move from V or as I move away from W, this value will go on reducing. So, I will have again after the min point as it approaches V the value will go on increasing. So, I will have a situation something like, this is not it and so as a move away from W the value goes on reducing. It will go on reducing up to midpoint, then again it will go on increasing because of influence of V.

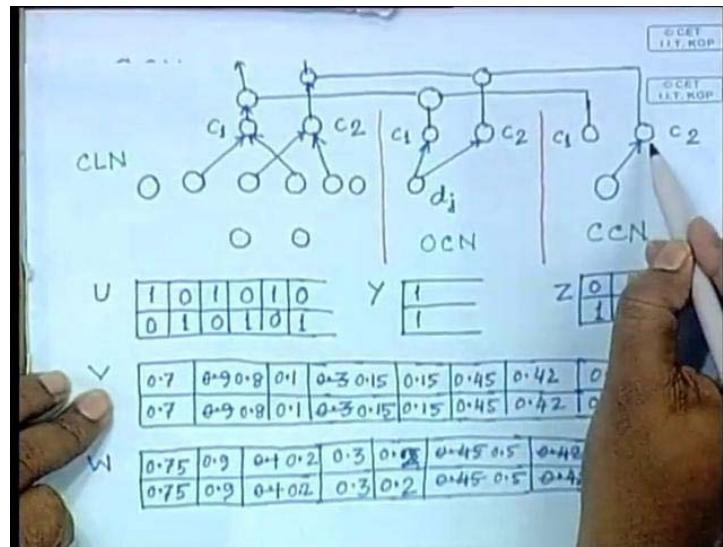
Similarly, over here if I have another hyper box. So, this is a V_p W_p i have another hyper box somewhere over here, it is V_q and W_q which are overlapped. Here also the compensation values will vary in this point. So, you find that if the point is nearer to V_q the compensation given by this is more. If it is nearer to W_p the compensation given by this is more.

This is another hyper box, I am considering only in one dimension, I cannot go multiple dimensions in a plane. So, this corresponds to one hyper box belonging to p^{th} class, this corresponds to another hyper box belonging to q^{th} class So, you find the compensation amounts, amount of compensation as computed by two hyper boxes. As computed by the hyper box for different classes is it okay? and that is being subtracted from 1, right?

So, wherever this component is more minus 1, this will be less. Where this component is less minus 1, that will be more and the magnitude will be more, and then this and obviously this component will be either 0 or negative, isn't it? Because this value will be greater than 1, its maximum value is 1, it can be less than 1 so this component is always negative. And this is, this component that I am adding to the membership functions as calculated by the classifying section neurons. So, the final output, the maximum will be 1 if there is no compensation or it will be less than 1 if there is compensation. And by how much less that depends upon the relative position of the point within the overlap region, whether it is nearer to the min point or max point of one class with respect to the other class. Is that okay?

So, as a result within the overlap region what we get is somewhere over here. The membership function computed to this class will be more and somewhere over here the membership function computed to this class will be more. So, that is what is about the overlap compensation regions. Now, what about this containment compensation? As we have said membership functions or the amount of compensation that provide will be different from the amount of compensation that you provided in this case. So, here this is what is given by the, let us put it as OCN or overlap compensation neuron.

(Refer Slide Time: 43:06)



Accordingly, I marked this as, I will put it as CCN section, C L N section classifying neuron section. This section is the overlap compensation section and this is CCN containment compensation section, okay? And as we said that unlike in this overlap compensation section where the output will be provided to the neurons of two classes in case of containment compensation, the output will be given to one of the class.

(Refer Slide Time: 43:47)

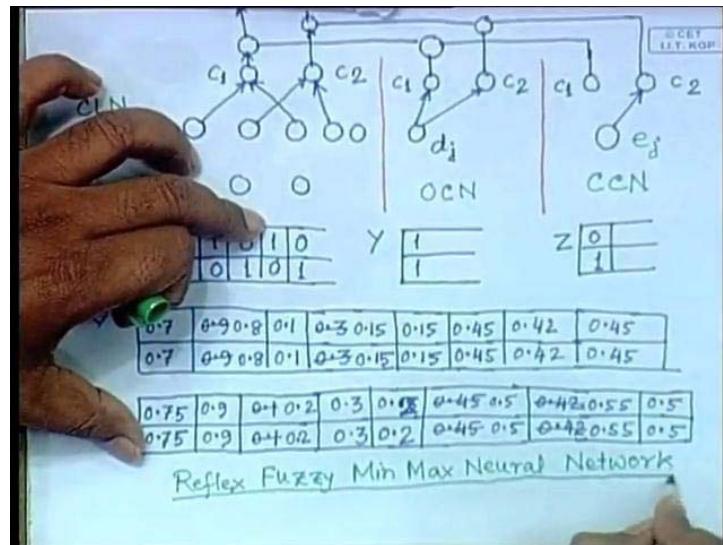
CCN

$$e_d = -1 \times U(b_d(A, V, W) - 1) = 1$$

So, accordingly a schematic diagram of CCN, a containment compensation neuron will can be something like this. Again it needs the input of the min points and max points and it has a single output because output will be going to only one output layer neuron, and if I put it as say neuron e_j , call this as neuron say e_j . So, the output of e_j , I can put it like this, that this will be $e_j = -1x(b_j(A,V,W)-1)$. So, this function simpler than the functions to be computed by overlap compensation neurons. So, we find that here what is it compute when an unknown point falls within the contained region. So, by contained region what I mean is either in this region, in case of partial containment, if it is full containment in this region and this V and W they are the min points and max points of the contained hyper box. That is either is min point or max point of this or it is the min point and max point this in this partial containment.

So, again as before if the point A falls within the contained region in that is $b_j(A,V,W)$ as before, this is equal to 1 and this quantity $b_j(A,V,W)-1$ will be equal to 0, right? If it falls outside this, then value of this will be equal to 0 indicating that there is no compensation. And if it is also within this then, what is the amount of compensation value of this function U ? This is equal to 1 if it falls within the containment region, because then only this argument is equal to 0. That means, the value of e_j will be equal to -1.

(Refer Slide Time: 46:34)



So, if the value of $e_j = -1$ and this output is connected to the output layer neuron which contains the other hyper box, right? So, in the classifying section neuron the corresponding output node has computed a value of +1. This gives a compensation of

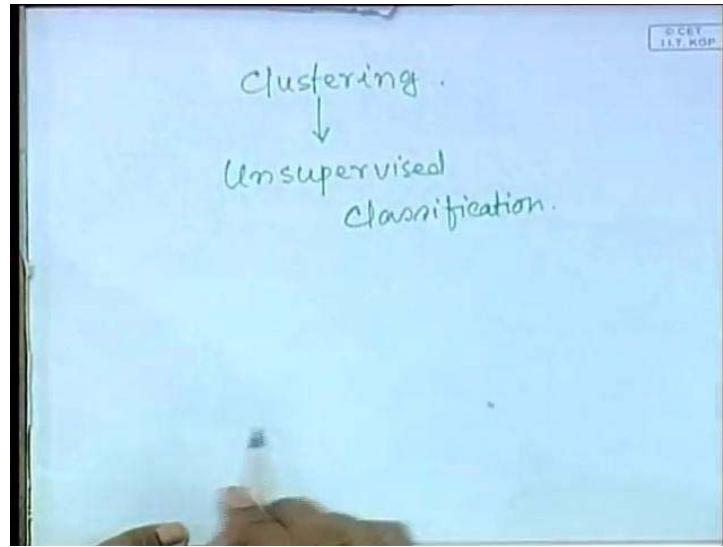
-1, when you add this two, another final value become equal to 0, right? However, this sort of containment compensation is not connected to, is not compensating out of other classifying neurons. So, those values will be as they are computed. So, by using this over here the membership function to class C2 will be equal to 0. Membership function to class C1 will be equal to 1, because this output is not connected to the output. It is not compensating the output of C1 classifying neuron, however it is compensating out of output of C2 classifying neuron. And the amount of compensation is -1. So, this neuron has computed a value of plus 1, this has given a compensation of -1. So, final value becomes equal to 0. Whereas, for this one this also has computed a value of plus 1 but there is no compensation.

So, the final output will be the final membership value will be equal to 1 to class c 1, right? Now, what happens if it falls in this region? The amount of compensation will be equal to 0, if the amount of compensation is 0 then the neurons in the classifying section, this neuron C2, yeah the neuron C2 will give you membership value equal to 1. This neuron C1 will give a membership function which is less than 1 and that depends upon the distance of this point from the min point and max point of this hyper box, which is obviously less than 1. Whereas, the membership function computed by C2 will be equal to 1.

So, here I get a membership function where output of C2 is one output of C1 is less than 1. So, if I go for hard classification using this membership values then obviously the point will be classified to point class C2, not class C1, whereas if it is within the content region, the classification will be to plus C1 and not to plus C2. So, you can make this modification for improvement of this fuzzy neural min max network and this is what we are calling as reflex fuzzy neural min max network.

And it has been found experimental that the performance of this reflex fuzzy min max neural network is much, much better than the performance of the fuzzy neural min max network as proposed by Simson. Obvious reasons we are not going to any contraction and as a result error introduced while learning or training of the neural network is avoided. And then we will see later when you talk about the clustering that a further modification of this neural network can also be used for clustering operation. So, what you have discussed so far is classification operation.

(Refer Slide Time: 50:39)



So, the next point that will discuss is the clustering, which makes use of the concept of unsupervised classification. So, what we have seen so far? We have said that those are supervised classification because for designing the classifier or even for training the neural network, we use a set of samples for which a class belongingness is known or the labelled samples. And using the labelled samples you train the classifier or you train the neural network, and then use that trained classifier to classify the unknown samples, so as we are training the classifier or you are designing the classifier using the labelled samples.

So, as if your training operation is supervised, you are supervising the training operation. So, that is why this is called supervised learning. Whereas, there is another kind of situation that we frequently come across where giving the raw set of data, we have to analyse the data and based on some sort of similarity we have to form clusters of data. So, it is something like this that have been given a set of data and you partition that set into number of sub sets. Where the data of the vectors belong to each partition will be similar based on some similarity measure. Whereas, if I take a vector from one partition and another vector from another partition, those two vectors will be different based on the same similarity measure. So, what you have to do is we have to analyze the data and based on the similarity measure we have partition the data into different subsets. And that is what is known as clustering and because in this case we are not using any labelled sample for doing this clustering operation.

So, this is what is also known as unsupervised learning or unsupervised classification. And if time permits, later on we will see how hybrid approach that we can have a mixture of supervised

and unsupervised learning. Because if your problem domain is very complicated where you will have thousands and thousands of data for training of classifier, it is not physically possible to label each and every training data for its class belonging. So, we can have a small set or few samples of data from different classes, label them using that you go for classification. And the unlabeled samples because this will be similar to the labelled samples taken from the same class.

So, the unlabeled samples can further refine your classifier, that is what is a hybrid approach where both clustering and classification or supervised and unsupervised learning approaches that used together. So, if time permits we will discuss about that also. But next class onwards I will talk about the unsupervised classification or clustering. So let us talk later about that.

Pattern Recognition and Application

Prof. P. K. Biswas

Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology, Kharagpur

Lecture - 34

Clustering

Good morning, what we are going to discuss today is unsupervised learning or clustering. So earlier whatever classification or the classifier design techniques or the learning techniques that we have talked about in all the cases we had some training samples that is the samples for which the class belongingness is known. So we have taken few samples from various different classes and using those samples or labeled samples we have tried to design the classifier or in case of neural network we have to we have tried to train the neural network.

And because in all those cases they will make use of labeled samples to design your classifier or to train the neural network such a kind of approach is called supervised learning because your learning process is supervised by the set of labeled training samples. Whereas in the other kind of problem domain where you have been given a set of raw data, okay?

And we have to partition that set into a number of subsets where we have to consider that the samples belonging to 1 subset they are similar based on some similarity measure whereas if I take samples from different clusters or the samples from different partitions, then those feature vectors will be dissimilar based on the same similarity measure. And this is a kind of approach where you try to partition the set of feature vectors into different partitions is what is called unsupervised learning, okay? Because here we don't have any set of labelled training data to guide the partitioning process and there are different types of approaches for this clustering problems 1 of them is called agglomerative clustering.

(Refer Slide Time: 02:21)

© CET
I.I.T. KGP

Unsupervised Learning - Clustering

Agglomerative Clustering:

$N: x_1, x_2, \dots, x_N \rightarrow d\text{-dimensional}$

{ 1. Begin with N clusters
2. Repeat step 3 $\underline{N-1}$ times \leftarrow
3. Find most similar pair of clusters,
merge them.

So what we'll be doing today is what is known as agglomerative clustering. So agglomerative clustering is actually hierarchical process that is either we can have a bottom-up approach or we can have a top-down approach. So in the bottom-up approach we can assume that initially all the samples or all the feature vectors which are given they form actually singleton sets that means initially if I have N number of feature vectors. Then initially I will have N number of clusters where every cluster will consist of a single feature vector, then hierarchically you go on combining the feature vectors into different groups and that way you generate a tree kind of structure, so this is what is bottom-up approach.

And the other one is top-down approach where initially I can think that all the samples belong to the same cluster that means, I have a single cluster and then try to divide that clusters into number of clusters in a top-down approach, that is also agglomerative clustering. But it is top-down agglomerative clustering or the approach that you are taking is are divisive approach that is starting from a single cluster you try to generate a number of clusters that is you are dividing the single cluster into number of clusters in a hierarchical manner.

So whenever we go for say initially let us take this bottom-up approach that initially I have N number of feature vectors. So the total number of each are vectors which are given is capital N , which I can mark as X_1, X_2 up to say X_N . So these are N number of which are vectors. Every feature vector will be of dimension say D , so I have this in number of the dimensional feature vectors, okay? So the simplest approaches as I said that, initially I will assume that I have N number of clusters but every cluster consists of a single sample okay, then hierarchically I'll go on combining them.

So it is something like this, so in the first step I begin with N number of clusters for every cluster consists of a single vector, okay? Then in the second step what I can do is, I can have a repetitive approach so I can repeat the next step $N - 1$ times. So in step 3 what I have to do is, I have to find most similar pair of clusters and merge those 2 clusters okay. So if I do it $N - 1$ number of times you'll find that at the bottom most level, I start with N number of each effectors so they are a total N number of which are vectors then what I do is in the next step I find out the most similar pair of clusters, okay?

So suppose these are the 2 clusters which are most similar, so I have to merge them. After merging them the number of clusters that I have is $N - 1$, okay? In the next step again may be that this cluster has become similar to some other cluster over here so to merge these two, okay? So in this step in the third step I have $N - 2$ number of clusters, so this way if I go on merging pair of most similar clusters in every step after this $N - 1$ number of steps, I'll have a single cluster where all the samples are put in the same cluster, okay?

So I have a complete tree structure and you find that this tree that you are going to create is a binary tree because at every step I'm combining only two of the clusters into a single cluster. So that tree will be a binary tree and I can break or I can terminate creation of, this is what I said is this is the maximum number of steps that I can perform I can terminate this algorithm at any step where I have attained the required number of clusters. So I can specify that what is the number of clusters that I need and I can terminate this algorithm at any step when I reach the desired number of clusters, okay?

So these are the basic steps in this agglomerative clustering or bottom-up agglomerative clustering, okay? So what I have to find out is, I have to find out the most similar clusters, the pair of most similar clusters and it is the measure of similarity that decides that what kind of clustering

algorithm or what kind of agglomerative clustering algorithm that we are going to have, okay? So as we said that when it comes to the terms of feature vectors then my pattern is actually represented by a point in that D dimensional feature space because our feature vectors we are assuming to be D dimensional feature vectors. So every pattern is actually represented by a point in the D dimensional feature space.

So to measure the similarity between two different patterns I have to measure what is the distance between the corresponding feature points okay. So if this distance is very high, I can say that the patterns are not similar, if the distance is small that means the points are very close to each other we can say that the patterns are similar to each other okay. So basically what we are doing is we are trying to find out some sort of distance value between two feature vectors say X_P and X_Q and this distance value is the measure of similarity or the measure of the similarity between those two feature vectors.

(Refer Slide Time: 10:09)

© CET
I.I.T. KGP

Single Linkage Algorithm (Nearest Neighbor)
 Complete Linkage Algorithm (Farthest Neighbor)
 Average Linkage Algorithm.

$$D_{SL}(c_i, c_j) = \min_{a \in c_i, b \in c_j} d(a, b)$$

$$D_{CL}(c_i, c_j) = \max_{a \in c_i, b \in c_j} d(a, b)$$

$$D_{AV}(c_i, c_j) = \text{avg } d(a, b) \quad a \in c_i, b \in c_j$$

Now when I say whether two clusters are similar or not, okay? That means I have to find out what is the distance between two clusters? And every cluster is nothing but a set of point so I have to find out what is the distance between two sets, okay? So accordingly there are different types of distance measures that can be defined between two sets, one of them is called single linkage and

according to the distance function the distance measure that we use I can have different types of algorithms one of them is called single linkage algorithm or this is also what is nearest neighbor algorithm, okay?

The other algorithm that I can have is what is called complete link as algorithm or this is also called for this farthest neighbor algorithm or we can have something called average linkage algorithm, okay. So what is the single linkage algorithm or the nearest neighbor algorithm? Suppose I have got two sets of points, okay and I find out a point from first set and a point from the second set which are nearest, okay. So I form every pair of the feature vectors with one pair from the first set and with one feature vector from the first set and the other feature vector from the second set.

So for every such possible pair of which are vectors I compute what is the distance okay? And the single linkage or the nearest neighbor algorithm says that out of all these distances that I have created I have to consider, I have to consider which is the minimum distance, okay. So formally it can be defined like this the distance D_{SL} or single linkage distance is nothing but, so single linkage distance between two clusters see C_i and C_j , it is nothing but compute the distance between two feature vectors A and B where say, a belongs to class cluster C_i and b is taken from cluster C_j , okay and I have to compute the minimum of these distances, okay.

So I take 1 feature vector from C_i and another feature vector from cluster C_j . Find out the distance and compute this distance for every pair of such possible feature vectors, okay. And I have to compute what is the minimum of these distances so the single linkage algorithm is based on this minimum distance okay whereas the complete link has algorithm. So if I put it as the D_{CL} between the same C_i and C_j , between the two cluster C_i and C_j this will actually be maximum, okay? So in case of single linkage it is the minimum distance, in case of complete linkage it is the maximum distance.

That means I have to consider the farthest pair of feature vectors with 1 feature vector from clusters C_i and the other feature vector from cluster C_j , okay. And in case of average linkage algorithm, I have to find out the average of the distances. So I have to compute what is the average of this distance values and that gives you the distance between two clusters C_i and C_j , okay. So in this case it is the minimum distance in case of completely because it is the maximum distance and in

case of average linkage it is the average distance of the pair of feature vectors with one feature vector again from class clusters C_i and the other feature vector from cluster C_j .

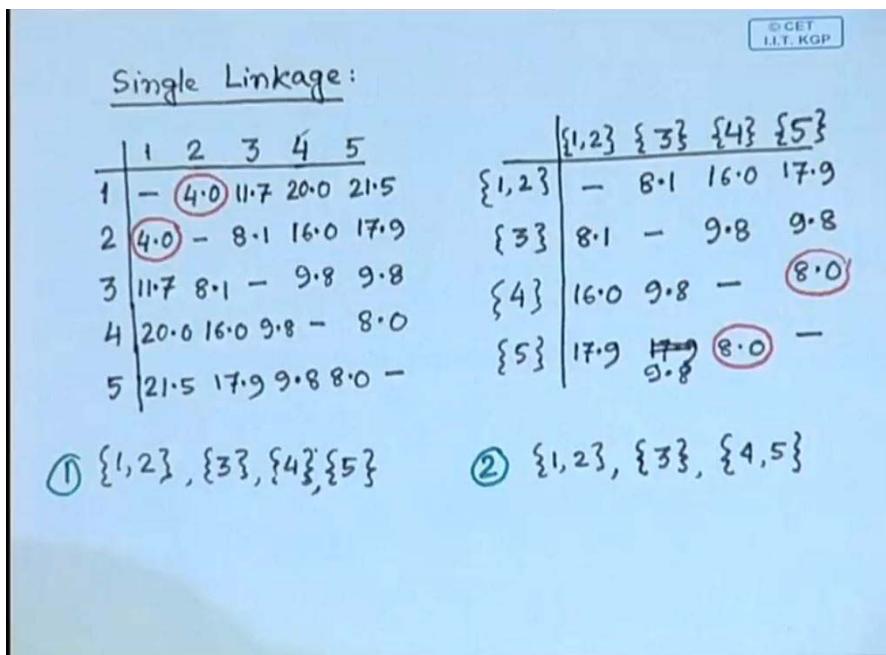
(Refer Slide Time: 15:26)

	X	Y
1	4	4
2	8	4
3	15	8
4	24	4
5	24	12

So now let us take an example to illustrate these different algorithms of what clustering result we obtain using these different algorithms. So let us take a set of feature vectors and I put this as and I assume that I have two dimensional feature vectors. So the feature components are X and Y ok and I take a number of points. So the first feature vector suppose it is (4, 4), the second feature vector is supposing (8, 4), third feature vector let me take as (15, 8), fourth feature vector maybe say (24, 4) and the fifth feature vector I take a say (24, 12), okay?

And let us first see that what will be our clustering output if I go for single linkage algorithm. So to have the single linkage algorithm or to have the clustering I have to find out that distance between every pair of clusters and what we are assuming that initially I assume that each of these feature vectors form a singleton cluster okay. So what I have to do is I have to compute the pairwise distance put it on another page.

(Refer Slide Time: 17:01)



So firstly what we'll consider is the single linkage algorithm okay. So I have to find out the distance between the pair of feature vectors so I have first feature vector, second, third, 4th and fifth total I have 5 feature vectors okay. So naturally distance between 1 and 1 is 0 because it is the same point, I have to find out what is the what is the distance between 1 and 2, ok. So here you find that your first point is the first feature vector is (4, 4) and the second feature vector is (8, 4)

So if I compute the distance between the first feature and second feature vector obviously it is 4.0, okay. So 1, 2 the distance is 4.0 and obviously 2, 1 that will also be 4.0 because this distance function is symmetric. Similarly, if I compute the distance between 1 and 3, 1 is (4, 4), 3 is (15, 8), okay? I can compute what is the distance between these two points 1 and 3, and the distance between 1 and 3 comes out to be 11.7 okay. So similarly the distance between 3 and 1 that will also come out to be 11.7

Then distance between 1 and 4 that will come out to be 20.0. So 4 and 1 that will also be 20.0. Similarly, the distance between 1 and 5 will come out to be 21.5 Here also 5 and 1, this will be 21.5. I don't really want what is the distance between 1 and 1 because that is equal to 0 similarly distance between 2 and 2, that is also equal to 0 so what is the distance between 2 and 3 again I

can compute 2 is, feature vector 2 is $(8, 4)$, 3 is $(15, 8)$, I can compute what is the distance between these two and this distance comes out to be 8.1.

Similarly, 3, 2 it will also be 8.1 okay. Then what is the distance between 2 and 4, this comes out to be 16.0. So 4, 2, that will also be 16.0 okay. Then 2 and 5 this comes out to be 17.9 okay, so 5, 2 that is also 17.9. So here it is 3, 3, again it is zero, what is the distance between 3, 4, if you compute this will come out to be 9.8. So 4, 3, this will also be 9.8, okay. 3, 5, this distance also comes out to be 9.8, 5, 3 will be 9.8, okay. Again 4, 4, I don't bother that is equal to 0. Distance between 4 and 5, this is 8.0. So 5, 4, this is also 8.0, what is left out is 5, 5, that's zero, okay.

So I have got a distance matrix and this distance matrix tells you that what is the distance between every pair of feature vectors okay. Now out of this, I have to take that pair which has got minimum distance value, okay? So if you look at this distance matrix you find that the pair $\{1, 2\}$, that has the minimum distance which is equal to 4. So at the first step I have to combine the feature vector 1 and 2 into a single cluster and other feature vectors will remain as singleton clusters okay.

So in the second step, so what pair I have formed over here is $\{1, 2\}$, that gives me one pair feature vector 3 remains as a singleton, cluster 4 domains are single cluster and 5 also remains as singleton cluster. so the first step is the clustering output that I got. Now I have to do that clustering using this, at the second level. So at the second level the clusters that I have is $\{1, 2\}$, coming to a single clusters 3, 4, 5, here also $\{1, 2\}, \{3\}, \{4\}$ and $\{5\}$, let us put them as set, okay.

So again distance between $\{1, 2\}$ and $\{1, 2\}$, obviously it is zero, what is the distance between $\{1, 2\}$ and 3, okay. To compute the distance between $\{1, 2\}$ and 3, I have to compute what is the distance between 1 and 3 I have to compute what is the distance between 2 and 3 and because it is the single linkage algorithm, I have to take the minimum of these two distances okay. so from here you can find that the distance between 1 and 3 is 11.7 the distance between 2 and 3 is 8.1. so minimum of these 2 is 8.1 okay. So I have to take that 8.1 as the distance between this cluster and this cluster, okay.

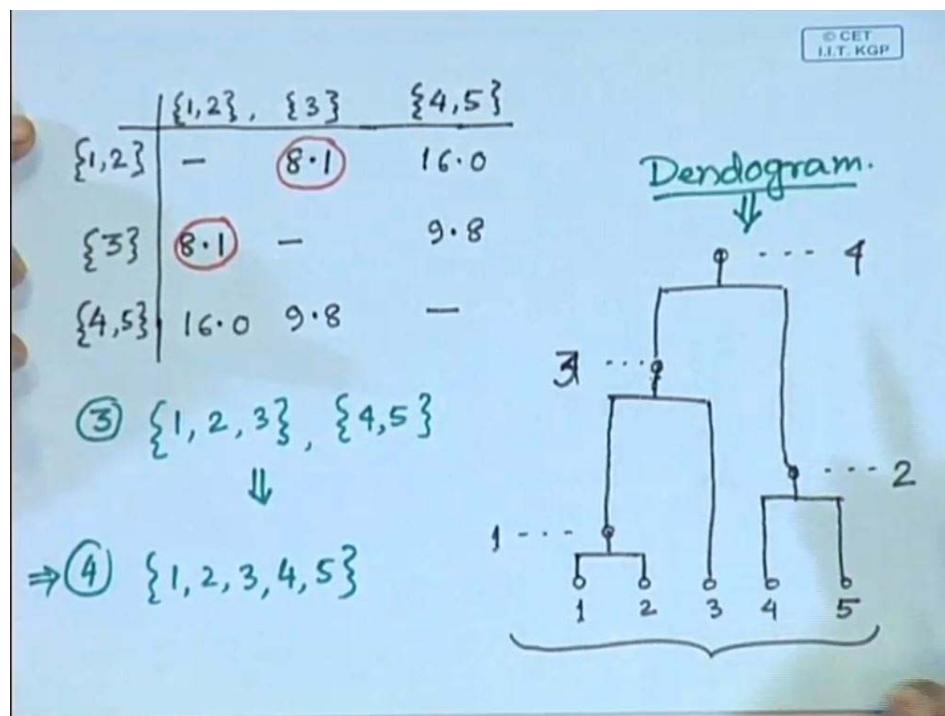
So similarly, here also the value will be 8.1, right? distance between $\{1, 2\}$ and 4 distance between 1 and 4 is 20, distance between 2 and 4 is 16.0. So the minimum of these two is 16.0. Similarly,

over here the distance will be 16.0. Between $\{1,2\}$ and 5, distance between 1 and 5 is 21.5, 2 and 5 it is 17.9, so the minimum is 17.9, here also it is 17.9. Then the distance between 3 3 obviously it is zero I don't need it, 3, 4 because 3 and 4 still remains a singleton sets so it is the same distance value which will repeat 9.8, here also 9.8.

4, 3, it is 9.8, 4, 4 is zero, 4, 5 it is 8.0, then 5, 3 it remains as 17.9, okay. 5, 3 is 9.8 or 3, 5 is 9.8 okay, and what is the distance between 5, 4 all right 5, 4 it is 8.0, 5, 5, it is zero okay. Now again I have to find out that what is the minimum of these distances okay, so here you find that minimum of this distance is 8.0. So at the next step I have to combine these two feature vectors 4 and 5, so after second step the feature vectors that I have or the clusters that I have is $\{1,2\}$ that remains as a single cluster then $\{3\}$ and then $\{4,5\}$. So these are the 3 clusters that have made after step 2.

So this is what I have got in the first step, this is what I have got in the second step. Then I again have to perform the same operation so next, the feature vector.

(Refer Slide Time: 26:35)



Now the clusters that I have are in step 3, $\{1, 2\}$ forms one cluster, $\{3\}$ is the other cluster, $\{4, 5\}$ that makes the other cluster, okay? Now in the same manner if I compute the distances between each pair of clusters. So we obviously $\{1, 2\}$, $\{1, 2\}$ has the distance equal to zero, so that I don't bother about this, $\{1, 2\}$ and 3 it remains same as 8.1 okay. So $\{1, 2\}$ and 3 the distance remains as 8.1 here also the distance remains as 8.1. when you compute the distance between these two clusters, cluster containing the feature vectors $\{1, 2\}$ and the cluster containing the feature vector $\{4, 5\}$.

So what I have to do is, I have to find out distance between 1, 4, distance between 1, 5, distance between 2, 4 and distance between 2, 5. So I get 4 possible distance values I have to take minimum of these 4 and here this minimum distance will come out to be 16.0, okay. Similarly, here also the distance will come out to be 16.0. Then 3, 3, I don't bother the distance between 3. And 4, 5 you can find out that this is 9.8 so obviously. 5 and 3 this is also 9.8 and here the distance value is zero. So now I have to take the minimum of all of them so the minimum comes out to be this 8.1, okay? So in step 3 my clusters are $\{1, 2, 3\}$, these are combined together and the other cluster consists of $\{4, 5\}$, okay?

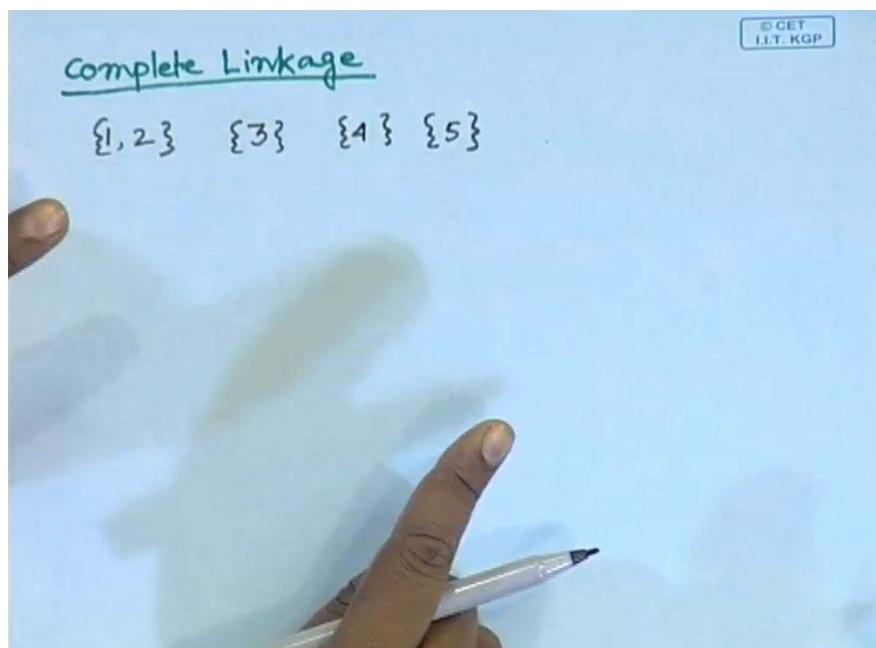
And obviously the final step that we can have in the fourth step now I am left with only two clusters, so these two clusters are to be combined together so I get a single cluster $\{1, 2, 3, 4, 5\}$, okay? So now you see that the initial algorithm that we stated that, if I have N number of feature vectors, I start with N number of clusters and this merging of two nearest clusters this has to be repeated for N number of times or N number of one steps.

So here we started with 4 5 different feature vectors initially taken as 5 different clusters, so after N minus 1 that is after the 4th step all of them have been combined into a single cluster, okay. So if I put it in the form of a tree a bottom-up tree, so I have this cluster number $\{1, 2, 3, 4, 5\}$, okay? So initially what we have done is we have combined feature vector 1 and 2 to give me a single cluster, so that is what we have done in step 1. Step 2 we have combined 4 and 5 into a single cluster, so this is step 2. In step 3 we have combined 1 2 with 3.

So this is what we have done in step number 3 and in step 4 this and this will be combined together to give you a single cluster. So this is after the 4th step, so you find that what I get is a tree structure and as we said that tree structure, this tree structure is a binary tree because activity step we are combining only two of the clusters into a single cluster, okay? And this complete diagram is called a dendrogram.

So if I'm interested in only two clusters and I will stop this agglomeration process at this level, after step 3. If I want three clusters I'll stop it after step 2, okay? If I want four clusters I'll stop it after step 1, so depending upon the number of clusters that I need, I can stop this agglomeration process at the appropriate level, okay. So this is the kind of clustering that we have got using this nearest neighbor or single linkage algorithm.

(Refer Slide Time: 32:35)



If I go for the farthest neighbor or complete linkage algorithm, then let us see that what kind of result that we are going to get. So if I go for complete linkage algorithm, your step 1 will remain the same because I'll have the same sort of distance values for every pair of feature vectors. So my step 1 will remain the same. After step 1, I have to take the minimum distance and using this minimum distance I can combine 1 and 2 only, okay. So clustering up to step 1 will remain same

as what we get in case of single linkage algorithm and then onwards there may be difference, okay?

So after the first step the clusters that I have is $\{1,2\}, \{3\}, \{4\}$ and $\{5\}$, okay,

The distance is minimum when I am computing the distance, I am taking the distance as maximum of the pair of distances, you come to this definition but your similarity is based on minimum distance.

(Refer Slide Time: 33:57)

©CET
I.I.T. KGP

Single Linkage Algorithm (Nearest Neighbor)
Complete Linkage Algorithm (Farthest Neighbor)
Average Linkage Algorithm.

$$D_{SL}(c_i, c_j) = \min_{a \in c_i, b \in c_j} d(a, b)$$
$$D_{CL}(c_i, c_j) = \max_{a \in c_i, b \in c_j} d(a, b)$$
$$D_{Av}(c_i, c_j) = \text{avg } d(a, b) \quad a \in c_i, b \in c_j$$

Come to this definition what is the single linkage distance between C_i and C_j ? The single linkage distance between C_i and C_j is the minimum of this pair of distances, I have to take one feature vector from C_i the other feature vector from C_j , I have to find out what is the distance and this I have to do for every pair of feature vectors with one taken from C_i the other one taken from C_j , okay.

So if I have K number of such pairs, I will have K number of such distances, I have to take the minimum of these distances to give me what is the distance between the clusters C_i and C_k . In case of complete linkage, here I have taken the minimum distance, here I will take it the maximum

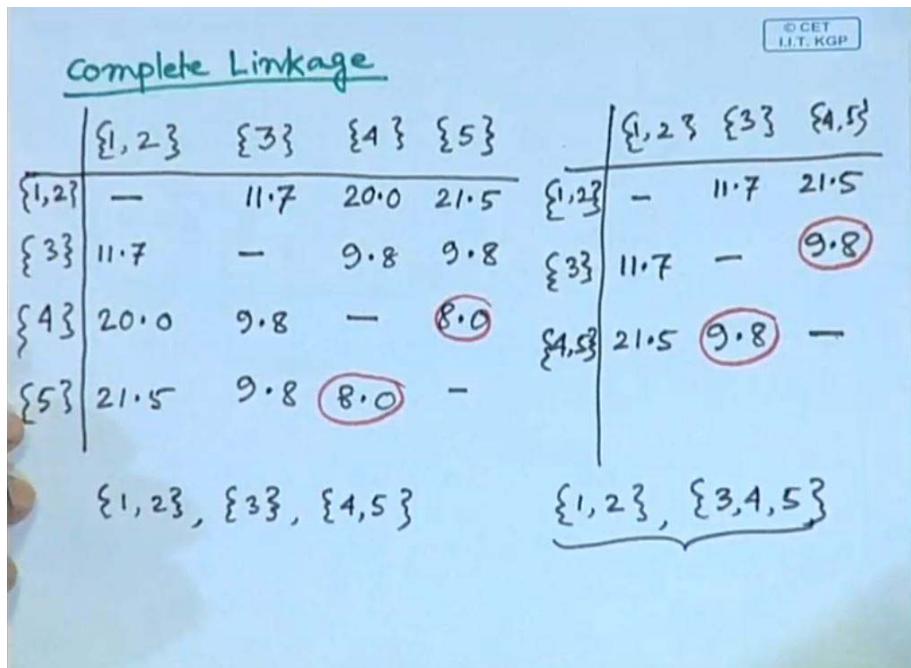
of those K number of distances, so maximum of those K number of distances is with distance between C_i and C_k , in case of average distance it is the average of those K number of distances that will give me the distance between C_i and C_j . So I am taking minimum, maximum or average to compute, what is the distance between the cluster C_i and the cluster C_k , okay?

Now the similarity between cluster C_i and C_j is based on these distance, okay. so I'll say that C_i and C_j are most similar if the distance between C_i and C_j is minimum among the distances between every pair of clusters, is that clear?

So to say whether two clusters are similar or not the distance between those two clusters have to be minimum but it is the distance between the clusters when I compute that computation is one of this, how do you compute the distance between two clusters? The distance between two clusters using the single linkage algorithm is this, using complete linkage algorithm is this, using average linkage algorithm is this and once I find out the distance between two clusters, the clusters having minimum distance they will be most similar, okay.

It is not the maximum distance among the clusters because then the clusters are farthest, so these algorithms single linkage, complete linkage or average linkage this tells how do you compute the distance between the clusters that's all and once I get the distance values then the pair of clusters having the minimum distance in whichever way I compute those, that pair of clusters is the most similar pair of clusters and had to combine them, is that okay? Right.

(Refer Slide Time: 37:15)



So here what I have is, I have 1, 2, 3, 4 and 5 okay. Now the entries in this matrix is likely to be different than what I got in this case okay. Here what I have done is to find out the distance between $\{1, 2\}$ and 3, I have computed the distance between 1, 3 and I've computed the distance between 2, 3 out of these two distances the minimum value I put.

Now out of these two distances, I have to put the maximum value, okay. So come over here what is the distance between 1 2? It is 4.0. Distance between 1 and 3 it is 11.7 and the distance between 2 and 3 it is 8.1, okay? So I have to take the maximum of these two values now, in the earlier case I have taken minimum of these two values. So maximum of these two values is 11.7, right? So over here to have the distance between 1 2 and 3, this distance value is 11.7 in complete linkage algorithm, whereas in single linkage algorithm the distance was the minimum value so it was 8.1. is that okay?

So similarly, obviously 1, 2, the distance is zero, okay? So this is 1, 2, 3, so here also it will be 11.7, right? Now the distance between $\{1, 2\}$ and 4, I have to find out what is the distance between 1 and 4 that is 20.0, 2 and 4 there is 16.0, I to take the maximum of these two distances which is 20.0, okay? So the distance between $\{1, 2\}$ and 4 is 20.0, then distance between $\{1, 2\}$ and 5,

distance between 1 and 5 is 21.5, 2 and 5 is 17.9, I will take maximum of these two distances which is 21.5, so in the same manner this becomes 20.0, this becomes 21.5, okay?

3, 3 distance is zero, 3 4 it remains the same because I have not, this a singleton clusters, so distance between 3 and 4 is 9.8, distance between 3 and 5 is also 9.8. So distance between 3 and 4, 9.8 there is also 9.8, 4 4 is zero, distance between 4 and 5 8.0, it is 8.0, here it is zero, okay? So now what I have to do is, I have to find out what is the minimum of these distances, okay? So what is the minimum of these distances, the minimum distance is in this case 8.0 okay.

So when the distance is 8.0, then naturally I have to combine these two feature vectors 4 and 5. So again so, how I've made a mistake? What is the distance between $\{1,2\}$ and 3, yeah so here also it comes out to be that minimum distance is between 4 and 5, so I have to combine this 4 and 5 so I've got $\{1,2\}$ as before then 3 and then $\{4,5\}$, ok. So this is the same cluster even up to step 2, I get the same set of clusters. So even up to step 2, I get the same result that $\{1,2\}$, $\{3\}$ and $\{4,5\}$ and next I have to compute the same distance between $\{1,2\}$, $\{3\}$ and $\{4,5\}$, okay.

And when you compute this distance you'll find that the distance will be something like this 11.7, 21.5, okay. Then here also it will be 11.7, 21.5 here the distance value will be 9.8 that is 9.8, okay. So following this complete linkage algorithm I get such distance values and the minimum of this is now 9.8, okay. So that means now I combine this particular cluster having these points 4 5 with the set 3, okay. So over here that masters that I've got is $\{1,2\}$ and $\{3,4,5\}$, okay?

And then finally these two clusters will be combined together to give you a single cluster having all the feature vectors in that cluster, is that okay? So here you find that there is difference between what we have got using the single linkage algorithm because in single linkage algorithm we had 1 2 and 3 in 1 cluster then 4 5 in another cluster here we have got $\{1,2\}$ in 1 cluster and $\{3,4,5\}$ in another cluster okay. So up to stage 2, we had the same result but in step 3 onwards our results are different okay. Now what is the conceptual difference between this single linkage algorithm and the complete linkage algorithm.

In case of single linkage algorithm, we have taken the minimum of the distances between pair of feature vectors where one pair feature vector is taken from one cluster or had one feature taken

from one cluster and the other feature vector is taken from the other cluster, okay? Whereas in case of complete linkage algorithm, we have taken the maximum of the distances, so complete linkage algorithm says that it is always guaranteed that the distance between two feature vectors with one feature vector from one cluster and the other feature vector from the other cluster is always bounded, okay.

Because whatever distance value I am taking that is the maximum distance between pair of feature vectors one from one cluster and other from other cluster, I cannot have any other pair having distance more than that. okay. So the clusters that will that you will form using this complete linkage algorithm is more compact, okay. Whereas in case of single linkage algorithm I am taking the minimum distance, so even if I have a linear cluster one linear cluster and another linear cluster, okay, I can have two points, one from one cluster, other from other cluster which are nearest but the maximum distance may be much more than this, okay?

So the cluster which will be formed by the single linkage algorithm will not be that compact as that cluster which is found by complete linkage algorithm, okay. And in case of average linkage algorithm what we have done in this case is that either we have taken the minimum distance to compute the distance between two clusters or the maximum value to compute the distance between two clusters, in case of average linkage algorithm we have to take the average of all the distance values between pairs of feature vectors as the distance between two clusters.

So the average linkage algorithm will give you a compactness which is in between the one which is given by single linkage and the one that is given by complete linkage algorithm, okay. There is another algorithm, a similar agglomerative algorithm but that is not based on simple distance values but it is based on the variance of the samples in a single cluster, okay. So it tries to form the clusters in such a way that the variance of the samples in the same cluster is minimum, okay.

(Refer Slide Time: 47:25)

© CET
I.I.T. KGP

Ward's Algorithm

$C_j \rightarrow m \text{ no. of feature vectors.}$

X_1, X_2, \dots, X_m

$E_j = \sum_{i=1}^m \|X_i - \mu\|^2 = m\sigma^2$

Total Error

$E = \sum_{\forall j} E_j$

And that is an algorithm which is called which is Ward's algorithm, okay. So Ward's algorithm forms the clusters where the variance within the cluster is minimum, okay. So suppose we have a cluster C_j with say M number of feature vectors, okay? we can term them as X_1, X_2 up to X_M , okay. So obviously the error within the cluster that can be defined as E_j which is equal to

$\sum_{i=1}^m \|X_i - \mu\|^2$ okay? Where μ is the mean of all the feature vectors, so this is the error within the j^{th}

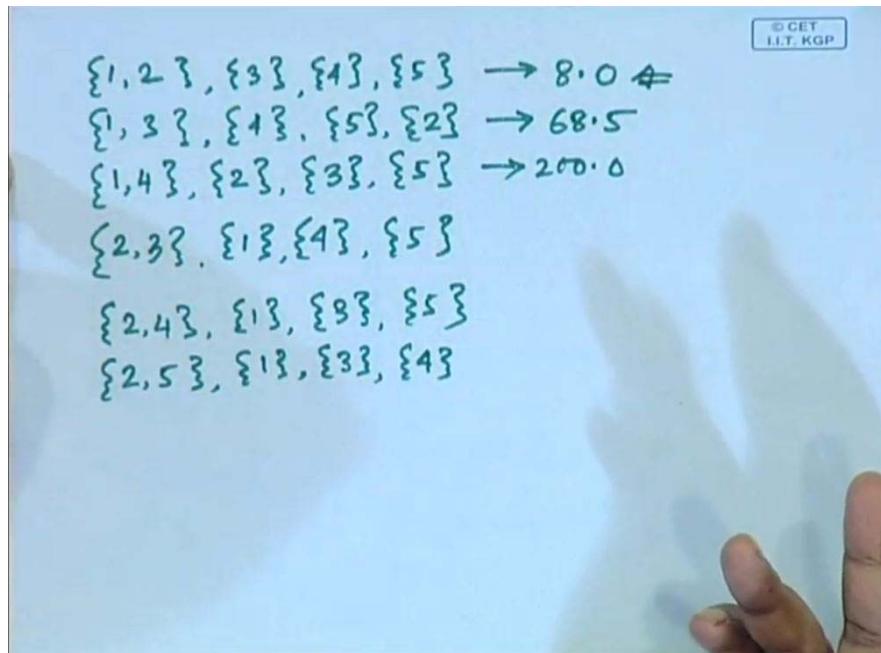
cluster, okay. And you find that this is nothing but $m\sigma^2$ where σ^2 is the variance of all these feature vectors, okay?

So using Ward's algorithm that tries to minimize this or try that that tries to minimize this error, okay? And the total error of clustering E that can be defined as $\sum_{\forall j} E_j$ for all which j , that is the within cluster error sum of within cluster error over all the clusters that gives you the total error of clustering, right?

So in this Ward's algorithm which tries to minimize this variance of individual clusters obviously that also tries to minimize the total error of clustering, so this ensures that when i form the clusters every cluster will be quite compact because only for a compact cluster the error within the cluster

will be minimum, okay? But at the cost of what obviously at the cost of computational complexity because at every step I have to consider every possible combination, so when I want to combine in the first step, when I want to combine two most similar clusters I have to see whether 1 and 2 can be combined together 1 and 3 can be combined together 1 and 4 can be combined together and so on, right?

(Refer Slide Time: 50.:25)



So I have to try with $\{1, 2\}$ in a single cluster, so in the first step I will be I'll have things like this one that will be in a single cluster then 3 will be in a single cluster 4 will be in a single cluster and 5 will be in a single cluster and because these 3 are singleton clusters, so obviously the within class error or within class, within cluster variance for these 3 clusters are zero, okay? So within cluster error will be coming only due to this and you'll find that if I compute following the same set of feature vectors that we have taken this will come out to be something like 8.0, so this is what is within class error or within square error.

Similarly, I also have to try out $\{1, 3\}$, 4, 5 and 2 there is another possible way of combining and if I compute that within class error or the total error in this case this comes out to be 68.5. Similarly

I also have to try out $\{1,4\}$, 2, 3 and 5 this will come out to be something like 200.0, Similarly after so all possible combinations with 1, similarly I have to try out all possible combinations with 2 ,so $\{2,3\}$, 1, 4 5, similarly I have to try up with $\{2,4\}$, 1, 3, 5. I have to try out $\{2,5\}$, 1, 3, 4.

So in this manner I have to try combining two clusters into a single clusters and all possible ways in which two clusters can be combined together and for each of these combinations I have to compute what is the total error and I have to accept that particular combination for which the total error is minimum, okay. And this operation I have to perform N-1 number of times, okay.

So though Ward's algorithm gives you the clusters which are much more compact which is because it is trying to minimize the total error or total within class errors so obviously in terms of error this clustering will give you the better result but obviously at the cost of computation because then amount of computation that I have in this case is quite enormous, okay. So I'll stop this agglomerative clustering with this, next class we will try with other clustering techniques.

Thank you

Pattern Recognition and Application

Prof. P. K. Biswas

Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology, Kharagpur

Lecture - 35

Clustering (contd.)

So we are discussing about the clustering algorithms and in the last class what we discussed is what is called agglomerative clustering and we have seen that agglomerative, agglomerative clustering is nothing but a hierarchical procedure which is a bottom of procedure but in every step two most similar clusters will be merged together to form a single cluster. And we have seen that if there are say N number of points then if you follow this agglomerative clustering but in every step you merge two most similar clusters then after N minus 1 number of steps all the points are all the feature vectors are put into the same group, ok.

So there we can have one of the two approaches that is, we can either decide that what should be the distance threshold between two clusters, so that they can be merged into a single cluster or they're to be separated and considered separately to form two separate clusters. And based on that we can decide about how many clusters we will have where I don't have any control over the number of clusters but it is the distance between the two clusters which will decide that what will be the two clusters or what will be the number of clusters.

The other one can be that we can decide about that how many clusters we want to form out of a given set of feature vectors, so depending upon that we can stop our agglomeration process or merging process at a particular step where I obtain the desired number of clusters. And when we talked about this agglomerative clustering we have also talked about, we talked about three or four different types of agglomerative clustering.

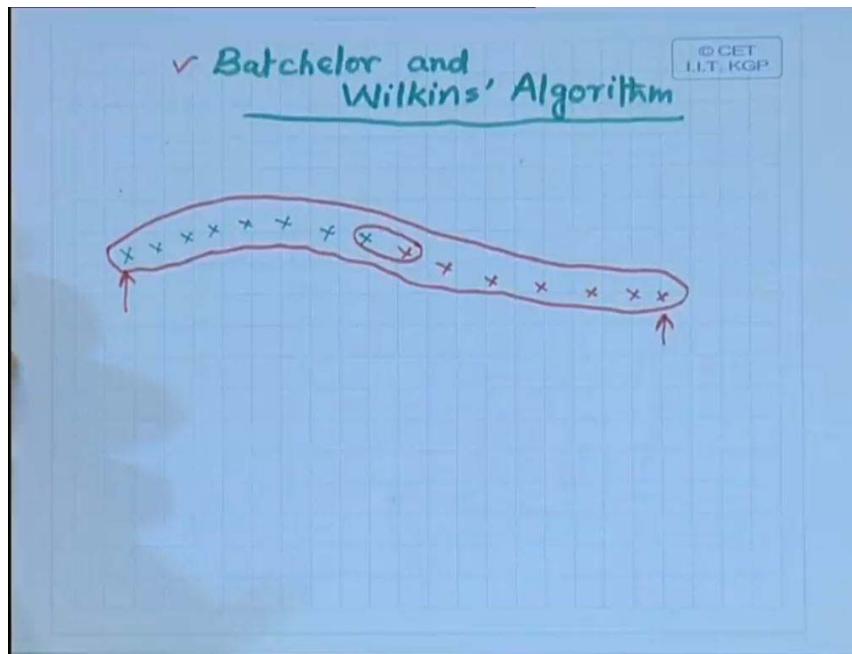
One of them was single linkage. that is the distance between the two clusters are defined to be the distance between two nearest points one from one cluster and the other one from the other cluster, that is what is single linkage clustering. Then other one was complete linkage plastering where the distance between two clusters was defined to be the distance between two farthest points but one is taken from one cluster and the other point is taken from another cluster.

And we had we had also talked about average linkage, where the distance between two clusters is taken as the average distance of the pair of points, where one of the pair is taken from one cluster and the other one of that pair is taken from other cluster, so that gives you the average distance between two clusters. And the other kind of algorithm that we also have talked about is that while doing the agglomerative clustering it tries to find out a cluster where the variance is minimum that is given a mean vector of a particular cluster variance within that cluster is the variance considering

all the points put into that cluster and you have seen like this variance is something which is related to the error within the cluster.

And then we have defined about the total error which is the sum of the errors of individual clusters and this method or Ward's method as we have said tries to form the clusters where the total amount of error of clustering will be minimum, okay? And we have also said that what is the nature of the clusters that we get when I use this different types of clustering say for example in case of single linkage clustering the cluster can be quite spread because the distance between the two clusters is taken as the minimum distance of the pair of points, okay?

(Refer Slide Time: 05:32)



So if I have a linear set of feature vectors say something like this, suppose this is one set of each a vectors and the other feature, set of feature vector is something like this, okay. Then as single linkage algorithm tries to find out the distance between the nearest pair of points. So you find that the distance between these two points will be minimum and this may be below the threshold, okay?

So in which case this entire set of points will be put into a single cluster, okay, which will be avoided in case of complete linkage because complete linkage does to find out what is the distance between this point and this point, okay. So that is the distance between two farthest points so taking that the complete linkage algorithm will try to avoid this sort of clustering rather it will try to find our clusters which are more compact, okay. And the average linkage is something in between the single linkage and the complete linkage and of course what Ward's method tries to find out is also a cluster where the cluster is quite compact, okay.

Now today what we are going to discuss about is, what is called Bachelor and Wilkin's algorithm. So in case of Bachelor and Wilkin's algorithm, it is also a sequential process, what you do is we

take the first feature vector and assume the first feature vector to be one of the cluster centers, okay? Then try to find out the distances of all other feature vectors from that first feature vector.

(Refer Slide Time: 05:50)

$$N \rightarrow x_1, x_2 \dots x_N$$

$$x_1 \rightarrow z_1$$

$$x_i \rightarrow z_2$$

$$\max_j \left\{ \min \left\{ d(x_j, z_1), d(x_j, z_2) \right\} \right\} > \tau d(z_1, z_2)$$

So what you can say is so given a set of feature vectors, say I have N number of each of vectors which is say X_1, X_2 up to X_N . So what I will do is one of these feature vectors I'll take arbitrarily and assume that to be the first cluster Center, so I can assume that the first feature vector which I'll take is X_1 and suppose this point X_1 is the cluster Center or the, of the first cluster which is that one, okay?

Then find out the distance of all other feature vectors from this Z_1 , take the farthest point or the farthest feature vector whose distance comes Z_1 is maximum and assume that to be the second cluster center, okay, that can be any of the other feature vectors from this set, okay?

So suppose that is X_i and say that is my second cluster Center which is Z_2 , now given these two cluster centers you try to find out the distances of all other feature vectors from these two cluster centers right. So given any other points say X_j , I want to find out what is the distance between X_j and Z_1 I'll also find out what is the distance between X_j and Z_2 , okay? I'll consider the distance which is minimum of these two, okay.

So for every point other than these two x_1 and say X_i which happened to be the second cluster Center for all other points I find out the distances from Z_1 and Z_2 and I consider that distance which is minimum among the two and that is computed for all other points, okay. And out of these minimum distances I find out what is the maximum distance, okay.

So it is like first I find out minimum of the $d(X_j, Z_1)$ and the $d(X_j, Z_2)$, okay? Then I find out what is the maximum of these distances where this maximum is over j okay, so whichever X_j gives this maximum value if that distance is greater than an appreciable fraction of the distance between Z_1 and Z_2 , okay. So if this $\max_j \{ \min \{ d(X_j, Z_1), d(X_j, Z_2) \} \}$ then some fraction $\tau d(Z_1, Z_2)$ then the corresponding X_j , I assumed to be the third cluster Center, okay, and this process continues.

So once I get the third cluster Center for all other points, I have to find out what is the distance from Z_1 what is the distance from Z_2 what is the distance from Z_3 which is the third cluster Center and out of these three distances I have to take the minimum, okay. So such minimum distances I have to compute for all other points and out of these minimum distances I have to find out the maximum and if this maximum distance is greater than certain fraction of the minimum of $d(X_j, Z_1)$, $d(X_j, Z_2)$, $d(X_j, Z_3)$ then that corresponding point X_j , I consider to be the fourth cluster center, ok.

So this way it continues until and unless the distance or the maximum of that minimum distances becomes less than that threshold which is defined, okay. So at that point I can stop my clustering and once I stalk my clustering till that point what I have decided is how many clusters centers I have. So once I have those cluster centers then remaining points are simply put into one of those clusters based on minimum distance criteria, okay?

So if I simply go by this distance threshold I don't have any control over the number of clusters that I found or otherwise what I can do is, if I want three clusters I can stop when I get three cluster centers and the remaining points which is put into one of those three clusters based on minimum distance criteria. If I want four clusters I'll stop when I obtain four cluster centers and remaining points will simply be put into one of those four clusters based on minimum distance criteria, okay?

So that is what is called Batchelor and Wilkins algorithm where by defining what will be my stopping criteria, either I can have a predefined a number of clusters or I can have variable number of clusters where I define that what is my criteria for the distance between two cluster centers, okay.

(Refer Slide Time: 11:48)

A	$\rightarrow (1, 1)$
B	$\rightarrow (1, 2)$
C	$\rightarrow (3, 2)$
D	$\rightarrow (3, 3)$
E	$\rightarrow (6, 6)$
F	$\rightarrow (6, 7)$
G	$\rightarrow (7, 6)$
H	$\rightarrow (7, 7)$
L	$\rightarrow (7, 1)$
J	$\rightarrow (8, 2)$
K	$\rightarrow (9, 1)$

So let us take an example suppose I consider a number of points where say point A is given by feature vectors say (1, 1), point B is given by feature vector (1, 2), point C is given by feature vector say (3, 2), point D given by feature vector say (3, 3), point E is given by feature vector say (6, 6), point F is given by feature vector (6, 7), G is given by feature vector say (7, 6). So these are the feature vectors which are to be clustered into various clusters, okay?

And I assume that the first cluster Center I consider is the point A corresponding to the feature vector (1, 1), ok. So as I said that once I use this Batchelor and Wilkin's algorithm from a I have to compute the distances of all other points, okay. And these distances will continue in next steps, so what we can do is we can form a distance matrix where the distance matrix will tell me that what is the distance between a pair of feature vectors, okay.

(Refer Slide Time: 14:03)

Distance Matrix													
A	B	C	D	E	F	G	H	I	J	K	A	H	K
A 0 1.0 2.2 2.8 7.0 7.8 7.8 8.4 8.4 9.0 8.0	B 1.0 0 2.0 2.2 6.4 7.0 7.0 7.8 6.0 7.0 8.0 7.8 8.0 6.0 8.0	C 2.2 0 1.0 5.0 4.9 5.6 6.4 4.1 5.0 6.0 8.0 7.0 8.1 6.4 8.0	D 2.8 2.2 1.0 0 4.2 5.0 5.6 4.5 5.5 6.0 7.3 7.0 8.1 5.6 7.3 8.0	E 7.0 7.8 1.0 5.0 0 1.0 1.0 1.0 1.0 1.0 1.0 7.0 7.8 1.0 1.0 1.0	F 8.0 7.8 1.0 5.6 1.0 0 1.0 1.0 1.0 1.0 1.0 7.8 7.8 1.0 1.0 1.0	G 8.4 8.4 1.0 6.4 1.0 1.0 0 1.0 1.0 1.0 1.0 8.4 8.4 1.0 1.0 1.0	H 8.4 8.4 1.0 6.4 1.0 1.0 1.0 0 1.0 1.0 1.0 8.4 8.4 1.0 1.0 1.0	I 9.0 9.0 1.0 7.0 1.0 1.0 1.0 1.0 0 1.0 1.0 9.0 9.0 1.0 1.0 1.0	J 9.0 9.0 1.0 7.0 1.0 1.0 1.0 1.0 1.0 0 1.0 9.0 9.0 1.0 1.0 1.0	K 8.0 8.0 1.0 7.3 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0 8.0 8.0 1.0 1.0 1.0	A 1.0 0 2.2 2.8 7.0 7.8 7.8 8.4 8.4 9.0 8.0	H 7.8 0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 7.8 0 1.0 1.0 1.0	K 8.0 8.0 0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 8.0 8.0 0 1.0 1.0

$\{A, B, C, D\}; \{H, E, F, G\}; \{K, I, J\}$

So what I will form is I'll form our distance matrix to compute the distances between every pair of feature vectors, okay. So I have feature vectors A, B, C, D, E, F, G, H, I, J and K. Similarly, on the column also write A, B, C, D, E, F, G, H, I, J and K, okay. So naturally the distance between AA same point, so this distance value will be 0, similarly BB this will also be 0, okay. Now you'll find out the distance between A and B and less let us assume that we want to compute the Euclidean distance, okay. So here you'll find that the feature vector A which has the vector 1, 1 and B has the feature vector 1, 2. So naturally that Euclidean distance between A and B is 1, ok.

So the distance between A and B is, let us write 1.0, ok. Similarly, I can also compute the distance between A and C, there is 1, 1 this is 3, 2 I can simply compute the Euclidean distance so likewise I can compute the Euclidean distance between every pair of feature vectors, ok. So if I complete this distance matrix the distance matrix will be something like this between A and C it will be 2.2 I'll go up to the first digit okay then between A and D it will be 2.8, between A and E it is 7.0, it is actually 7.07 but I am putting 7.0, okay.

A and F, if it will be 7.8, between A and G it will be 7.8 again, between A and H, it will be 8.48, A and I it is 6.20, between A and J again 7.07, I am writing 7.0 between A and K it is 8.0, okay. So between B and A it will be same 1.0, between B and C it will be 2.0, between B and D it will be 2.23, I am writing 2.2, between B and D it is 6.4, between B and F it is 7.07, so I am writing 7.0, between B and G again 7.07, between B and H it will be 7.8, B and I it is again 6.08, B and J it is 7.0 and B and K it is 8.06, between C and D is same as A and C so it will be 2.2.

Between B and C it is same as this 2.0, between C and D the distance is 1.0, between C and E the distance is 5.0, between C and F the distance is 4.89, let me write it as 4.9, between C and G in the distance is 5.6, between C and H, the distance is 6.4, between C and I the distance will be 4.1, then the distance is 5.0 and the distance will be 6.08.

Then when I compute the distance from D it is 2.8 then comes 2.2 actually 2.23, then 1.0, then D and E it is 4.2, D and F this will be 5.0, then D and G again it will be 5.0, between D and H the distance will be 5.6, between D and I the distance will be 4.47 so let me write it as 4.5, then 5.09 and then the distance is 6.32. So like this I can complete this distance matrix ok, I'm not going for all the interest and once you complete this distance matrix you have to find out that what we have done is initially you have assumed that my first point A that is the first cluster center Z_1 , ok.

So I have to find out a point a feature vector we just for this from A and the farthest feature vector I will get from the first row, which is having the maximum distance and here you find that out of all these the maximum distance is the distance of the feature vector H, okay. So this becomes my second cluster center, so the first cluster center that I have is the feature vector A and the second cluster center is the feature vector H, ok. So once I have these two feature vectors or these two cluster centers I have to compute the distances of all other points from these two cluster centers

So other points that I have are point B, C, D, E, F, H is a cluster Center, okay. Then G, I, J, K, okay. So if I compute the distance between A and B, the distance I'll simply get from this matrix once I have this complete matrix, I'll get this distance from this matrix itself okay. So the distance between A and B is 1.0 and the distance between B and H again I will get from this matrix which

is 7.8, okay. And as I said that I have to take the minimum of these two and the minimum is nothing but this one, okay. Similarly, I have to compute the distance between C and A, and C and H, okay.

So the distance between C and A will be 2.2 or 2.23, which I put as 2.2 and the distance between C and H again I can compute from here which is 6.4 so this distance is 6.4 again I have to take minimum of these two so the minimum distance is 2.2, okay. Between D and A, and D and H the distances are 2.8 and 5.6, again the minimum distance is 2.8 because I am always considering the minimum of this pair of distances.

Then if I, when I consider point E the distance between E and A is 7.0 and between E and H the distance is 1.41, okay. So here the minimum distance is 1.41 so I have to consider this distance. Similarly, when I compute between F and A in the distance between F and A is 7.8 or 7.81 and here the distance is 1.20 between G and A, the distance again is 7.81 and here the distance is 1.0 between I and A, the distance is 6.0 here also it is 6.0, so you find that both these distances are same that is point I is equidistant from a and H so I can consider any of them.

For J it is 7.07 and it is 5.09, for K it is 8.0 and 6.32. okay. So when I considered all the minimum distances, here the minimum distance is this, here the minimum distance is this here I can take any of them so let me take this one, okay, here the minimum distance is this and here the minimum distance is this, okay? So these are the minimum distances that I have and then what I said is I have to take that point where, for which this minimum distance is maximum, okay?

So if I find that that is point K, so this point K will be taken as the third cluster center, my first cluster Center was A, second cluster Center was the H and the third cluster Center is K, okay? So once I have this third cluster Center let me include this third cluster Center also. So I want to find out the distances from all these points, of all these points from the point K as well, okay.

So when I do that the distance between B and K; I already have distance between B and A and B and H, okay, so I have to compute the distance between B and K and distance between B and K that comes out to be 8.06, distance between C and K that comes out to be 6.08, between D and K the distance comes out to be 6.32, E and K it comes out to be 5.83, between F and K it comes out to be 6.70, between G and K that distance comes out to be 5.38 between I and K the distance comes out to be 2.0, between G and K the distance comes out to be 1.41, okay.

So now find that once I have all these different in these different distances now again when I take the minimum in each, for each of these distances in the first case the minimum remains as 1.0 minimum of all these three distances, in the second case also the minimum remains to be 2.2, third case also the minimum remains to be 2.8 here the minimum remains to be 1.4, here the minimum remains to be 1.0, here also it is 1.0, here now 6.0 is no more the minimum distance but the minimum distance is 2.0.

So this is not the minimum anymore the minimum distances 2.0, right? Similarly, over here the minimum distance is 1.41 okay. This is no more important because K has already become a cluster Center so if I decide that I'll stop my clustering at this point when I have three different clusters then I have one cluster Center which is A, the second cluster Center which is H and the third cluster

Center which is K. So given these three cluster centers I have to cluster the remaining points to one of these clusters whichever is nearest, okay.

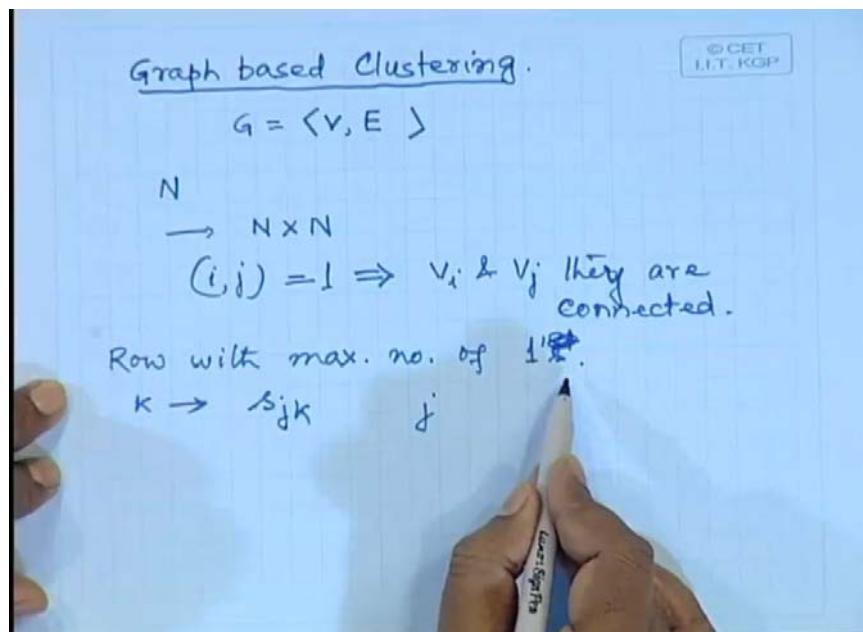
So naturally that information I get from this particular table, the distance between B and A is minimum among these three distances so naturally B will be put into cluster formed by this cluster Center A. Similarly, C will be put into the cluster for my cluster Center A, D will also be put into the cluster formed by cluster Center A. Coming to E you find that the distance between E and H is minimum, okay. So the point E will be put into the cluster formed by cluster Center H.

F similarly we will go to the same cluster, G will also go to the same cluster, okay. Coming to point I, the distance between I and K that is minimum, so point I will go to the cluster of cluster Center K, similarly point J also will go to the cluster found by cluster Center K, okay. So I get three different clusters one cluster contains point A, B, C and D this becomes one cluster, the second cluster is H, E, F and G and the third cluster is K, I, L and J. okay.

So these are the three clusters that is found by following this Batchelor and Wilkin's algorithm. Now here you find that when I have used to this Batchelor and Wilkins algorithm, the Batchelor and Wilkins algorithm the output is likely to depend on the first feature vector which I used as cluster, cluster Center because instead of using A as the first feature vector if I use any other points say E then the clustering output might have been different, different from these three different clusters, okay.

So because it is a sequential algorithm so naturally the final output that I get is likely to depend on the type of or which of the feature vectors is taken as the first feature vector or taken as the first cluster Center because all other cluster centers are decided based on the first cluster Center, okay? So that is one of the drawback of this Batchelor and Wilkins algorithm.

(Refer Slide Time: 31:31)



Now the other algorithm which can be formed out of this distance matrix that is a graph based algorithm, or graph-based clustering, okay? Now what is this graph-based clustering? I hope all of you know what is graph, okay. A graph G is defined as collection of vertices and edges or collection of nodes and edges, so what I can do is given the feature vectors, every feature vector can be considered to be a vertex or a node in the graph, okay.

And two nodes in the graph will have an edge or there will be a connectivity between one node and another node depending upon the distance between those two nodes. So if I put some threshold on that distance then I can say whether the two nodes are connected or two nodes such things are not, okay. If the distance is more than the threshold I'll say that the two nodes are not similar that means those two nodes are not connected so accordingly in the set of edges, I will not have an edge in between those two vertices, okay.

And there are various ways of representing a graph, one of the way of representation of a graph is by a similarity matrix, okay. So similarity matrix is again a two dimensional matrix so if I have in a number of nodes in the graph then the similarity matrix is an $N \times N$ matrix, okay. Where in every element in this $N \times N$ matrix, say if for this matrix the element i, j that is equal to one, this indicates that node V_i and node V_j they are connected, okay.

And in our case if node V_i and node V_j are connected that means the feature vector V_i and the feature vector V_j they're similar. So if they are similar they should be put into the same cluster, okay. So for this graph-based clustering algorithm, the graphical representation of the feature vectors that we have that will be represented by a similarity matrix, where an element in the similarity matrix can have a value either 0 or 1.

So if the two feature vectors are similar or the distance between them is less than certain threshold then I'll say that those two feature vectors are similar and accordingly I'll have an edge between those two nodes and if the distance between those two feature vectors is greater than the threshold then I will say that those two nodes or those two feature vectors are not connected and correspondingly the element within the similarity matrix will be equal to 0, okay.

So when I go for clustering using this similarity matrix what can be done is, I can take a row in this similarity matrix which has got maximum number of ones, that is maximum number of which are vectors are put into a single cluster, okay. So I will take a row with maximum number of ones, okay and I consider all other nodes, okay. Say a point or a feature vector key will be put into the same cluster if I find that in the similarity matrix $S_{jk} = 1$. where the j^{th} feature vector is already in the cluster.

So that is the relation of transitivity, okay. So if j^{th} feature vector is similar to I , similar to a feature vector I and K feature vector is similar to j^{th} then by transitivity the K feature vector we is also similar to i^{th} feature vector, okay. So when I take a row with maximum number of ones and all the corresponding feature vectors are put into a single cluster, okay. Then I consider every point in that cluster come to the corresponding row in my similarity matrix all other locations or all other points which has got a one in the corresponding row those feature vectors will also be put into the same cluster.

And following this, once I cannot do the cluster anymore, then one of the cluster formation is complete, okay. Then I have to repeat the same operation with the other rows and columns of that similarity matrix. So that rows and columns corresponding to the feature vectors which are already put into a cluster they will be removed and with the remaining rows and columns, I have to repeat the same operation to form the next cluster and this will continue until and unless all rows and columns in your similarity matrix is complete, okay.

So if I take the same example, that is the same set of feature vectors which I have considered for Batchelor and Wilkins algorithm, okay. And once I form for these feature vectors this type of distance matrix it is quite straightforward to form the similarity matrix from this distance matrix because within this any distance value which is less than or equal to threshold, the corresponding element in my similarity matrix will be equal to one.

(Refer Slide Time: 38:01)

Distance Matrix

© CET
I.I.T. KGP

	A	B	C	D	E	F	G	H	I	J	K
A	0	1.0	2.2	2.8	7.0	7.8	7.8	8.48	6.07	0.8	0
B	1.0	0	2.0	2.6	4.7	0.7	0	7.8	6.07	0.8	0.6
C	2.2	2.0	0	1.0	5.0	4.95	6.64	4.15	0.6	0.08	0
D	2.8	2.2	1.0	0	4.25	5.05	5.05	6.45	5.09	6.3	0
E	7.0	7.8	5.0	4.95	0	1.0	1.0	1.0	1.0	1.0	0
F	6.07	7.81	5.6	5.6	1.0	0	1.0	1.0	1.0	1.0	0
G	0.8	7.81	6.4	6.4	1.0	1.0	0	1.0	1.0	1.0	0
H	0	0	6.4	6.4	1.0	1.0	1.0	0	1.0	1.0	0
I	0	0	0	6.0	6.0	6.0	6.0	0	1.0	1.0	0
J	0	0	0	0	0	5.09	5.09	1.0	1.0	1.0	0
K	0	0	0	0	0	0	6.32	6.32	1.0	1.0	0

$\{A, B, C, D\}; \{H, E, F, G\}; \{K, I, J\}$

Wherever the distance is greater than the threshold, the corresponding element in the similarity matrix will be equal to 0, so it is quite straight forward. So if I form this distance matrix I could not, I did not complete, okay. You can find out the distance between pair of feature vectors and complete this distance matrix. Once you have this distance matrix then I can form a similarity matrix for the graphical representation of the same set of feature vectors.

(Refer Slide Time: 38: 17)

Similarity Matrix: $d(x_i, x_j) \leq 2 \Rightarrow S_{ij} = 1$												
{A, B, C, D}		A	B	C	D	E	F	G	H	I	J	K
→ {A, B, C, D}		1	1	0	0	0	0	0	0	0	0	0
→ {E, F, G, H}		0	1	1	1	0	0	0	0	0	0	0
→ {I, J, K}		0	0	0	0	1	1	1	1	0	0	0
{E, F, G, H}		0	0	0	0	1	1	1	1	0	0	0
{I, J, K}		0	0	0	0	0	0	0	0	1	1	1
{A, B, C, D}		0	0	0	0	1	1	1	1	0	0	0
{E, F, G, H}		0	0	0	0	0	0	0	0	1	1	1
{I, J, K}		0	0	0	0	0	0	0	0	1	1	1
{A, B, C, D}		0	0	0	0	0	0	0	0	1	1	1

So let me form, write this similarity matrix, so I have the feature vectors A, B, C, D, E, F, G, H, I, J, K, so let me complete this similarity matrix. So as before the distance between when I take this pair AA, obviously the distance is 0, okay. So by default the corresponding elements in the similarity matrix will be equal to one because here the distance values 0, obviously it is less than or equal to threshold, okay.

Now while forming the similarity matrix from this distance matrix, I assume my threshold to be equal to 2, that is if the distance between two points x_i and x_j is less than or equal to 2, then correspondingly S_{ij} will be equal to one, okay. So my threshold I am taking equal to 2, so considering this your similarity matrix will be something like this one.

So the similarity matrix will be something like this, so as I said when you start your clustering operation, clustering algorithm, I have to consider a row within this similarity matrix which has got maximum number of ones, okay. So find that within this the first one which has got maximum number of ones is the row E, okay, which has got four ones, okay.

And here E, F, G and H all these will be put into a single cluster. okay. So E, F, G and H all of them will be put into a single cluster, come to the row corresponding to F all the corresponding feature vectors within this row where I have one in the similarity matrix they will be put into the same cluster, okay. But you find that all the points in corresponding to which this entry is equal to one, they're already put into the same cluster so nothing more is to be added, next you come to the row corresponding to G here also the same thing all these points are already in the cluster, so nothing more is to be added.

Come to the row corresponding to H, again I have the same situation that all these points are put into the same cluster, so nothing moves to be added, okay. And here I have considered the rows corresponding to all other points which are put into the same cluster that means one clustering is already complete, okay. I cannot put any other point into the same cluster because following these points I cannot move to any other row.

So this forms one of the clusters, once that is done I remove all the rows and columns corresponding to these nodes or these vertices which are already put into a single cluster. So the rows corresponding to E, F, G and H, they are to be removed from this similarity matrix. So I will remove all these rows and I'll remove all the columns corresponding to E, F, G and H.

So I don't have to consider these doors at columns for anymore clustering operation. Next out of the remaining rows and columns I have to repeat the same operation, so I have to find out which of these rows contains maximum number of ones. So if I start from the beginning you find that row B it contains three ones, which is maximum.

So in the second cluster I'll have B, A and C, so A, B and C they'll be put in one cluster and I have considered row B, so this row the row corresponding to B is already considered I have to come to row corresponding to this feature vector A, okay. And I have to put into the same cluster all the points where I have a one in this row, so those points are A and B which are already included. So from this row of A, I cannot add any other feature vector into this cluster. Come to the row corresponding to feature vector C, you'll find that B is already included C is already included, there is another feature vector which is not included, so that is equal to D, so I have to include D into the same cluster, okay.

And once D is included then I have to come to the row corresponding to D where C and D which are already included, okay. So this cluster cannot be expanded any further, so that second cluster which is formed is A, B, C and D, okay. So once I get this cluster, I have to remove the rows and columns corresponding to A, B, C and D from the feature vector, so what is remaining is this, okay. So you can find that following the same logic, all these points will be put into the same cluster so the third cluster is formed by I, J and K, okay. So I get three clusters, one cluster contains A, B, C, D, the second cluster contains point E, F, G, H and the third cluster contains points I, J and K.

Now incidentally, this cluster that I have got is same as the cluster that was obtained using Batchelor and Wilkins algorithm but this is accidental. I may not get the same one, okay. But one thing is guaranteed in case of this graph-based clustering is that because when i go for this graph based clustering approach, I am considering all the points together.

It is not a sequential one because when I form the similarity matrix, the similarity matrix is formed by considering all the points together. That means the output of the graph based algorithm once I decide about this threshold, is unique, okay. Just for algorithmic convenience, we have said that we have to first consider the row which has got maximum number of ones but even if I don't consider the row which has got maximum number of one's first, my final output will not change.

Even if I had considered this row A first, say A and B will be put into the same cluster, come to point B, point C will be put into the same cluster A, B, C. Come to point, come to the row corresponding to C, D will also be put into the same cluster. Come to that row corresponding to this the feature vector D, I cannot put any other point in to this cluster. So A, B, C, D becomes one cluster.

You remove the rows corresponding to sat feature vectors A, B, C and D. Next you come to point E and following this, this is the three cluster that I can form I cannot form any other cluster remove all the rows and columns corresponding to E, F, G and H what will be left out is only this, okay. So these remaining three points I, J and K that from another cluster

So whichever row I considered first out of this similarity matrix to start my clustering operation that does not influence what will be my final set of clusters because this clustering is done using the similarity matrix, where similarity matrix is a global information for the graphical representation of the given set of feature vectors. So my clustering output will remain the same, what will matter is the distance threshold because depending upon the distance threshold, I'll have different types of similarity matrix, okay.

So instead of taking threshold value equal to 2, if I take this value equal to one I would have got more number of clusters, okay. If I increase this instead of 2, if I take this threshold to be 5 then I'll have more number of ones in the similarity matrix okay and if I get more number of ones in the similarity matrix that means in a single cluster I'll accumulate more and more number of points. So effectively the number of clusters will reduce where the size of individual cluster will be quite big.

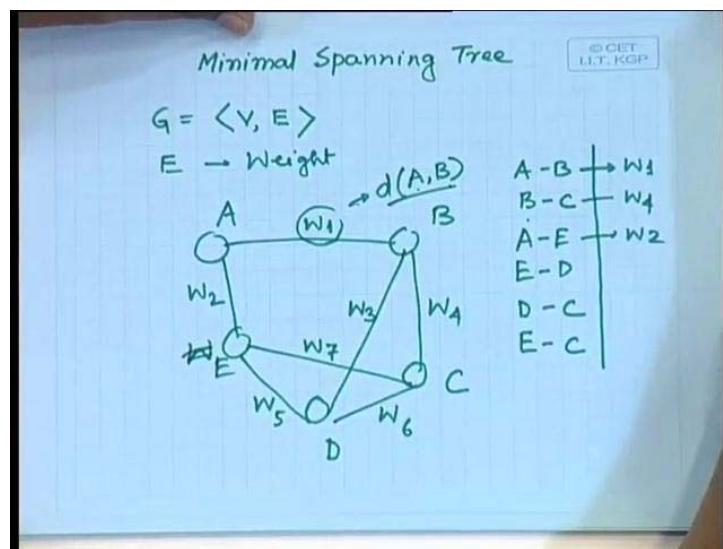
So if I go for this graph based algorithm, what decides the final cluster output is the threshold that I considered, it does not depend upon in which order I go for clustering the points unlike in case of Batchelor and Wilkins algorithm, okay. So I'll stop here today, next day I'll start with some other algorithm for clustering techniques.

Pattern Recognition and Application
Prof. P.K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 36
Clustering Using Minimal Spanning Tree

Good morning. So, in the last class we have talked about the clustering using Batchelor and Wilkins algorithm, and then we have talked about a graphical approach, where the graph is represented by an adjacent matrix. And from the adjacent symmetric we have seen that how we can cluster the given set of feature points. So, today we will discuss about another robust technique, which is spanning tree based clustering technique. So, when we talk about spanning tree or in our case we will be talking about minimal spanning tree. A spanning tree is nothing but a tree representation of a graph. So, you know that a difference between a graph and a tree. In case of a graph, you can have a cycles in a tree, you cannot have a cycle. So, given two nodes I can have only one path between two given nodes. I cannot have multiple path between two given nodes, which is allowed in case of a graphical representation. So, when we talked about the minimal spanning tree, there, the graph is represented as an weighted graph.

(Refer Slide Time: 01:32)



Weighted graph in the sense, when I say that of weight edges, every edge has a weight or a cost associated with the edge. So, if I have a graph something like this, say having a number

of nodes. So, if this is a graphical representation say any arbitrary curve something like this, the nodes are A, B, C, D, E. Then every node will have a weight associated with it, say W_1 , W_2 , W_3 , W_4 , W_5 , W_6 , W_7 and so on.

So, when we talk about the minimal spanning tree, as we said that a graph can be represented by, in various ways. One of the representation that we have discussed in the last class is the adjacent symmetric representation. That means if there is an edge between two nodes in a graph then the corresponding element in the adjacent symmetric will be equal to 1. If the nodes are not connected, the corresponding elements in the adjacent symmetric will be equal to 0. In case of a weighted graph, I can have a matrix representation, where in the corresponding elements in the matrix I will have a value corresponding to weight of the corresponding edge. The other form of representation of the graph is by an edge list.

So, wherever I have an edge, I have edge between A and B. So, this graph can be represented in the form AB, then BC, then let us say this is E, then AE, then ED, DC and we have EC. So, I have the set of edges and this set of edges along with the vertices represent the graph. When I consider a weighted edge, then for every edge I will have the corresponding weight. So, this will have W_1 . Similarly, BC will have W_4 , AE will have W_2 and so on, ok?

So, given a set of feature vectors, what I have to first to do is I have to represent that feature vector in the form of a weighted graph. So, if I assume that the graph is completely connected between, that means in every pair of nodes I have an edge, ok? And the weight of the edge will be the distance between those two vertices or those two feature vectors, corresponding feature vectors. So, in our case this weight W is nothing but the distance between the nodes A and B, where node A represents one feature vector, node B also represents another feature vector. So, it is the distance between feature vectors A and B which represents, which will represent the weight of the corresponding edge, ok?

So, if I represent the set of feature vectors by a completely connected graph. Completely connected graph means between every pair of nodes I will have an edge and the corresponding edge will have a weight, which is nothing but the distance between the corresponding vertices, which are connected by the edge and this distance can be various distance measures. We will assume that Euclidean distance between the two vertices, ok?

So, given this how we can have a minimal spanning tree representation of this graph? The minimal spanning tree says that when I represent, when I find out the spanning tree corresponding to a graph, then the sum of the weights of the edges which are included in the

spanning tree. So, you find that the spanning tree representation of this graph will obviously have all the vertices of the graph, but the edges that will be included in the spanning tree that will be a subset of the edges, which are there in the graph. So, when I consider that subset of the edges, the minimal spanning tree says that the sum of the weights of the edges which are included in the spanning tree that should be minimum. So, that is what is minimal spanning tree?

So, a minimal spanning tree algorithm can be very easily find out, that when I have this edge list. So, here we assume that we will use a representation of the graph which is an edge representation and every edge has a corresponding weight. So, when I have this edge list, this edges will be ordered in the ascending order of the weights. So, the first edge in my list will have the minimum weight and the last edge will correspond to maximum weight and in between the edges will be ordered in ascending order of the corresponding weights, ok? So, I will again take an example to explain how this minimal spanning tree can be constructed from a given graph and how this minimal spanning tree can be used for clustering purpose.

(Refer Slide Time: 07:23)

The whiteboard displays a list of feature vectors and a mathematical calculation:

- Feature vectors listed on the left:
 - A (1, 1)
 - B (1, 2)
 - C (3, 2)
 - D (3, 3)
 - E (6, 6)
 - F (6, 7)
 - G (7, 6)
 - H (7, 7)
- On the right, a calculation is shown for the number of edges in a complete graph with 8 nodes (labeled g_{C_2}):

$$g_{C_2} = \frac{8!}{2! 6!}$$

$$= \frac{8 \times 7}{2}$$

$$= \underline{\underline{28}}$$

So, I will use a set of feature vectors, say feature vector A which is (1, 1), feature vector B which is say (1, 2), feature vector C which is say (3, 2), feature vector D which is (3, 3), E which is (6, 6), F is (6, 7), G is (7, 6), H which is (7, 7). So, I will consider this eight feature vectors and as I said that when I represent this feature vectors by graph or a completely connected graph and that to an weighted completely connected graph. Then obviously I will have eight nodes. And the number of edges because the graph is completely connected is

nothing but 8C_2 and which you can compare, that this is the $\frac{8!}{2!6!}$, which will be $\frac{8 \times 7}{2}$, it is equal to 28.

So, maximum I can have 28 edges within this graph as the graph is completely connected, and for each of this edge I will have a weight which is nothing but the Euclidean distance between the corresponding feature vectors. So, accordingly as we were doing in the last class we can form a distance matrix or in this case it is an weight matrix.

(Refer Slide Time: 09:13)

Distance / Weight Matrix.								
	A	B	C	D	E	F	G	H
A	0	1.0	2.23	2.82	7.07	7.81	7.81	8.48
B	1.0	0	2.0	2.23	6.40	7.07	7.07	7.81
C	2.23	2.0	0	1.0	5.0	4.89	5.65	6.40
D	2.82	2.23	1.0	0	4.24	5.0	5.0	5.65
E	7.07	6.40	5.0	4.25	0	1.0	1.0	1.41
F	7.81	7.07	4.89	5.0	1.0	0	1.41	1.0
G	7.81	7.07	5.65	5.0	1.0	1.41	0	1.0
H	8.48	7.81	6.40	5.65	1.41	1.0	1.0	0

So, given these feature vectors I can find out what will be the weight matrix for this particular graphical representation. So, this matrix gives you the weight matrix. So, you find that the this is the same one which we were trying to consider in the last class, that is the distance between AB is 1.0, distance between AC is 2.23, distance between AD is 2.82. So, the edge between A and B will have a corresponding weight, which is 1 and edge between A and C will have the weight, which is equal to 2.23. Edge between A and D will have the weight which is 2.82. Edge between A and E will have the weight 7.07 and so on, ok?

So, I can find out the edges within the graph using this particular weight matrix. And once I have this weight matrix, from this weight matrix I can find out the ordered list of the edges which are put in ascending order of weights, ok? So, as I said that the weight between these two vertices A and B is 1.0 and when you look at this weight matrix, you find that this is the minimum weight. Similarly, I will have other edges where also the weight is 1.0 say for

example, here E and F, the edge between them also has a weight 1.0. Edge between E and G, that also has a weight 1.0, ok?

(Refer Slide Time: 10:55)

Ordered Edge List			
A-B	1.0	B-D	2.23
C-D	1.0	A-D	2.82
E-F	1.0	D-E	4.24
E-G	1.0	C-F	4.89
F-H	1.0	C-E	5.0
G-H	1.0	D-F	5.0
E-H	1.41	D-G	5.0
F-G	1.41	C-G	5.65
B-C	2.0	D-H	5.65
A-C	2.23	B-E	6.40

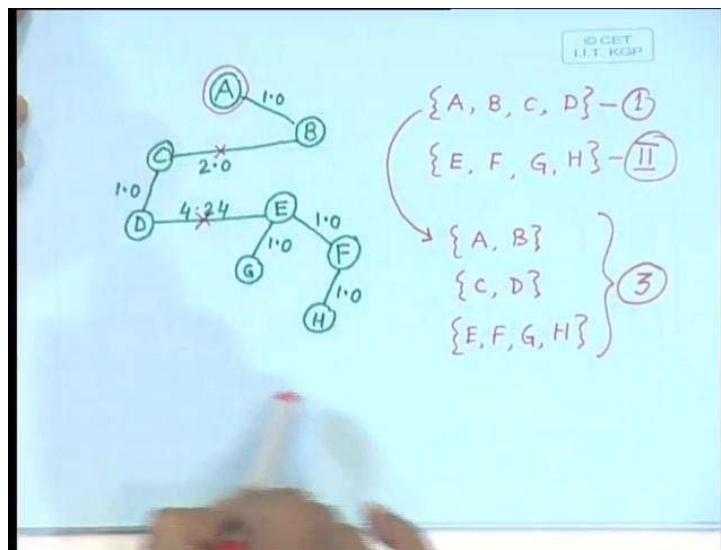
So similarly, I can find out an edge list and based from the weights I can put the edges in ascending order of the corresponding weight values. So, I get the edge list like this AB is 1.0, CD is 1.0, EF is 1.0, like this EH which is 1.41, FG is 1.41, BC is 2.0, AC is 2.23 and so on, ok? So, once I have this ordered layered edge list using these edges, I have to form the spanning tree. So, what is the algorithm for finding out the spanning tree? The algorithm says that you take the first edge because initially my tree is empty to take the first edge. One of the edges you take as root of the spanning tree. One of the vertices you take as root of the spanning tree and the other vertex is connected by an edge, whose weight is same as the weight of the edge as in the graph. Next, we consider the next stage if they have a common vertex, then you simply add another node which are connected to one other edges of already existing tree by the edge that we are considering.

Now, while doing so it may happen that I can create number of trees, because the first edge and the second edge they may not be connected. So, effectively what I will have is two trees, but subsequently when one of the edges connecting one of the nodes of one tree and other node of the other tree, you encounter that you connect those two trees by the corresponding edge. And while forming this tree, always you have to find that if you encounter an edge which is already placed in one of the trees. Then you should not consider that edge anymore,

not only that if any edge produces a cycle within the tree you should not consider that edge also, ok?

So, following this rules you can form the spanning tree and as in this case the spanning tree is formed by considering the edges. First which have got the minimum weights so that will ensure that the spanning tree that you construct that will also be a minimal spanning tree. So, from this example let us try to construct the spanning tree. So, the first edge that I have is an edge AB which has a weight 1.6. So, I will take this particular edge to start my spanning tree and I assume that out of this edge let A be the root node, ok?

(Refer Slide Time: 13:52)



So, I will have node A as the root node and AB forms a tree. Where the weight of this edge is 1.0, the next edge is CD, you find that you do not have any common vertex. So, CD will form another tree whose weight is also 1.0. So, let me take CD somewhere over here, it is weight is also 1.0. The next one is EF, again there is no common node of EF with AB or CD. So, this will be another tree. So, EF let me consider this EF, this edge is also weight of this edge is also 1.0, then EG. Now, you find that between EF and EG the node E is common. So, I will include this EG and connect it to this existing tree by this edge EG and its weight is also 1.0. The next one is FH whose weight is also 1.0, F I already have. So, this is another node FH and this weight is also 1.0.

Next, what I have is FG. So, here you find that if I consider this edge F and G, because both the nodes are already in one existing tree. So, this will form a cycle. So, I should not consider this node at all. The next one is BC, so you find that C is in one tree and B is in

another tree. So, I should connect these two trees using this edge BC and the weight of this edge BC is 2.0.

The next one is AC, you find that both A and C after inclusion of this edge B and C, both A and C are in the same tree. So, if I connect this A and C by this edge, in that case there will be a cycle within the tree. So, I should not consider this edge A and C, next comes BD. So, by using BD again I have similar equation, if I connect B and D there will be a cycle. So, I should not connect B and D.

Next comes AD, again the same situation if I connect A and D there will be a cycle. So, it will not remain a tree anymore. So, I should not connect AD, next comes DE and here you find that D is in one tree and E is in another tree. So, I should consider this node, this particular edge DE and connect D and E by the edge DE and you find that weight of this edge is 4.24. So, that is the weight of this edge and here you find that we started with eight nodes, all the eight nodes have been put in this particular tree. I can go on checking each and every other edge, but if I try to include any other edge within this tree, there will be a circle.

So, I can stop creation of the tree when I have a single tree and all the nodes in the graph are included in that single tree, so that tree formation can be stopped at that particular point. So, here what I have is, I have formed a tree where node A is the root node. This is the root node and if you take any pair of vertices or any pair of nodes, you will find only one path existing between these pair nodes and as the edges have been considered in ascending order of the weight values. So, this algorithm shows that the sum of the weights which are included in this tree, that will be the minimum possible sum. Do you have any question? So, that is the minimum possible sum.

However, this minimal spanning tree is not unique, that is the reason why I said minimal, not minimum. The reason is I started with the node A as the root node if I take any other node as the root node. Let us start formation of the tree with any other node of the root node, then I can have another form of tree, but this ensures that assuming one of the node to be root node, your spanning tree will be unique, but for various nodes as root nodes, the spanning tree will be different.

So, that is why it is the minimal spanning tree. Now, the question is once I have this minimal spanning tree, then how this minimal spanning tree can be used for the

clustering purpose? The operation is very simple, you just check the weight of the edges within this minimal spanning tree and because it is a tree structure, it is quite obvious, that if I remove any of the edges then the tree will be broken into two sub trees.

If I remove two edges the tree will be broken into three sub trees and so on. Unlike in case of a graph, if I have a completely connected graph, removal of one the edges does not ensure that the graph will be partitioned into two unconnected sub graphs, which is not true in case of a graph. But in case of a tree because there is no cycle, if I remove any of the edges then the tree will be partitioned in two subtrees and so on.

So, by deleting the number of edges I can attain the desired number clusters. So, what you can do once I have this minimal spanning tree is that you find out an edge whose weight is maximum. And as I said that in this, in our case the weight of an edge is nothing but the distance between the corresponding feature points or the distance between the corresponding nodes. So, it simply says the weight represents that what is the degree of dissimilarity between two different points. So, if I choose an edge whose weight is maximum or whose cost is maximum that simply says that the distance between the corresponding vertices is maximum.

So, coming to this spanning tree you find that the maximum weight of an edge in this spanning tree is 4.24. So, if I remove this edge then you find that the tree is partitioned into two sub trees. The nodes ABCD the other tree contains, the other sub tree contains EFG and H. So, obviously I get two clusters, one cluster containing nodes A B C D. This is my cluster number 1 and the other cluster is EFG and H, this is cluster number 2. If I want to go further subdivided, this into more number of clusters then within each of them you find out which is the next maximum.

So, here you find that the next maximum weight is between edges C and B which is equal 2. So, I can remove this, if I remove this then this cluster ABCD is now broken into two clusters, one containing AB, the other containing CD. So, I have cluster containing feature vectors AB, CD and the other cluster remains as it is because here none of the edges is having the same value, ok?

So, other cluster is EFG and H, so total I have three number of clusters in this case. So, I can approach one of the two criteria's, that is I can say that this is the number of clusters I want or minimum number of clusters that I want. Or I can also say that what is the distance between two given clusters I should tolerate, so that those sets of points can be put into a

same cluster. So, like in this case when I consider this cluster considering EFG edge and I consider this cluster ABCD, you find the distance between these two clusters is 4.24.

Similarly, over here between the cluster AB and cluster CD, the distance between these two cluster is 2. So, I can have either or I can either select the number of clusters that I want or I can also select that what is the minimum distance between the two clusters which can be tolerated. So, I can either have distance ratio or weight ratio or I can have number of clusters.

So, accordingly I can form the number clusters. So, here you find that because it is a graphical technique and the graph is a global approach. So, the number of clusters that you get or the points which are put in the clusters, will also be independent of order in which you consider the points unlike in case of any other sequential algorithms, pardon cluster 1 and cluster 3. That will obviously be more than 4.24, the reason is when have I have constructed this spanning tree, all the edges are taken are considered in ascending order of weight values. So, had there been any edge between any of the points or any of the nodes A and any of the nodes, within this cluster, had there been any edge whose weight was less than 4.24, that edge would have been considered before this edge, is that ok?

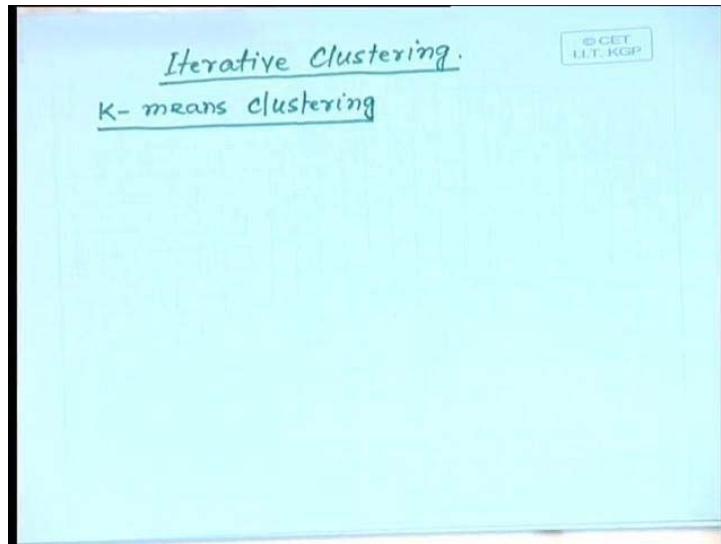
So, that is quite obvious that, obviously there is an edge between A and E. There is an edge between A and F, edge A and H, there is an edge, but the weight of this edge weight of such edges is much more has to be more than 4.24. You come to this weight matrix. Let us take AE, see the weight is 7.07. Consider AF, weight is 7.81. Similarly, AG 7.81. AH is 8.48. It has to more than four point, this 4.24. Otherwise, those edges would have been considered before this edge, ok? So, naturally the distance between this cluster AB and the cluster EFGH will be more than 4.24, that is how this spanning tree is constant, is that ok, correct?

During the course of formulation of spanning tree, I do not know that has to be thought, not necessarily, see what can be done is if I have as you have done in this case. We have put a threshold on value. So, while construction of the spanning tree, because I know the maximum weighted edge is the last one which is been considered while forming this minimum spanning tree. So, if I put a threshold on the weight, so the moment I get an edge of a particular weight I will stop construction of the spanning tree, but even then the problem is there might be other vertices which has not been included in the tree at all.

So, if there are more number of edges whose weights might have been more than 4.24. So, if I put a threshold on this 4.24 when this edge is considered, I stop construction of the spanning

tree. There might other vertices which has not been included the spanning tree at all, ok? So, then you have to go on checking whether every node has been considered. Is that okay, right? So these are the graph based techniques for clustering operation. Now, let us talk about one iterative technique, iterative clustering.

(Refer Slide Time: 29:10)



So, one of the very popular iterative cluster technique is called k-means clustering. So, it is, concept is very simple, I assume that whatever be the set of given feature points I will construct k number of clusters out of this feature points. So, the approach is something like this. Initially or arbitrarily partitioned the set of data into k number of clusters, so that is quite arbitrary once you partition the data into k number of clusters. For every cluster you find out the cluster mean which is the mean of the feature factors?

So, I get k number of mean vectors, once I get this the k number of mean vectors, then you try to reassign the set of points into various clusters based on minimum distance. It may so happen that initially because your clustering was totally arbitrary, so a feature vector which has been put into say cluster one. But after computation of the cluster centers you may find that for this feature vector it is distance to cluster center of cluster 5 is less than it is distance center from cluster 1, and it is distance from center cluster 5 is minimum among all other distances.

So, you reassign this feature vector into cluster number 5. So, likewise you try to reassign all the feature vectors using minimum distance, based on minimum distance from all the cluster centers. So, after reassignment you recompute the cluster centers and then reassign the

feature vectors again. So, your recomputation of the cluster centers and reassignment of the feature vectors goes on iteratively until and unless you come to a stage, when the clusters have not changed. So, that is the condition of vectors that in a pass no point has been or no feature vector has been reassigned to any other cluster, ok? So, that is a condition of convergence. So this is an iterative problem. So, this is an I can have a formal way original way of this iterative problem, where I can define that or I can define a criteria function where this criteria function is based on sum of squared error. So, if I say let us take i^{th} cluster.

(Refer Slide Time: 32:26)

Criterion Function based Clustering.

$i \rightarrow \text{cluster.}$

$$J_i = \sum_{\forall x \in D_i} \|x - m_i\|^2$$

$$m_i = \frac{1}{n_i} \sum_{\forall x \in D_i} x$$

$$J_e = \sum_{i=1}^k J_i$$

So, criteria function based like clustering, so again in this case I assume that I will form k number of clusters. So, initially all the feature vectors will be arbitrary partitioned into k number of clusters. Now, if I take i^{th} cluster, in i^{th} cluster the error in the i^{th} cluster J_i will be

computed as $\sum_{\forall x \in D_i} \|x - m_i\|^2$, sum of this for all X which belongs to partition D_i . However, m_i

is nothing but the mean of this feature vectors which are put into this cluster D_i . So,

$m_i = \frac{1}{n_i} \sum_{\forall x \in D_i} x$. n_i is the number of feature vectors put into say D_i and summation of X for

all X belonging to D_i . So, this I compute for every cluster i , where i is from 1 to k . As I am saying that I consider k number of clusters and the total error is given by J_e is equal to sum of J_i , where i varies from 1 to k .

So, you find that this term tells you within clusters some of square area and this tells you what is the overall error? So, this clustering technique tries to find out the clusters for which

this total error will be minimum. So, how I can do it iteratively? Suppose, after that initial partitioning I take one feature vector from one cluster and try to push into another cluster. And then try to find out that if I push this feature vector from one cluster to another cluster, whether the total error can be reduced or not, ok?

(Refer Slide Time: 35:15)

$$\begin{aligned}
 & \hat{x} \text{ move from } D_i \text{ to } D_j \\
 & m_i \quad m_j \\
 & m_j^* = \frac{1}{n_j+1} [n_j m_j + \hat{x}] \\
 & = \frac{1}{n_j+1} [m_j(n_j+1) + \hat{x} - m_j] \\
 & = m_j + \frac{\hat{x} - m_j}{n_j+1}
 \end{aligned}$$

So, I take a feature vector \hat{X} and try to move it from subset D_i to subset D_j , you take this feature vector \hat{X} from the i^{th} cluster, try to push it into j^{th} cluster. So, when I try to do this, then the i^{th} cluster initially had mean of m_i . Similarly, j^{th} cluster had mean of m_j . So, as I am pushing \hat{X} from i^{th} clusters to j^{th} cluster, this m_j mean of j^{th} cluster will be changed. Let us put it as m_j^* , that is the new mean of the j^{th} cluster which will be nothing but initially before putting this \hat{X} into this j^{th} cluster, the number of points in the j^{th} cluster was the n_j as put this new point into the j^{th} cluster, the number of points in this j^{th} cluster will be $n_j + 1$, ok?

So, this will simply be $m_j^* = \frac{1}{n_j+1} [n_j m_j + \hat{X}]$, so this is the mean, new mean of the j^{th} cluster D_j . And if I simplify this, you find that it will be $m_j^* = \frac{1}{n_j+1} [m_j(n_j+1) + \hat{X} - m_j]$, I can rewrite this

expression as $m_j^* = m_j + \frac{1}{n_j+1} [\hat{X} - m_j]$. So, as this new feature vector \hat{X} has been pushed from the i^{th} clusters to j^{th} cluster, the mean of the j^{th} cluster which before this \hat{X} was pushed into this was m_j . After this the \hat{X} is pushed into the j^{th} cluster, this m_j has been changed to m_j^* . However, you find that there is an increment and in m_j which is given by $\frac{1}{n_j+1} [\hat{X} - m_j]$, ok? So, in the same manner the total error I can also compute, what is the total error, change in total error of the j^{th} cluster. So, the change in total error of the j^{th} cluster will be something like this.

(Refer Slide Time: 38:28)

The derivation shows the calculation of the change in total error $J_j^* - J_j$. It starts with the definition of J_j as the sum of squared distances of all points in cluster D_j from the mean m_j . Then it defines J_j^* as the sum of squared distances of all points in D_j from the new mean m_j^* , plus the squared distance of the new point \hat{X} from m_j^* . This is then simplified into two terms: the sum of squared distances of points in D_j from their original mean m_j , plus the sum of squared distances of points in D_j from the new mean m_j^* . Finally, it is shown that this is equal to the original error J_j plus the term $\frac{n_j}{n_j+1} \|\hat{X} - m_j\|^2$.

$$\begin{aligned}
 J_j &= \sum_{\forall x \in D_j} \|x - m_j\|^2 \\
 J_j^* &= \sum_{\forall x \in D_j} \|x - m_j^*\|^2 + \|\hat{X} - m_j^*\|^2 \\
 &= \sum_{\forall x \in D_j} \left\| x - m_j - \frac{\hat{X} - m_j}{n_j+1} \right\|^2 + \left\| \frac{n_j}{n_j+1} (\hat{X} - m_j) \right\|^2 \\
 &= J_j + \frac{n_j}{n_j+1} \|\hat{X} - m_j\|^2
 \end{aligned}$$

So, before pushing this \hat{X} the error in the j^{th} cluster was J_j which is nothing but $\sum_{\forall X \in D_j} \|X - m_j\|^2$. So, this was the total error in the j^{th} cluster before putting this \hat{X} . After I put this \hat{X} , then the total error in the j^{th} cluster is changed to J_j^* and which is nothing but $\sum_{\forall X \in D_j} \|X - m_j^*\|^2$, because now the new mean m_j^* , it is not m_j anymore, square where for all X belonging to cluster D_j , plus as this new feature vector \hat{X} has been included in the j^{th} cluster.

So, it will be $\|\hat{X} - m_j\|^2$, this is the total error $J_j^* = \sum_{x \in D_j} \|x - m_j^*\|^2 + \|\hat{X} - m_j\|^2$ in the j^{th} cluster

and again I can simplify this.

So, this will be and this can be simplified $J_j^* = \sum_{x \in D_j} \left\| x - m_j - \frac{\hat{X} - m_j}{n_j + 1} \right\|^2 + \frac{n_j}{n_j + 1} \|\hat{X} - m_j\|^2$ and

the total error squared error has been increased from J_j to $\frac{n_j}{n_j + 1} \|\hat{X} - m_j\|^2$. So, as there has

been change in the mean and error some of squared error in the j^{th} cluster. Similarly, there will also be change in the mean and sum of the squared error in the i^{th} cluster, ok?

(Refer Slide Time: 41:44)

The notes show the following steps:

- Condition: $n_i \neq 1$
- Modified cluster center: $m_i^* = m_i - \frac{\hat{x} - m_i}{n_i - 1}$
- Modified total error: $J_i^* = J_i - \frac{n_i}{n_i - 1} \|\hat{x} - m_i\|^2$
- Inequality: $\frac{n_i}{n_i - 1} \|\hat{x} - m_i\|^2 > \frac{n_j}{n_j + 1} \|\hat{x} - m_j\|^2$

So, if I consider the i^{th} cluster assuming that $n_i \neq 1$. Why I am considering this $n_i \neq 1$ because if $n_i = 1$, that means the i^{th} cluster or set D_i had a single point. If it has a single point, if I want to push that into another cluster then the number of clusters will be reduced, ok?

So, if any clusters contain only one feature vector, that cluster should not collapsed. So, that is the reason that this condition $n_i \neq 1$. So, if $n_i > 1$, then only I can take a feature vector from the i^{th} cluster, trying to push it into another cluster. So, consider, having this particular condition that $n_i \neq 1$, the cluster center for the i^{th} cluster will be modified to m_i^* will be equal

to $m_i - \frac{\hat{x} - m_i}{n_i - 1}$.

By similar, such calculation you can find out this and similarly the error within the i^{th} cluster will be changed to J_i^* . So, it is by similar calculation that we have done earlier. I can find out that what is the change in the mean of the i^{th} cluster and what is the change in the sum of squared error in the i^{th} cluster.

So, now if you compared these two, you find that for the j^{th} cluster, so this is my J_j^* . So, for the j^{th} cluster where I have pushed this feature vector \hat{X} , there the total error has been increased by amount $\frac{n_j}{n_j+1} \left\| (\hat{X} - m_j) \right\|^2$. This is the amount of increase in the error in the j^{th} cluster.

Similarly, in the i^{th} cluster, the total error has been reduced, it has been decreased by $\frac{n_j}{n_j-1} \left\| (\hat{X} - m_i) \right\|^2$. So, here I have a net increase over here, there is an increase by this amount and over here there is a reduction by this amount. So, this transfer of \hat{X} from the i^{th} cluster to j^{th} cluster will be profitable, in the sense that it will reduce the total error because total error is nothing but sum of these two errors.

So, it will reduce the total error, if the reduction in the error in the i^{th} cluster is greater than the increase of error the j^{th} cluster. So, where ever I have error reduction, the amount of reduction has to be more than the amount of increase that is happening in the other cluster. Then only when I take the sum then I will have a net reduction in the total error. So, I will take a feature vector from one of the cluster, try to push it into another cluster and this transfer of the feature vector from one cluster to another cluster will be effective, will be done, only when the decrease in error in one cluster is more than the increase in error in the other cluster, ok?

So, my condition for transfer will be that $\frac{n_j}{n_j-1} \left\| (\hat{X} - m_i) \right\|^2 > \frac{n_j}{n_j+1} \left\| (\hat{X} - m_j) \right\|^2$. So, only

when this condition is true then only I can push this \hat{X} from this i^{th} cluster to j^{th} cluster. Now, again you find that I am taking x hat from the i^{th} cluster, that means this amount is fixed, but i^{th} cluster is one of k number clusters. So, I have $k - 1$ number of other clusters and this j can take on any of this $k - 1$. This j^{th} cluster can be any of this $k - 1$ number of clusters, right? So,

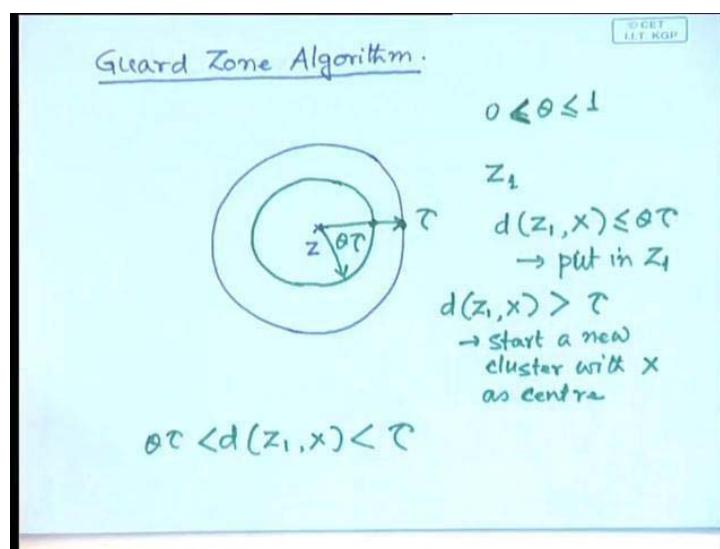
what I have to do is I have to compute this value for all those j^{th} clusters which is not i^{th} cluster, ok?

So, for all values of $j \neq i$, I have to compute this value and because this is the amount of increase, naturally I should select that particular cluster where the increase is minimum. So if I have two such clusters, cluster number 2 and cluster number 3, I find that the increase in the error in cluster number 2 is more than the increase in error in cluster number 3.

So, obviously I should push this \hat{X} into cluster number 2, not in cluster number 3 because in that case the total amount of reduction that will be, that I will get is not minimum. So, this ensures that by pushing this \hat{X} from i^{th} clusters to j^{th} cluster you have a net reduction in the total error. And to maximize that reduction I should push x hat, push \hat{X} into cluster which gives minimum increase in the within class, within clusters error, ok?

So, out of all those $A - 1$ number of clusters for whichever cluster the increase in the sum of squared error is minimum, this \hat{X} should be pushed into that. So, for the first condition without this I should not try to push \hat{X} from one cluster to another cluster. And the next condition is I should try to find out the cluster where if I put \hat{X} , the increase in the sum of squared error will be minimum, ok? So, this is another iterative algorithm following which we can also find out the cluster for a given set of feature vectors. There is another algorithm which is called guard zone algorithm.

(Refer Slide Time: 49:41)



So, guard zone algorithm is something like this. Suppose, somehow I get a cluster center z said and then around this cluster center z , I define a zone, ok? So, if a point falls outside this zone, then that feature vector will not be put into same cluster as that cluster center z and within this zone I have a sub zone something like this, ok? So, suppose this zone is defined by this parameter τ and this is a fraction of this parameter. So, let us put it as θ times τ , ok? So, the algorithm works like this, you take one of the feature vectors, assume that to be the cluster center. Take the next feature vector, if the next feature vector falls or the distance between the next feature vector and this cluster center is less than $\theta\tau$, you put that feature vector into the same cluster as this cluster center. If it is between $\theta\tau$ and τ do not take any immediate action or immediate decision about that feature vector, and if the feature vector is beyond τ then that feature vector should not be clustered into this z , ok?

So, normally this τ it is decided by the designer. Similarly, what fraction of τ I should consider for putting a feature vector into the same clusters, that is also decided by the designer. So, θ is obviously between 0 and 1. So, if it is equal to 1 then whenever the feature vector falls within this region τ , it will be put into the same cluster. I should not put it as 0, but it is greater than 0, 0 not inclusive, but 1 maybe inclusive. So, the algorithm is like this, if you take the first feature vector consider that as your first cluster center, z_1 . Take the next feature vector, find out the distance between the feature vector and this cluster center z_1 .

So, x is the next feature vector, distance between z_1 and x . If this distance is less than or equal to $\theta\tau$, if this is less than or equal to θ times τ put it into the center stage if distance, this distance is greater than τ . So, in this case put in the same clusters z_1 , if this distance is greater than τ , then you can start a new cluster with x as center. So, that says that x is somewhere over here or some over here that is far away from z_1 . So, you start a new cluster with x as the new cluster center. However, as if $d(z_1, x)$, if this is between $\theta\tau$ and τ that means it is within this region. Do not take any immediate decision about this, you keep the case pending.

So, like this when all the feature vectors are considered, some of the feature vectors will be put into some of the clusters. Some of the feature vectors this is what is called guard zone, will be in the guard zones of other clusters, which are already formed. Then what I can do is you can find out that, what is the number of points which are there in the guard zone of a particular cluster. If the number of points within the guard zone of a cluster is not significant is quite small. Then you put them into the same cluster, but if the number of points is quite significant then I can do this interoperation once more with a modified value of τ , maybe with a modified value τ for to try to re-cluster all those points which are there in the cluster.

So, this depends upon how many points in the guard zone. If the number of points in the guard zone is quite small, then try to put them into same cluster in whose guard zones the points are existing. But if the number of points in the guard zone are quite high, then it is quite logical that I can try to form more number of clusters using those points as their degree of belongingness within the same clusters in guard zones. They are belonging, they are existing is not that strong.

So, you can try to form other clusters using this one.

So, again you find that this an iterative algorithm and as it is iterative algorithm, the final clustering output as in case of any other iterative algorithm will depend upon what is the order in which the points are presented, ok? Unlike in case of graph graphical algorithm, where we said that the when I form a graph it is a global structure. So, in case of a graphical algorithm, the final output does not depend upon the order in which the points are presented. But in case of any of the iterative algorithms, the final output may vary depending on the order in which the feature vectors are presented to that algorithm. So, we will stop here today.

Next day we will talk about some other topic.

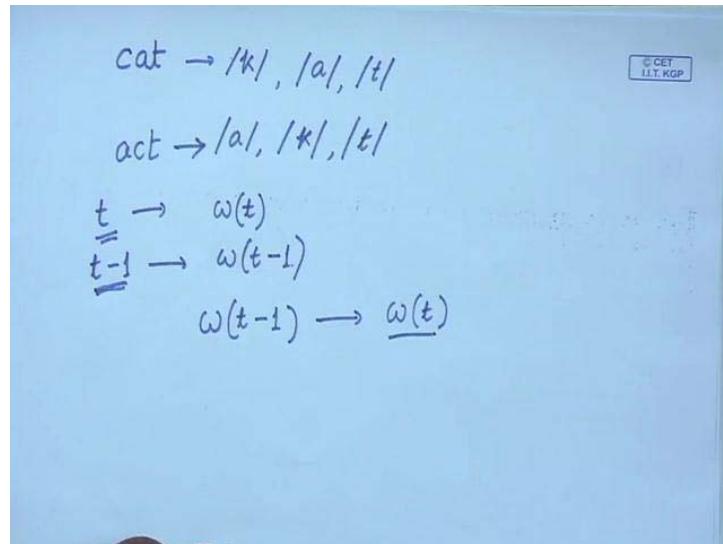
Thank you.

Pattern Recognition and Image Understanding
Prof. B. K. Biswas
Department of Electrical and Electronic Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 37
Temporal Pattern Recognition

Hello, so till now what we have discussed is the different classification or recognition techniques of patterns, as well as different clustering techniques of patterns. So, the patterns that we have discussed so far, they are actually static patterns, that is the patterns, which do not have any temporality in it. So, if I have a set of feature vectors which are static over time, then I can either classify those feature vectors or I can cluster those feature vectors. Today, what we are going to discuss is the classification of Temporal Patterns or patterns, which actually unfolds in time. So, what do you mean by the patterns, which have temporality or the patterns, which unfolds in time.

(Refer Slide Time: 01:38)



Let us take for example as speech signal in, suppose I want to recognize a word or spoken word say cat. So, if I simply write cat, c a t, so if I want to recognize this word cat as it is spoken, you find that this word cat it can be broken into three different phonemes. So, the first phoneme say /k/ second phoneme is /a/ and the third phoneme is /t/. So, I can represent

this as the first phoneme which is /k/, then /a/ phoneme and then a phoneme /t/. So, if I utter these phonemes as given in this /k/, /a/, /t/ I pronounce the word cat or whereas if I simply altered the temporal sequence of these three defined phonemes suppose I first, I utter /a/ then I utters the phoneme /k/, then I utters /t/ its simply becomes act, so the word that I pronounce in this case is act. So, though the phonemes are same, but they are uttered in a different sequence in time. So, as the same phonemes are uttered in different sequence in time you find the word that is being pronounced, the word itself changes. So, this kind of pattern, which is a temporal pattern. So, when we talk about the problem of speech recognition, when we talk about the problem of speaker recognition any signal, which varies with time say for example, some system identification task. There we get a signal where the signal varies with kind it is a temporal signal.

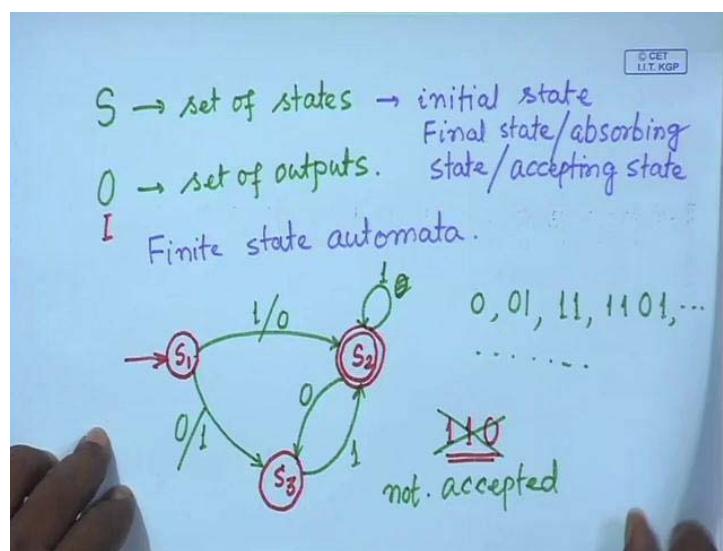
So, these are the examples of temporal patterns or the patterned which actually unfold in time. Similarly, coming to our vision context of vision applications we can have the different applications of say gesture recognition. So, we want to identify whether a person is running or the person is walking, the person is scrolling. So, that these different kinds of act that is running, walking, scrolling. So, we want to classify or we want to recognize these different kind of tasks or possibly all of yours over sign language.

In case of sign language, you find that it is also the sequence of hand gestures, the different hand positions, which actually, convey the meaning of the information. This convey the information to the others, so that particular sign language, this is also an example of temporal pattern. So, today what we are going to discuss is, how to recognize some temporal patterns or how to classify such temporal patterns? So, hope the kind of tools that we will use for such temporal patterned recognition is something, which is similar to sequential machine or finite state machine, which all of you might have done.

I hope all of you have done during your digital circuits course, so what is a sequential machine or what is finite state machine in case of sequential machine or finite state machine? We actually define a finite set of steps and a machine can be in any of the states at any given time instant. And that same time we also have a set of outputs so in any state when it makes the transition from one state to other it outputs a symbol. So, we have a set of states and a set of output symbols, so we can say that at time instant t the machine can be at state say ω_t . And at time instant t the previous time instant the machine can be at state, so at time instant $t - 1$ the machine can be at state ω_{t-1} .

So, in case of a sequential machine or a finite state machine the state of the machine, at time step t the time step t is directly influenced by the state of the machine, at time step t minus 1. So, there is a transition from state $\omega(t-1)$ to the state $\omega(t)$. So, the state of the machine at time step $\omega(t)$ is influenced by what is the state of the machine, at the time step $t-1$. So, let us just try to briefly review what we had in case of a sequential machine or a finite state machine. So, in case of a sequential machine it has a finite set of states say S , which is actually set of states and O , which is the set of outputs.

(Refer Slide Time: 07:55)



So, when we have this set of finite set of states one of the state is called an initial state and one of the states is a final state or absorbing state. So, we can have out of these sets one state is an initial state and one state is final state or it may also be called absorbing state or accepting state.

And the set of outputs is actually the set of symbols, which the machine outputs or visible symbols, which can be measured or it can be observed, when the machine makes the transition from one state one state to another. If you remember, if these set of outputs symbols is limited to only two symbols that is either 0 or 1. Then this finite state machine is called finite state automata. So, a finite state of automata is nothing but a sequential machine or a finite state machine, which outputs only one of the two symbols one of them is 0 and the other one is 1. So, let us just take one such finite state machine, suppose I have a finite state machine, which have got three states one is S_1 , one state is S_2 and the other state is S_3 .

And suppose, out of these three is two is my final state or observing state and S_1 is the initial state. Now, along with the set of states is and the set of output O there is another set, which is I that is the set of inputs. So, the machine makes the transition from one state to another state with some input, which is actually the trigger. So, in this sequential machine having three different states I have assumed the state S_1 is my initial state, S_2 is my final state and S_3 is any other state. So, the machine can make a transition from state one to state two and give him output. Let us say 1/0, it can make a transition to S_1 to S_3 and give him output, which is say 1 it can make a transition from S_3 to S_2 .

And it can give an output 1 it can also make a transition from S_2 to S_3 . For this transition it can output symbol, which is 0, it is also possible that a machine will have a self-transition that there will be self-loop. In this case I can have something like this like S_2 , I can have transition to S_2 itself and for that it will output a symbol let us say 1. So, given such a sequential machine and the outputs, which are emitted by this machine whenever it makes a transition from one state to another state. And over here, as I have assumed that S_1 is my initial state and S_2 is final state.

So, you find the kind of the outputs, which are actually accepted by this machine or the kind of or the sequence of outputs, which are generated by this machine. That can be 0, 1 it can be 1, 1, 1 it can be 0 then 1, then 0, then 1, then 1 and all this sort of things. So, the kind of sequences, which is accepted by this machine, will be say 0. As with output 0 the machine is making a transition from S_1 to S_2 and S_2 is the accepting state it can be 0 followed by 1. Because, with 0 we are making a transition from S_1 to S_2 , then S_2 it can make transition with in S_2 to S_2 with an output equal to 1.

So, that is still in accepting state, so I can also accept a stream 0, 1 I can accept a stream 1 when the machine makes a transition from S_2 to S_3 , then it makes a transition from S_3 to S_2 with an output 1. So, 1, 1 there is a stream, which will be generated by this sequential machine. It will also be accepted by the sequential machine, I can also have 1 1, 0 1, this is also an accepted sequence. So, I can have a number of sequences of such output symbols, which are accepted by this machine, but you find that if I have a symbol say 1, 1, 0. So, from the initial state the machine comes to state S_3 with an output 1 from S_3 it goes to S_2 with output 1 from S_2 it comes to S_3 with output 0, but S_3 is not an accepting state.

So, this symbol or this sequence of symbol does not terminate the machine in an accepting state. So, a result this symbol will not be accepted by this machine or this sequence of symbols will not be accepted by this sequential machine. So, because for acceptance of a

sequence of symbols, by a sequential machine or a finite state machine with the sequence, the sequence must terminate in one, in one accepting state. So, this sequence 1, 1, 0, this is not accepted by this sequential machine, this is not accepted, so this is what I have a sequential machine. In case of a sequential machine the probability of transition from one state to another it depends upon an input.

Say for example, here if I give input say 1 it moves to the state S_2 with an output 1, but if my input is 0 it moves to state S_3 with an output 1. So, with 1 as input the machine moves from state one to S_2 with an output 0. If the input is 0 in state S_1 the machine moves from S_2 to S_3 with an output 1. So, this is what a sequential machine or a finite state of automata, if the set of output symbols is limited to 0 and 1. So, a tool which is similar to this sequential machine or very similar to this finite state machine, which is used for temporal pattern recognition, is called a Hidden Markov model.

(Refer Slide Time: 17:19)

Hidden Markov Model
(HMM)

$$t \rightarrow \omega(t)$$

$$t \rightarrow t+1 \Rightarrow \omega(t) \rightarrow \omega(t+1)$$

$$\underline{\omega^T = \{ \omega(1), \omega(2), \dots, \omega(T) \}}$$

$$\underline{\omega_1, \omega_2, \omega_3}$$

$$\underline{\omega^4 = \{ \omega_1, \omega_3, \omega_1, \omega_2 \}}$$

So, what we will discuss today is, what is called a Hidden Markov model? or in short it is written as HMM. So, as we said that this Hidden Markov model, which is used for temporal pattern recognition is very similar to the sequential machine or the finite state machine. That is this hidden Markov model will have a set of states and the hidden Markov model, the machine can be in any of the states at a time instant t . So, suppose this at time instant t the machine is in state say $\omega(t)$. And what I can have is as the machine moves a transition from one state to another that is from time instant t to time instant $t + 1$.

So, from t to $t + 1$ the machine makes a transition or HMM makes a transition from state $\omega(t)$ to state $\omega(t+1)$. So at time step t , HMM was in state $\omega(t)$ at time step $t+1$, at the next time step the machine was in state $\omega(t+1)$. So, that is a transition from $\omega(t)$ to $\omega(t+1)$. Because, the machine makes transitions from one state to another from different time steps. So, the machine generates a sequence of states and a sequence of states are sequence of length, say capital T is usually represented as ω^T , which is nothing but a sequence of states $\omega(1), \omega(2), \omega(3)$ continues like this up to $\omega(T)$.

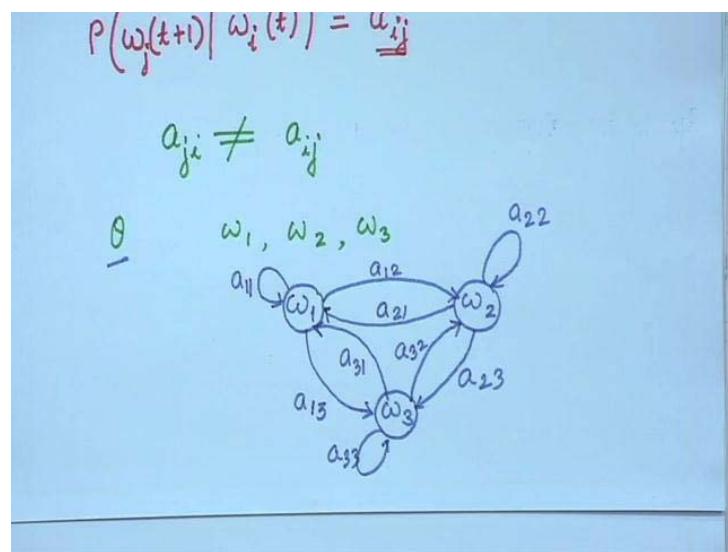
So, it simply says that at $t = 1$, the machine was in state $\omega(1)$, so this $\omega(1)$ may be any of the permissible states of the machine. Similarly, at time step 2 the machine makes a transition from $\omega(1)$ to $\omega(2)$. So, this $\omega(2)$ again may be any of the permissible states of the hidden Markov model. So, this way over total duration of capital T the machine moves from one state to another. And the sequence of steps or the sequence of states through which the machine moves this sequence is represented by this ω^T . So, it is a sequence of capital T number of states through which the machine moves and this sequence is nothing but $\omega(1), \omega(2), \omega(3)$ up to $\omega(T)$.

So, this is the state in which the machine was at time step 1, this is the state in which the machine was at time step two and so on. So, this is I have total capital T number of states, so for example if I have a machine with say states $\omega_1, \omega_2, \omega_3$, say these are the states of the machine. Now, you find the difference between this $\omega(1)$ where 1 is put with in parenthesis and this to ω_1 , this is a particular state of the machine ω_1 is one of the states. As the machine is having 3 states ω_1, ω_2 , and ω_3 , whereas I when I write in this form it is one of these states in

Suppose, I can have a sequence of states of length say four this may be say $\omega_1, \omega_3, \omega_1, \omega_2$, so this may be my sequence of four states. So, a machine moves from one state to another in this particular sequence. So, at $t = 1$ or $\omega(1)$ was these ω_1 , at $t = 2$, the $\omega(2)$ state of the machine might have been ω_3 , at $t = 3$, in the third step the machine might have been in state of ω_1 , in the fourth step the machine is in state ω_2 . So, the four states through which the hidden Markov model makes a transition, so as I have four different states. So, this is the sequence of length four, which is usually represented as ω^4 .

Here, again you find that it is not necessary that every state, a single state has to appear only once in a sequence. A state can appear more than once in different orders. So, here you find that ω_1 and ω_1 , these two states appear twice. So, it means that from ω_1 the machine may have made a transition to ω_3 from ω_3 , again it has made a transition to ω_1 , from ω_1 again it has made a transition to ω_2 . Now, this production of state sequence in case of a hidden Markov model is described by a transition probability. So, the transition probability is something like this, the probability P that the machine makes a transition to states, $\omega_i(t)$ to state $\omega_j(t+1)$ given the machine was in state ω_i at time step t .

(Refer Slide Time: 24:14)



So, what is the probability that the hidden Markov model will make a transition from ω_i , which is the i^{th} state to state ω_j ? If the machine was in state ω_i at time step t and the machine moves to state ω_j at time step $t+1$, that is at the next time state. So, the probability of this transition that the machine makes a transition from state ω_i at time state t to state ω_j at time state $t+1$, this is what is given by the transition probability a_{ij} , compare with this our sequential machine. In case of sequential machine, we said that depending upon input the machine moves, makes a transition from one state to another. So, if my input is 1 the transition probability from S_1 to S_2 is equal to 1 where as if my input is 0 the transition probability from S_1 to S_3 is 1.

So, this probability it will make a transition from S_1 to S_3 , but if my input is 0 the machine cannot make a transition from S_1 to S_2 . Similarly, if the input is one this sequential machine

cannot make a transition from S_1 to S_3 . So, that is the kind of restriction we have in case of a sequential machine. Where as in case of hidden Markov model from state ω_2 the machine make to transition to state ω_1 , it may make a transition to state ω_3 , it may make a transition to ω_5 .

So, all these transition are possible and there is a transition probability that it makes a transition ω_1 to ω_3 . There is a probability that it makes a transition from ω_1 to ω_5 , there is a probability that it makes a transition from ω_1 to ω_{10} and all this. So, from any state at any instant of time it can make a transition to any of the other states or to the state itself with a finite probability of transition. So, that is what we are saying over here that, if the machine was in state ω_i at time instant t. Then the probability that it will make a transition to state ω_j at time instant t plus 1 is given by a_{ij} . So, this is the probability of transition from state ω_i to ω_j . It is also possible, that the machine will make a transition from ω_j to ω_i .

So, at the probability of transition is actually given by a_{ji} , this is the probability transition from ω_j to state ω_i . The probability transition from state i to j is given by a_{ij} . And in general, this a_{ij} and a_{ji} there are not equal, that is the transition probability from state ω_i to ω_j may not be same as the transition probability, from ω_j to state ω_i . So, these defined transition probabilities will actually define the different hidden Markov models. So, when we design a hidden Markov model or when we go for training of the hidden Markov model. The major task of the hidden Markov model learning these transition probabilities a_{ij} , which we will see later on.

Now, suppose you are given a particular hidden Markov model, so hidden Markov model which can recognize a particular temporal pattern. So, we use a represent such a model by a symbolic θ . So, this θ represents a specific hidden Markov model. And we have say, for example a machine this particular hidden Markov model θ has, say three different state one of the state ω_1 , one of the state is ω_2 and the other state is ω_3 .

Now, when we have such a kind of hidden Markov model having a number of different states. One of the state is an absorbing state or an accepting state the concept is, once the model moves to that particular state the model cannot come out of that state, however, from

any of the other states, the model can move, make a transition to that absorbing state that once in absorbing state, the model cannot come out of it.

We will come to the utility of such a specific state, which is an absorbing state or an accepting state or a final state later on. So, suppose this machine, this three different states given by $\omega_1, \omega_2, \omega_3$. And let us represent this as say I have state ω_1 I have state of ω_2 and I have state of ω_3 . So, this machine can make a transition from ω_1 to ω_2 with a probability of transition, which is given by a_{12} it can make a transition from ω_2, ω_3 , sorry ω_2 to ω_1 with a probability of transition given by a_{21} .

It can make a transition from ω_1 to ω_3 here I will have a probability of which is a_{13} . It can make a transition from ω_3 to ω_1 with probability of transition a_{31} . It can make a transition from ω_3 to ω_2 with probability of transition a_{32} . It is also possible that it will have a transition to the same state. So, over here ω_2 to ω_2 , here the probability transition will be a_{22} . Similarly, here the transition from ω_1 to ω_1 I can have two different states consequently I am in same state appearing successively, I can also have similar such situation.

So, this transition from ω_1 to ω_1 , which will be represented by a_{11} . Similarly, transition from ω_3 to ω_3 itself, which is over here, which will be written as a_{33} . So, suppose I have hidden Markov model specified by θ , and this hidden Markov model let us say as four different states $\omega_1, \omega_2, \omega_3$, and ω_4 .

(Refer Slide Time: 32:39)

$$\theta = \omega_1, \omega_2, \omega_3, \omega_4. \quad a_{ij}$$

$$\omega^6 = \{ \omega_1, \omega_4, \omega_2, \omega_3, \omega_1, \omega_4 \}$$

$$P(\omega^6 | \theta) = a_{14} a_{42} a_{22} a_{21} a_{14}$$

later $\rightarrow /t/, /a/, /t/, /a/, /r/$

alter $\rightarrow /a/, /t/, /t/, /a/, /r/$

$$P(\text{later} | \theta) = a_{12} \cdot a_{23} \cdot a_{32} \cdot a_{24}$$

$$P(\text{alter} | \theta) = a_{21} \cdot a_{13} \cdot a_{32} \cdot a_{24}$$

So, I have four different states, so I have a model θ that is the hidden Markov model having four states ω_1 , ω_2 , ω_3 and ω_4 . And suppose, we have been given a sequence of states, so sequence of six states ω^6 , where this sequence is say ω_1 , ω_4 , ω_2 . I can have ω_2 again I have again ω_1 , ω_4 . So, this is the sequence of state to reach the machine makes a transition, now given this I can find out the probability that machine, θ has made this sequence of transition. How I can do it? So, what I have to find out is, I have to find out what is the probability $P(\omega^6/\theta)$.

So, when I have this hidden Markov model θ , that means I have the sequence of states. I also have the complete table or probability of state transition table, that is I have all the a_{ij} is for all values of i and j . That is what specifies a hidden Markov model. So, suppose this θ is specified by these are the states and I also have a_{ij} , which is to be learnt, but once the model is specified, I have the complete set of a_{ij} for all values of i and j . So, given such a machine θ and given this sequence of states, that is machine makes a transition from ω_1 to ω_4 , ω_4 to ω_2 , ω_4 to ω_2 again ω_2 to ω_1 , and then ω_1 to ω_4 .

So, what is the probability that this model θ has made such a sequence, has transited through such a sequence of states. So, effectively what I want to find out is what is $P(\omega^6/\theta)$. You

find that this $P(\omega^6/\theta)$ is
nothing but the probability of transition from ω_1 to ω_4 , which is nothing but a_{14} , then probability of transition from ω_4 to ω_2 , so this multiplied by a_{42} , which is the probability of transition from state ω_4 to ω_2 . Then the probability of transition from ω_4 to ω_2 itself. So, I have a_{22} , which is the probability of transition from state ω_4 to ω_2 . Then from ω_2 to ω_1 , which is nothing but a_{21} , and then the probability of transition from ω_1 to ω_4 which is a_{14} .

So, the probability that this model θ has made a transition through a sequence of states is nothing but the probability of transitions, between successive states. So, it is the product of the probability of transition between successive states. So, the machine this θ will have these transitions the probability of that. That is $P(\omega^6/\theta) = a_{14}a_{42}a_{22}a_{21}a_{14}$.

So, this is the probability that, this machine θ has made a transition through this sequence of six states as specified by the ω^6 . So, that that is fine I mean I have a model the model has a number of states. The machine makes a transition through a sequence of states over time and what is the probability that the machine has generated? Or has transited through a given sequence of states? I can compute that probability, now what are these states actually.

Let us say that I have again in with an example to my speech recognition. I have two different words one is the later other one is alter. So, you find that they use the same phonemes one is later other one is alter. If I break them into phonemes this consists of a sequence of phonemes, /l/, /a/, /t/, /a/ again and /r/, where as in case of alters this is /a/, this is /l/, this is /t/, this is /a/, again this is /r/.

So, you find that it is the same set of phonemes, but occurring in different order in time sequence. That is what makes the difference between later and alter. So, I can say that in my hidden Markov model every state corresponds to one such phoneme, every state will correspond to one such phoneme. So, as here I have one two three and four, four different phonemes, because this phoneme and this phoneme is same. So, I have four different phonemes, so I can have four different states to model this particular spoken word. Similarly, the same different states, but occurring in the different sequence will generate this spoken word alter.

So, this may be the different states and if I say the first phoneme will /l/ corresponds to states ω_1 , /a/ corresponds to state ω_2 , /t/ corresponds to state ω_3 and /r/ corresponds to state ω_4 . In that case the probability $P(\text{later} / \theta)$, that is my model it will simply be here you are making a transition from /l/ to /a/, so ω_1 to ω_2 . So, it will be $a_{12}a_{23}a_{32}a_{24}$. So, this is the probability that hidden Markov model will generate the spoken word later or it can recognize the spoken word later is this.

Similarly, the probability that it has generated alter given θ this will be simply $P(\text{alter} / \theta) = a_{21}a_{13}a_{32}a_{24}$. So, because of this the probability that this θ generates later and the θ generates alter they will be different for a given hidden Markov model θ .

So, if I actually train a hidden Markov model to recognize the word later, the spoken word later. Then when I have the same word later, which is spoken with these different phonemes. Then the probability that the θ will return or the word later will be more, then the same θ

which will return the probability for the spoken word alter. So, once you train a hidden Markov model with a particular sequence, the model actually learnt the probabilities of transition from one states to another. For another word, another spoken word this probability of transition may not match though the states might be same, so that means we will generate a lower probability of generation or lower probability of performance.

So, this is how given a model and given a sequence of states I can find out what is the probability that the machine has transited or the hidden Markov model has transited, through that given sequence of state. Now, as I said that in this hidden Markov model, there is a specific state, which is an accepting state or an absorbing state and this specific state is usually represented by the symbol ω_0 .

(Refer Slide Time: 44:43)

ω_0

$a_{00} = 1$

$a_{0i} = 0 \text{ for all } i$

$V \rightarrow v(t) \quad \omega(t)$

$V^T = \{v(1) v(2) v(3) \dots v(T)\}$

And as I said that once this hidden Markov model enters the state ω_0 it cannot come out of that state. However, it can make infinite number of transition within that state. So, as a result I will always have a_{00} that is the probability of transition from state ω_0 to itself will always be equal to 1 whereas, because I cannot have a transition from ω_0 to any other state.

So, I will always have $a_{0i} = 0$, because from ω_0 I cannot make a transition to any other state ω_i . So, a_{0i} will always be 0 for all i , so this is what is very important. I can have, I will have $a_{00} = 1$, because from the final state, it can make infinite number of transitions within the same state, but from the final state it cannot make transition to any other state.

So, as a result a_{oi} for all states i will be equal to 0 whereas a_{00} will equal to 1. Now, you find that we have, so far discussed this hidden Markov model with respect to states. Now, these states are the ones, which are not really visible or it cannot be absorbed by the preserver what is visible or what is observable to the preserver is certain set of symbols, which can be observed or which can be measured. Right? Whereas, these states are not really visible I am in the same thing applies to our sequential Machine.

These states v_1, v_2, v_3 . They are not really observable, I cannot observe them, but what I can observe is the output, which is emitted by the machine. So, I can observe this output 1, I can observe this output 0.

So, this is the outputs which are really observable the state is not really observable. So, in the same manner, in case of this hidden Markov model the states that we have talked about. These states are not really observable, but what is observable? Or what can be measured based on which we can take a decision whether a given word or spoken word or a sequence of actions belongs to a particular class or belongs to a particular category. This classification has to be done based on observable symbols I cannot observe the state through, which the machine will transit. So, along with the states I also have to have a set of observable symbols.

So this set of observable symbols is actually, this given by a set say V this is the set of observable. Accordingly, I can have a visible state, which is given by $v(t)$. So, for hidden Markov model I have two types of states one type set of states, which cannot be observed which are not visible or hidden, those states are hidden, which are represented by Ω and a set of states, which have observed it, which can be observed, which can be measured. That is the visible state, so at a time step t the machine can be state $\omega(t)$, which is hidden state and it can emit an observable state, which is given by $v(t)$. As you have done in case of the states or the hidden states, that the machine can transit through a sequence of hidden states.

Given a model I can find out what is the probability that the model has transited through such sequence of hidden states, in a similar manner for the visible states. I can have a sequence of visible states, so a sequence of visible states of length capital T , which is given by V^T in the same manner. It is represented by $v(1)$ that is visible state, which is emitted by the machine at time step $t = 1$, similarly $v(2)$, similarly so this a visible state which is emitted by the machine at time step $t = 2$, similarly $v(3)$ and it continues up to $v(T)$. So, this is the sequence of visible states or visible symbols, which are emitted by the machine at different steps in time.

This is actually such sequence of observable symbols they actually constitute your pattern or temporal pattern. I have to recognize such temporal pattern using hidden Markov models, so that is the task that we have. So, as we have two different sets of states, one is the set of hidden states given by ω^T . And another is the set of visible states or visible symbols given V^T . We have said that, we have a transition probability a_{ij} . That is the probability of transition from state ω_i at time step t to state to ω_j at time step $t + 1$.

(Refer Slide Time: 51:51)

$a_{ij} = P(\omega_j(t+1) | \omega_i(t))$

$P(v_k(t) | \omega_j(t)) = b_{jk}$

$\left\{ \begin{array}{l} \omega \rightarrow \text{set of hidden states} \\ V \rightarrow \text{set of visible states/symbols} \\ a_{ij} \rightarrow \text{state transition probability} \\ b_{jk} \rightarrow \text{visible symbol emission probability.} \end{array} \right.$

So, we have said that this a_{ij} is equal to probability that the machine will be in state ω_j at step $t+1$, given the machine was in state ω_i at time step t . So, this was the probability of transition, in the same manner there is also a finite probability. State ω_j the machine will output a visible or a visible state, which is $v_k(t)$.

So, I also have the probability P that the machine will output the visible symbol $v_k(t)$ at time step t gives him the machine is in state ω_j at the same time state. So, while in the hidden state ω_j at time state at time step t , the machine outputs are visible symbol $v_k(t)$ at the same time step t with a probability given by $P(v_k(t)/\omega_j(t))$ and this probability is usually represented by b_{jk} .

So, this is the probability that the machine emits a visible symbol v_k or visible state v_k at state ω_j . So, this is what is b_{jk} ? So, whenever I have a hidden Markov model I have to have,

whenever I specify hidden Markov model I have to have three things. One is the set of ω_s which of the hidden states, a set of V , So, I have to have ω_s , which is nothing but set of hidden states. I have to have a set V , which is the set of visible states or visible symbols. I have to have a_{ij} , which is the state transition probability.

And I have to have b_{jk} , which is this also I can term as state transition probability. And this state transition probability is from a hidden state to a visible state, but actually this is the probability that my hidden Markov model emits a visible symbol v_k , when it is in the hidden state of ω_j .

So, this is actually visible symbol emission probability. So, these are the four different items which must be specified for specification of a hidden Markov model. So, I will stop this discussion today here. In next class I elaborate on this hidden Markov model. We will see that how the temporal pattern recognition can be achieved through hidden Markov models.

Thank you.

Pattern Recognition and applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 38
Hidden Markov Model

The discussion on the problem of recognition of temporal patterns, that is we said that we can have patterns which actually unfold in time. So, the examples under this category is one you can consider is speech recognition problem, because we know that the speech signals are actually time varying signals. So, at different instance of time we have different signal properties or we have different signal strength, so speech is one of the signals which is actually time varying signal. For speech recognition purpose what people usually does is the speech the signal is divided into a number of phonemes.

You try to find out in which sequence the phonemes occur and based on that you try to recognize or you try to classify the word that has been spoken. So, the other applications in this temporal pattern recognition may be say activity recognition say for example, we want to find out whether some person is walking, some person is running somebody is crawling and all that. It can be say a sign language recognition because sign language recognition is basically you expressed your ideas by using hand movements and it is the sequence in which the different parts of the hand move.

That actually gives you or sends you the information or you have to decode that particular sequence of hand gestures and from the sequence you have to interpret what message is been conveyed. Similarly, the activity recognition when I say running, walking, crawling this kind of applications is very useful for security surveillance purpose. So, we want to identify whether the movement of a person is a normal movement or it is an abnormal movement. So, through a video camera if I can capture that the movement of a person is that abnormal which obviously has to be detected based on the kind of movement a person does.

And obviously, it is a temporal sequence, so when I say the movement it is the body pose at different instance of time which occurs in a given sequence in a particular sequence and based on that the body poses that different instance of time. You try to identify whether the movement is a normal movement or abnormal movement and if the movement is abnormal you try to track that person because definitely a person going to an abnormal movement is a suspicious person. So, this temporal sequence or the patterns having

temporality in it have lots of applications. I have just given two or three examples, but there are many other applications, application domains in which this temporal sequence is need to be analyzed and they need to be classified.

Now, what we said in our previous class is that to recognize or to identify such temporal sequences, what we do is we make use of a machine which is something similar to with very similar to our sequential machine or finite state machine. So, as in we have a finite set of states through which the machine makes transition we have a finite set of output symbols we have a finite set of input symbols. So, if the machine is any states at time instant t , the machine will go to another state at the next time instant depending upon the input that is fed and when it makes a transition to the next state it emits an output.

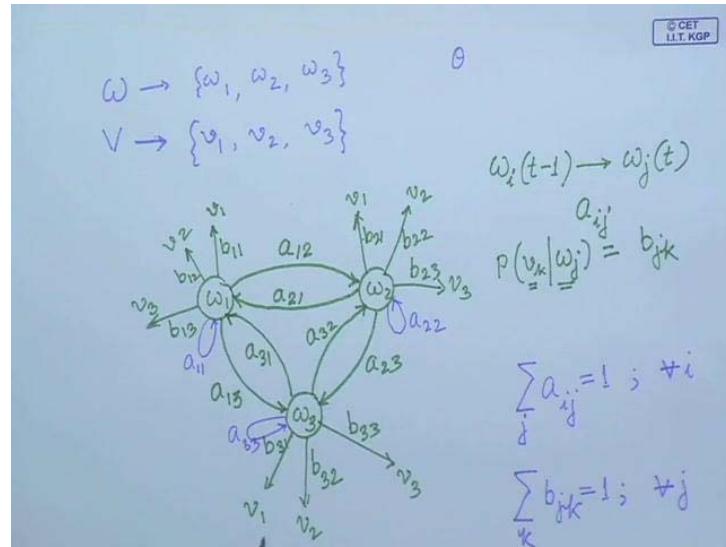
So, accordingly for a sequential machine I can have state transition table that is based on which input to which of the states the machine makes a transition. Similarly, I can also have a output table that for which type of transition for which type of transition from which state to which state what is the output the machine generates. It is this is output which actually absorbable state of a machine is not very absorbable.

So, I can identify the state from the input that is given and the output that is generated, so in this of a sequential machine we have said in this previous class in a sequential machine if the set of outputs is limited to only two symbols that is 0 and 1. The sequential machine of finite state machine actually becomes a finite state automaton and such a finite state automaton has got huge application in sequence generation, bit sequence generation or bit sequence detection. So, bit sequence detection is a problem which is very useful in communication because this is what gives you the synchronization if I want to Identify a point from were see information starts.

So, that starting of the information can have a particular bit sequence and I can have a finite state automaton to detect the bit sequence and once the bit sequence is detected, then I can say that the next portion of the sequence. Actually, it gives the information, but that sequence which is detected is only for the synchronization purpose. So, similarly, in case of hidden Markov model, we have said that there are two types of states one kind of state is say ω .

So, this set of states we said that they are the hidden states these states cannot be observed by the perceiver and we have another type another type of a states which are actually visible states and that type of states is called say V . So, this is a state of visible states, so hidden Markov model can have a number of hidden states and it can have a number of visible states.

(Refer Slide Time: 07:29)



So, let us take an example having a hidden of model having say three states, so I have under ω_i have states I have ω_1, ω_2 and ω_3 . So, these are the three states of this hidden Markov model say θ we are saying a model as θ and v is the state of visible states or the symbols visible symbols which are emitted by this hidden Markov model when this HMM θ is in a particular state.

So, let us assume that this can generate say three visible symbols v_1, v_2 and v_3 , so let us draw this HMM θ , so we have the states ω_1, ω_2 and ω_3 . So, the machine of this hidden Markov model can make a transition from state ω_1 to ω_2 where this probability of transition is given by a_{12} . As we said that if the hidden Markov model is in state say ω_i at time instance say $t-1$ and it Makes a transition to state ω_j at time instant t , and the probability of such transition from state ω_i to state ω_j is demoted by a_{ij} .

So, from ω_1 this hidden mark Markov model can make a transition to state ω_2 for the probability of transition is given by a_{12} , similarly it can have a transition from 2 to 1 for the probability of transition is a_{21} . Similarly, a_{13}, a_{23} , it will be a_{13}, ω_1 to ω_3 , this will be a a_{32}, ω_3 to ω_2 it will be a_{23}, ω_2 to ω_3 it will be a_{31} . So, these are transition probabilities between two different states, now in every state the machine can emit one of the visible states in each of these hidden states, the machine can emit one of the visible states.

So, from ω_1 it can generate, it can emit the visible symbol or visible state v_1 with this probability of this emission which is given by v_{11} because as we said that the probability that

the machine emits v_k when the machine is in state ω_j . This is actually given by b_{jk} , so from state ω_j it emits a symbol, a visible symbol v_k with the probability which is given by b_{jk} . So, from ω_1 it emits a symbol v_2 with a probability b_{12} it emits v_3 with probability b_{13} .

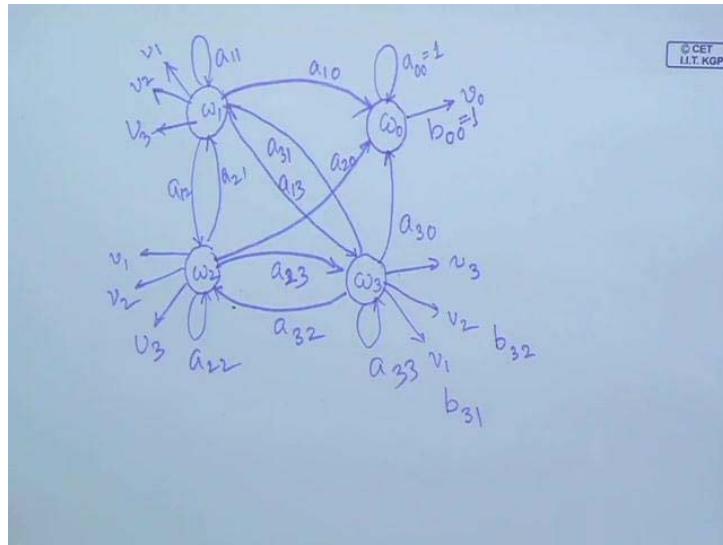
Similarly, here it will be v_1 is emitted with probability a_{21} , v_2 is emitted with probability b_{22} , v_3 is emitted with b_{23} from ω_2 , v_1 is emitted with probability b_{31} , v_2 is emitted with probability b_{32} and v_3 is emitted with b_{33} . So, these are the probabilities of emission of the visible states from different hidden states of this hidden Markov model. Now, you find that given any state there will always be transition to some of the states including the state itself.

So, I can have a transition from ω_1 to ω_1 also and this transition probability will be a_{11} , similarly ω_2 to ω_2 , this transition probability will be a_{22} . Similarly, ω_3 to ω_3 , this transition probability will be a_{33} , so this is also possible that in two different time steps, two subsequent time steps, the machine can be in the same state same hidden state. So, because from any hidden state there is always a transition to one of the states. So, naturally sum of a_{ij} for when I take the summation for all j , this will be equal to 1 and that will be true for all values of i because from any state the machine always makes a transition to one of the states.

So, the sum of the probabilities a_{ij} when I take the summation a_{ij} that has to be equal to 1 and that has to be true for all i and similarly when the machine is in any state it always emits a visible symbol. So, because it always emits the visible symbol, so I must have b_{jk} , when I take the summation over k , because v_k is the symbol and b_{jk} indicates that from hidden state ω_j . The machine emits a visible symbol v_k and it always emits a symbol, so this b_{jk} the sum of all k that also has to be equal to 1 and this will be true for all j that is for all the hidden notes.

So, given this and we have also said that hidden Markov model has a specific hidden state, which is called a receiving state or an accepting state or a final state and we have said that once the machine reaches the state it cannot come out of the state. So, all the transitions will be within the accepting states only and while in that accepting state it will emit only one visible symbol. If you see it will not emit any other visible symbol, so by incorporating that accepting state into our hidden Markov model, I can redraw this hidden Markov model something like this.

(Refer Slide Time: 15:00)

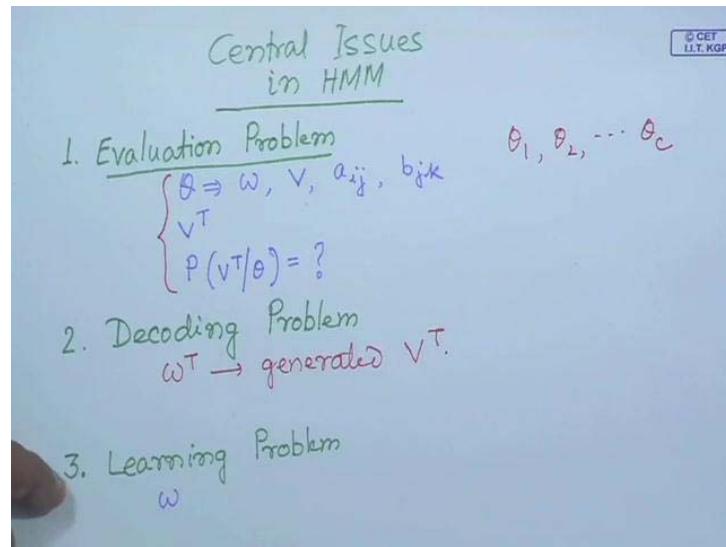


So, I have this ω_1 , I have say ω_2 , I have ω_3 and this is the accepting state which is ω_0 , so from ω_1 to ω_2 , I have transition a_{12} , ω_2 to ω_1 it is a_{21} . Similarly, all this different transitions, so after this I can have self-transitions this is a_{11} , a_{22} , a_{33} , but from any of the state it can make a transition to ω_0 . So, I can have this sort of transition, so here it will be here a_{10} , a_{20} , a_{30} , but once the hidden Markov model is in this final state ω_0 it cannot come back to any of the states. However, it can have transition within the state and because this transition will always be within ω_0 .

So, I must have a_{00} is equal to 1 and within ω_0 , it emits only one visible symbol which is v_0 , so you find that the probability of emission from ω_0 that is v_0 of the symbol v_0 that will also be equal to 1. This is the only symbol which is emitted from this accepting state whereas for all other states they have v_1 , v_2 , v_3 . Similarly, from here I will have v_1 , v_2 , v_3 from here also I will have v_1 , v_2 , v_3 with the respective probabilities of emission.

So, here from ω_3 , v_1 will be emitted with a probability b_{31} , similarly v_2 will be emitted with probability b_{32} and so on. So, this is the final hidden Markov model that we are going to use for our temporal pattern recognition. Now, given such a hidden Markov model you find that in hidden Markov models, there are three central issues which need to be addressed, so what are the central issues.

(Refer Slide Time: 18:18)



In hidden Markov model or HMM, the first issue or the first problem is an evaluation problem, the second problem is what is called a decoding problem. And the third problem is which is very, very important not only in case of hidden Markov model, a very important problem in all the classifiers is the training of the classifier or the classifier learning. Because unless the training is done properly, the classifier cannot classify the patterns, so the third problem which is very, very important problem is the learning problem that is how the hidden Markov model is actually learns the patterns or how do you train the hidden Markov model.

Now, what is this evaluation problem, now let us come to this problem first, we said that when we have a hidden Markov model θ , when this hidden Markov model θ is specified. We say that we have three things, one is the number of hidden states the number of whatever the hidden states, we have the visible states or visible symbols. We have the transition probabilities a_{ij} whenever the hidden Markov model makes a transition from state ω_i to state ω_j . We will have the transition probabilities corresponding to the visible states though that actually probability, but all the visible symbols are also considered to visible states.

So, we can also call that as a transition probability, so it does not matter whether we call it transition probability or we call it an emission probability. So, we have to have the transition probability a_{ij} for all i and for all j and we have to have the transition probability that is b_{jk} or the emission probability that is b_{jk} . This is the probability of emission of the visible symbol

v_k or visible state v_k from the hidden state ω_j . So, the model θ is specified by Ω that is the set of hidden states, V the set of visible states a_{ij} for all values of i and j , which are the transition probabilities state transition probabilities and b_{jk} which is the, which are the visible symbol emission probability

So, once you have such a hidden Markov model θ and you have a sequence of visible symbols V^T which is a sequence of length capital T . So, this evaluation problem is that given V^T and θ we have to find out what is the probability that V^T was generated by θ which is expressed as. So, this is what the probability that this is visible sequence V^T was generated by the hidden Markov model θ , but θ is specified over here. So, this we want to find out and this is represented as $P(V^T | \theta)$ and this is the problem which is evaluation problem.

And obviously once you evaluate this probability, we can try to classify we can try to classify this visible sequence V^T provided we know what this hidden Markov model is. So, you find that in case of hidden Markov model for recognition of temporal sequences if I have say have C number of sequences. And an input sequence or an unknown sequence is to be classified to this to one of number of sequences, so I have C number of model sequences for every sequence I have to have a hidden Markov model θ . So θ_1 , the first hidden Markov model will encode or will represent the first sequence, first model sequence model θ_2 will be for the second model sequence.

Now, θ_3 will be the third model sequence and so on, so once we have so many models say θ_1 to θ_C , θ_1 , θ_2 or to θ_C if I have C number of classes or if the input sequence is to be classified into one of the C classes. For every class, I have to have a Markov model, so every θ_i we have corresponding that is the state of hidden states it will have a state of corresponding V that is the state of visible states. It will have corresponding a_{ij} it will have corresponding b_{jk} , θ_2 will also have its corresponding a_{ij} , b_{jk} and all that, so every sequence or every class is represented by a separate hidden Markov model.

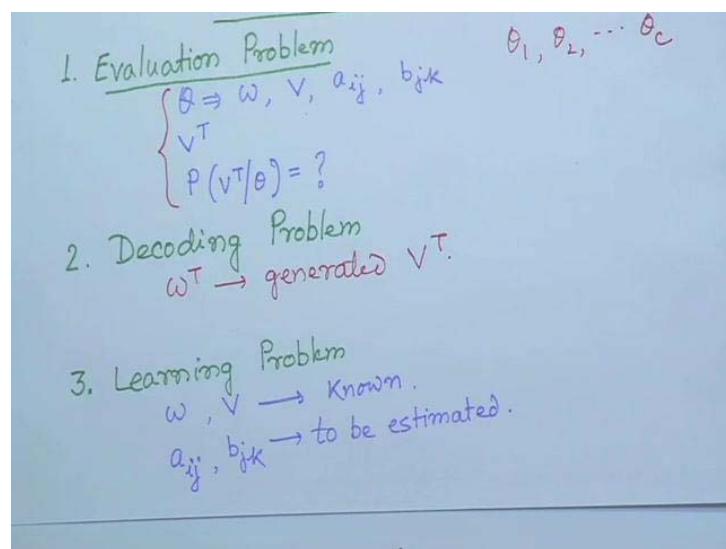
So, given an unknown sequence V^T we have to find out probability, what is the probability that V^T was generated by θ_1 , we have to find out the probability what is the probability that V^T was generated by θ_2 . Similarly, what is the probability that V^T was generated by θ_C and from all these different probabilities then we can apply the Bayes rule to classify the visible sequence V^T . So, the evaluation problem actually deals with this that given a model θ and V^T ,

a visible sequence symbols, we want to find out that what is the probability that V^T was generated by this model θ , so this is what my evaluation problem.

Now, the decoding problems are that I have this v^t and I have generated these probabilities what is the probabilities which was generated by the model θ ? The decoding problems says that what is the most likely sequence ω^T of the hidden states which had led to generation of V^T . So, we want to find out that sequence ω^T that has laid to the generation of, or which as generated this sequence V^T so that is what is the decoding problem and the learning problem is given a coarse structure or rough structure of the hidden model.

So, when I say the rough structure means that how many hidden states this hidden Markov model has and how many visible states this hidden Markov model has. So, that is what is a rough structure of a hidden Markov model, but I do not know the probabilities of transition the transition probabilities are unknown. So, through a set of given training sequences is I have to find out this transition probabilities, so I know what is ω that is the set of hidden states, I also know what is v that is the set of visible states.

(Refer Slide Time: 27:23)



So, these two are actually known, what I have to find out is I want to know a set of training sequences, a large number of training sequences is and using those training sequences, I have to estimate the transition probabilities a_{ij} and b_{jk} . So, these two transition probabilities are to be estimated. So, this is the problem which is known as learning problem and as we said that this learning problem is one of them is a very, very important problem not only in case of

hidden Markov model, but for designing any classifier. We will take up these central issues the evaluation problem, the decoding problem and the learning problem one by one.

(Refer Slide Time: 28:30)

Evaluation

$$P(V^T | \theta) = ?$$

$$P(V^T | \theta) = \sum_{r=1}^{r_{\max}} P(V^T | \omega_r^T) P(\omega_r^T)$$

$$\omega_r^T = \{ \omega(1), \omega(2), \dots, \omega(T) \}$$

~~r~~ → $N \rightarrow$ no. of hidden states

~~r~~ ≠ $r_{\max} = N^T$

So, first let us take up the evaluation problem, so as we said that evaluation is nothing but given a hidden Markov model θ and a sequence of visible states V^T , I have to estimate what is $P(V^T | \theta)$ that is what the probability is? This model θ has generated this sequence of visible V^T , now what I can do is I can find out all possible sequences of the hidden states of length T or length capital T. So, that means I want to find out or I can that is the crude approach that all possible sequence that as we said that this V is nothing but sequence of T number of invisible states.

So, when I have a θ , a number of visible states, I can find out all possible sequence of T number of visible states and for each of these sequence I can try to find out what is the probability that, that particular of hidden states ω^T has generated this visible sequence. So, what I can say is in a good form this $P(V^T | \theta)$ this can be estimated as I can find out what is

$P(V^T | \omega_r^T)$. Now, why writing omega r t am is, I said that I want to find out all possible sequence of hidden states of length capital T. This index r indicates one of those sequences it is one of those possible sequences.

So, I can find out what is $P(V^T \mid \omega_r^T)$ that is one of the possible sequence which is to be multiplied by $P(\omega_r^T)$ that is what is the probability that this r sequence occurs. And if I take the summation of this for $r = 1$ to r_{\max} , where r_{\max} is the number of such possible sequences that I can generate. So, if I take the summation r equal to 1 to r_{\max} of this, then what I get is $P(\theta \mid V^T)$ that is the probability that this sequence of visible states V^T is generated by this model θ . Now, over here this ω_r^T as I said that this is nothing but omega at time instant one, omega at time instant 2 continues omega at time instant capital T.

So, this is the sequence were for different sequences I have different values of ω_1 ω_2 and ω_3 and so on. Now, find that if there are say capital T number of hidden states if there are T number, let me put it because T values for something else. So, let me say if there are capital N number of hidden states this the number of hidden states, then you find that r_{\max} will be of the order of N^T . So, this is r_{\max} is N^T , so I will have N^T number of possible sequences of hidden states of length T. So, this is what the value of r_{\max} and here I can compute, you can compute that this $P(\omega_r^T)$ because this is nothing but the sequence of states, sequence of hidden states of length capital T.

(Refer Slide Time: 33:50)

The image shows handwritten mathematical derivations on a light blue background. At the top, the formula for the forward pass is given as:

$$P(\omega_r^T) = \prod_{t=1}^T P(\omega(t) | \omega(t-1))$$

Below it, the formula for the backward pass is given as:

$$P(v^T | \omega_r^T) = \prod_{t=1}^T P(v(t) | \omega(t))$$

Then, the overall probability of the sequence given the model is derived as:

$$P(v^T | \theta) = \sum_{r=1}^{r_{\max}} \prod_{t=1}^T P(v(t) | \omega(t)) \cdot P(\omega(t) | \omega(t-1))$$

At the bottom, a hand is pointing to the term $O(N^T \cdot T)$, which represents the complexity of the computation.

So, I can easily estimate that $P(\omega_r^T)$, this is nothing but $\prod_{t=1}^T P(\omega(t)/\omega(t-1))$. This is the sequence of T number of such states this is actually a probability a transition of state at time step t - 1 to another state at time step t. So, this is nothing but a probability of transition and from a particular sequence if I multiply this probability of transitions.

So, what I have to do is, I have to multiply this over, $t = 1$ to capital T, so if I take a product of this transition probabilities from $t = 1$ to T that gives me what is the probability of a particular sequence of hidden states. So, this $P(\omega_r^T)$ is nothing but transition probabilities, so this $P(\omega_r^T)$ is nothing but the product of the transition probabilities is from one state to another at subsequent or at consecutive time steps for this r^{th} sequence of invisible states or hidden states, similarly I can also compute $P(V^T/\omega_r^T)$, so you find that this is a visible symbol if I take a particular instant.

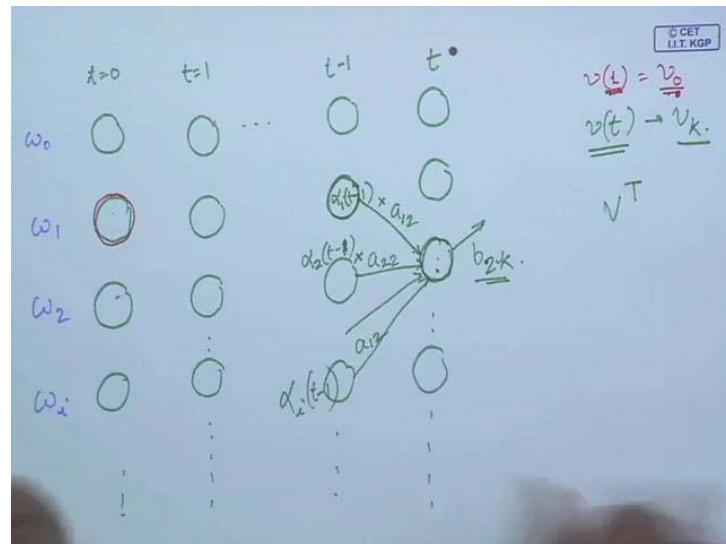
Say, first instant the visible symbol, the first visible symbol is emitted by the first state in the sequence ω_r , so this is nothing but the probability of emission of visible symbols. So, accordingly I can write this in the form this is the probability of emission of visible symbol at time step t given the machine is in emission at time step t. The product of this from t equal to 1 to capital T that gives me what is the probability $P(V^T/\omega_r^T)$. You find that this is nothing but $\prod_{t=1}^T P(v(t)/\omega(t))$.

So, because of this two, I can now write this $P(V^T/\theta)$, you find that $P(V^T/\theta)$ was this expression of p v T given θ . Now, in this expression if I put $P(V^T/\omega_r^T)$ the expressions of which we have said that we have set the product forms. So, I can write $P(V^T/\theta)$ will be is equal to $\sum_{r=1}^{r_{max}} \prod_{t=1}^T P(v(t)/\omega(t)).P(\omega(t)/\omega(t-1))$ say just look at this expression and this expression. So, this is nothing but this, so I have to do this, take this product is t equal 1 to T and to take the summation of this over r is equal to 1 to r_{max} .

So, given a hidden Markov model θ and a sequence of visible states V^T , I can find out the probability that this model θ has generated with this sequence of visible symbol because V^T as by this expression. Now, if I look at the complexity of this particular expression the complexity of this expression will be at the order of $N^T \cdot T$ which is huge. So, I have to have a high number of computations a very large number of computations to find out this $P(V^T / \theta)$.

So, instead of using this direct expression for computation given θ , I can make use of a simpler approach. I can make use of a recursive algorithm, the algorithm which will use that given a sequence of visible symbols V^T what will be the probability that the hidden Markov model will be in a particular state at a particular step in time. So, how is that possible let us just try to explain it conceptually, say I have a number of states is as I said one of the states is that I have is an accepting state or an absorbing state or a final state ω_0 .

(Refer Slide Time: 40:09)



So, I have state $\omega_0, \omega_1, \omega_2$ like this I have ω_i continues, I have a number of states, so I will put it like this, so these represents states and at different time steps I will have say number of states. So, these is a time step 0, $t = 0$, this is at $t = 1$, similarly I will have this is say $t-1$, I will have time step t , it continues like this. So, the first row indicates ω_0 , second row is ω_1 , ω_2 , say ω_i , somewhere I will have ω_j . So, you find that if somehow I know that at $t=0$ what is the hidden state in which the machine is?

So, let me assume that the machine is in hidden state ω_1 , then I can find out that at $t = 1$ what is the probability that the machine state in state ω_0 ? what will be the probability that the machines will be in state ω_1 ? What is the probability that the machine will be in state ω_2 ? what is the probability in the machine state ω_i ? And so on. The reason is from ω_1 , it can make a transition possible to make a transition to ω_0 , it is possible to make a transition to $\omega_1, \omega_2, \omega_i$ like this.

So, in $v(t)$ the symbol v_0 which is the visible state emitted by the final state only. So, if this is equal to this, then obviously from ω_1 I will have a transition to ω_0 because it is this only ω_0 which can emit this v_0 and it times step T, I have this visible symbol v_0 . If this is the case, then from ω_1 , I definitely have a transition to ω_0 if this is not the case if v_1 is not equal to v_0 . Then obviously from ω_1 , the model has not made a transition to ω_0 at time step $t = 1$, it has a transitions to some other states, now let us consider that any of the step.

So, I have said this time step t and so on, so let us consider this particular state ω_2 so what are possible ways that in which the machine can come to this state ω_2 ? Obviously, it cannot make a transition to ω_1 to ω_2 because as we said that this model enters the final state ω_0 , it cannot come out of that state. So, from ω_0 , it cannot make a transition to this, this state ω_2 , however it can make a transition to ω_2 from ω_1 ; it can make a transition to ω_2 to ω_2 . It can make a transition from ω_3 to ω_2 , it can make a transition ω_i to ω_2 and so on.

Now, if somehow I know what is the probability I have been able to compute the probability that I have been machine state this ω_1 at time instant $t-1$ and that is given by $\alpha_i(t-1)$. Here, it is $\alpha_2(t-1)$, here it is $\alpha_i(t-1)$, so this the probability that the machine is in these states at time instant $t-1$. Then I also know that what is the probability that the machine will take a transition from ω_1 to ω_2 which is given by the transition probability a_{12} . So, the probability that the machine can make a transition from ω_1 to ω_2 in time step t is given by this $\alpha_i(t-1)a_{12}$ plus I also have a possibility that machine will make a transition to ω_2 to ω_2 itself and this probability transition say 2 to 2. So, the probability that the machine makes a transition from ω_2 to ω_2 is nothing but $\alpha_2(t-1)a_{22}$. Here, it is $\alpha_i(t-1)a_{i2}$ and here if I make the sum of all these products what I get is the probability that there will be a transition from one of the states. From any of the states in time state $t-1$ to state ω_2 in time step t and I also know what is $v(t)$, that is the visible symbol in time step t .

This visible symbol in time step t can be emitted by this state ω_2 with a probability b_{2k} , suppose this emitted symbol is say v_k . So, this will have a probability b_{2k} , so finally I can say

that the probability that the machine is in state ω_2 in time step t , after emitting the first t number of visible symbols from my visible sequence of states V^T is given by sum of all this product terms multiplied by this b_{2k} . So, by using this logic I can simplify or I can have recursive algorithm to find out the probability that the sequence V^T was generated by hidden Markov model θ .

(Refer Slide Time: 47:42)

The image shows a handwritten derivation of the forward recurrence relation. At the top right, there is a small blue box containing the text "© CEF I.I.T. KGP". Below it, the formula for $\alpha_j(t)$ is given as:

$$\alpha_j(t) = \begin{cases} 0 & t=0 \text{ and } j \neq \text{initial state} \\ 1 & t=0 \text{ and } j = \text{initial state} \\ \left[\sum_i \alpha_i(t-1) a_{ij} \right] b_{j k v(t)} & \text{otherwise} \end{cases}$$

A hand holding a pen is visible at the bottom left of the image.

So, for that what I do is, I define the term say $\alpha_j(t)$ which as we have just explained that this $\alpha_j(t)$ this simply says what is the probability that the machine will state ω_j in times step t , after emitting first t number of visible symbols in the sequence of visible symbols V of capital T, so that is what and we can define that this $\alpha_j(t)$ will be equal to 0, if $t = 0$ and j is not an initial state, $t = 0$ means we are talking about our initial condition as per this diagram and the probability as I said that if I know that the machine is initially in state ω_1 .

So, I am saying that this is equal to 1, the probability of ω_1 in time step $t = 0$ is 1, the probability that the machine will be in any other states will be equal to 0. So, this is as per definition I said $\alpha_j(t)$ t is equal to 0 if I am in the initial state that is $t = 0$ and this j is not an initial state. If it is an initial state then this will be equal to 1 that is $t = 0$ and j is initial state, otherwise I define this $\alpha_j(t)$ as $\left[\sum_i \alpha_i(t-1) a_{ij} \right] b_{j k v(t)}$. You come to this diagram that we have

drawn this is, if the sum of all these products this is equivalent to $\left[\sum_i \alpha_i(t-1) a_{ij} \right] b_{j k v(t)}$.

So, what we have said here earlier is $v(t) = v_k$ then I take the initial probability or transition probability to b_{jk} that is from the j^{th} state at time t whatever symbol is emitted, I choose only that corresponding probability to decide what is the probability that the machine can be in state ω_j at time step t because I know what is the t^{th} visible symbol in my sequence of visible symbols of this, I can make use of that.

So, this is what I have, so this $b_{jkv(t)}$ indicates that this b_{jk} , that is the probability of transition is chosen by the t^{th} visible symbol, so if $v(t) = v_1$ this will be simply b_{j1} , if $v(t) = v_4$ this will be b_{j4} . So, I make use of this particular definition and using this definition, now write, I can write an algorithm to find out what is the probability that the machine θ , the HMM θ has generated this sequence of visible symbols V^T .

(Refer Slide Time: 52:03)

© CET
I.I.T. KGP

Algorithm (Forward).

Initialize: $t \leftarrow 0, \alpha_{ij}, b_{jk}, V^T, \alpha_j(0)$

for $t \leftarrow t+1$

$$\alpha_j(t) = b_{jk} v(t) \sum_{i=1}^N \alpha_i(t-1) \alpha_{ij}$$

until $t = T$

Return $P(V^T | \theta) \leftarrow \alpha_o(T)$ for final state.

end.

So, I write an algorithm and in particular I call it as forward algorithm the reason is why am writing forward is I will make use of a similar algorithm that is call a backward algorithm. Both this forward algorithm and backward algorithm will be used in a third issue or learning of the hidden Markov model. So, there will use forward and backward algorithm both of them together, so I will simply write this forward algorithm, so obviously the first one is initialization state.

So, I have to make time step $t = 0$, I have to know what is a_{ij} , I have to know what is b_{jk} , I have to know what is V^T because actually this is the one for which I am trying to find out the probability and I have to initialize $\alpha_j(0)$. So, this depends upon if any

initially assume that the machine is in state ω_1 , $\alpha_1(0) = 1$, $\alpha_2(0) = 0$, $\alpha_3(0) = 0$ and so on.

And then I go for, what I do is, for t increment, $\alpha_j(t)$ will be $b_{jkv(t)} \sum_{i=1}^N \alpha_i(t-1) a_{ij}$, where i varies from 1 to N as I have N number of hidden states.

And this has to continue until t becomes to capital T because I have to take all the symbols from given sequence of symbols. So, this has to continue for till t becomes equal to capital T.

At the end of this what I have to do is I have to return $P(V^T | \theta)$ which is nothing but $\alpha_0(T)$ and this $\alpha_0(T)$ is the probability of final state. So, this is for final state and the algorithm stops here.

So, this is the algorithm which can be used called forward algorithm used to find what is the probability that a given sequence V^T is generated by a given model θ and while doing so we have also found out that in every step, what is the probability that the machine can be given in a particular hidden state? So, will stop this lecture now, in the next lecture I will explain this algorithm further with an example.

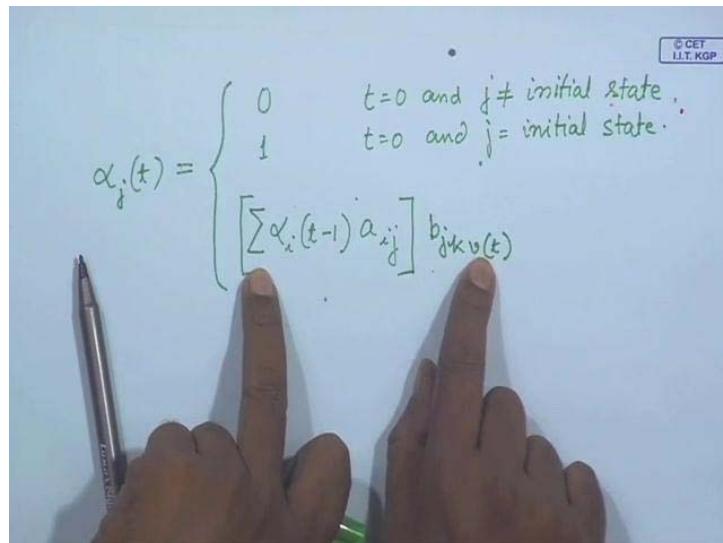
Thank you.

Pattern Recognition and Applications
Prof. P.K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 39
Hidden Markov Model
(Contd.)

Hello. So, in the last class, what we have done is, we have determined a probability that the hidden Markov model is in a particular state, say ω_i at time step t after generating first t number of visible symbols from the given sequence of visible symbols V^T .

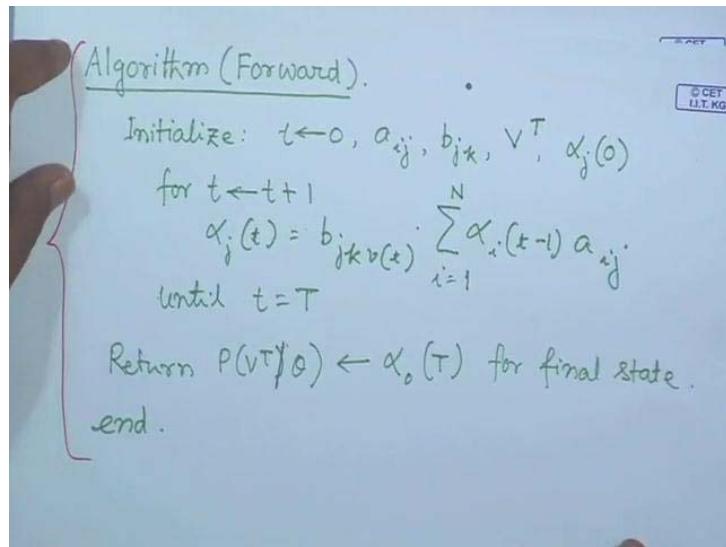
(Refer Slide Time: 00:59)



$$\alpha_j(t) = \begin{cases} 0 & t=0 \text{ and } j \neq \text{initial state}, \\ 1 & t=0 \text{ and } j = \text{initial state}, \\ \left[\sum \alpha_i(t-1) a_{ij} \right] b_{jv(t)} & \text{otherwise.} \end{cases}$$

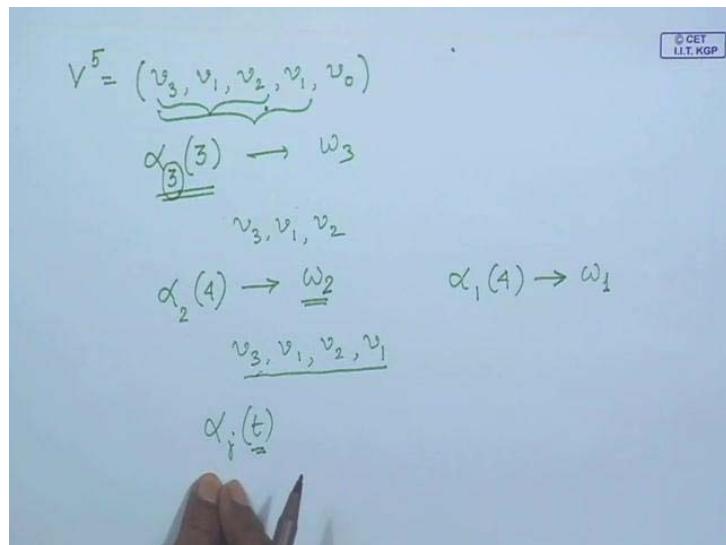
So, for doing that, what we have done is we have defined a term $\alpha_j(t)$, while we have said that initially $\alpha_j(t) = 0$, if the model is in time step $t = 0$ and this j is not an initial state. Whereas if we are in initial state at time step $t = 0$, then this $\alpha_j(t) = 1$ and otherwise $\alpha_j(t)$ will be given by this particular expression that is $\left[\sum \alpha_i(t-1) a_{ij} \right] b_{jv(t)}$. So, this actually tells you that that what is transition probability from all the states at the previous time step $t-1$, so that multiplied by the transition probability or the emission probability of the t^{th} symbol from state ω_j . So, that is what is $\alpha_j(t)$?

(Refer Slide Time: 02:14)



We had also written algorithm, which we have termed as forward algorithm to find out this value of this alpha j t and alpha j t when the machine reaches the final state which is say omega naught; that is what is the probability that the machine theta or hidden Markov model or theta model θ has generated the given sequence V^T . So, let us slightly elaborate on this concept what does this $\alpha_j(t)$ mean?

(Refer Slide Time: 02:49)



Suppose, the given V^T . So, let us take that we have sequence of five symbols and this sequence of five symbols are nothing but say v_3, v_1, v_2, v_1 and the final one as we said has to be v_0 . So, this is the sequence of symbols V^5 that we have. If we want to find out the term say α_3 , at

$t = 3$, this says what is the probability that the machine or the hidden Markov model will be in state ω_3 and it will be in state ω_3 after generation of first three symbols that is v_3, v_1, v_2 .

So, this after generating first three symbols, what is the probability that the machine will be in state ω_3 ? Similarly, if I say what is $\alpha_3(4)$, this says what is the probability that the machine will be in the hidden state ω_2 after generating first four symbols? That is, it has already generated v_3, v_1, v_2 and v_1 . So, these are the symbols, sequence of symbols, which have already been generated and after generation of these four symbols, what is the probability that the machine will be in state ω_2 ?

Similarly, $\alpha_1(4)$, this will specify, this will actually indicate that after generating these four symbols, what is the probability that the machine will be in state ω_1 . So, this is what this $\alpha_j(t)$ means. So, this $\alpha_j(t)$ means that after generating first t number of symbols, what is the probability that the machine will be in state ω_j . So, let us elaborate on this with the help of an example.

(Refer Slide Time: 05:31)

Example.

© CET
 I.I.T. KGP

$\omega_1, \omega_2, \omega_3, \underline{\omega_0} \rightarrow$ hidden States

$v_0, v_1, v_2, v_3, v_4 \rightarrow$ visible States.

$$a_{ij} = \begin{bmatrix} \omega_0 & \omega_1 & \omega_2 & \omega_3 \\ 1 & 0 & 0 & 0 \\ 0.2 & 0.3 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.7 & 0.1 & 0.1 & 0.1 \end{bmatrix}_{\omega_i} \quad b_{jk} = \begin{bmatrix} v_0 & v_1 & v_2 & v_3 & v_4 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0.4 & 0.1 & 0.2 \\ 0 & 0.1 & 0.1 & 0.7 & 0.1 \\ 0 & 0.5 & 0.2 & 0.1 & 0.2 \end{bmatrix}_{v_k}$$

$$\sum_j a_{ij} = 1 ; \forall i$$

$$\sum_k b_{jk} = 1 ; \forall j$$

So, we consider an example. So, first what I have to specify is, I have to specify the model θ and to specify the model θ , we have said that we have to have what are the numbers, what are the hidden states, what are the visible symbols or visible states and what are the state transition probability tables that is a_{ij} and b_{jk} ? So, I assume that the machine has let us say four hidden states including the final state. So, I will have $\omega_1, \omega_2, \omega_3$, and the final state, which is ω_0 , let me write it in a different color.

So, ω_0 is the final state and in addition to this final state, it has three more states ω_1 , ω_2 and ω_3 and all these are hidden states. And at the same time, I also have to have that the transition probabilities for the visible states or visible symbols and for that, let me assume that there are four different visible symbols, which are v_1 , v_2 , v_3 and v_4 . So, these are the hidden states and these are the visible states and in addition to these visible states, we also said that when the machine is in hidden state ω_0 , which is the final state or absorbing state, in that state, the machine emits only one visible state. So, that particular visible state let us say that it is v_0 . So, these are the visible states that I have v_0 , v_1 , v_2 , v_3 and v_4 and we also have the transition probabilities. So, the transition probability is a_{ij} . So, let me assume

that the a_{ij} is specified as
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.3 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.7 & 0.1 & 0.1 & 0.1 \end{bmatrix}$$
. So, these are the transition probabilities among the hidden states.

And similarly, I have to have the transition probabilities for the visible states, which actually said that it is the emission probability that is b_{jk} . So, I write b_{jk} , which will be given by say

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0.4 & 0.1 & 0.2 \\ 0 & 0.1 & 0.1 & 0.7 & 0.1 \\ 0 & 0.5 & 0.2 & 0.1 & 0.2 \end{bmatrix}$$
 So, these are the two different transition probabilities tables. The first one

is the transition probability table or a_{ij} as shown over here. This is the transition probability table for the hidden states and b_{jk} , which is shown over here, this indicates that the transition probability table for the visible states. You find that both these tables are indexed from 0. So, this is for ω_0 , ω_1 , ω_2 and ω_3 .

Similarly, the rows are ω_0 , ω_1 , ω_2 and ω_3 . Similarly, for b_{jk} , this is for v_0 , v_1 , v_2 , v_3 , v_4 and this is ω_0 , ω_1 , ω_2 , and ω_3 . So, as we said earlier that once the machine enters the final state ω_0 , it remains in ω_0 only. It cannot come out of the ω_0 . So, here you find that once it is in ω_0 , a_{00} is equal to 1, but a_{01} is 0, a_{02} is 0, a_{03} is 0. That means that once the machine is in state ω_0 , it will always remain in ω_0 . The transition probability to ω_0 is equal to 1. Transition probability to any other state is 0, so which indicates that from ω_0 , the machine cannot make a transition to any state other than ω_0 . So, it will remain in the accepting state only.

Similarly, if you look at this transition table, which is actually the symbol emission table in ω_0 , here it says that b_{00} is equal to 1 that in state ω_0 , the machine will only emit symbol v_0 . It will not emit any other symbol, whereas from any other state, there is a finite probability that machine can emit any of the symbols except v_0 , say b_{10} is equal to 0; that indicates that from state ω_1 , the machine cannot emit those visible symbols v_0 , v_0 will only be emitted from ω_0 and this is the only symbol that is emitted from ω_0 . We had also put two more constants.

We had said that sum of a_{ij} has to be equal to 1 when you take the summation over j. So, here you find that if you take the sum of any of the rows, the sum will be equal to 1 and this is true for all i that means for all the rows, this has to be true. Sum of the transition probabilities in a particular row is always equal to 1. Similarly, the other constants that we said that sum of b_{jk} , when you take the summation over k that will be equal to 1 and this is true for all j.

So, here also, we find that if you take the sum of all the elements in any of the rows that is also equal to 1. So, these are the restrictions on these transition probabilities that we have to have. Now, given these transition probabilities, my problem is I want to find out that if we are given sequence of visible states, what is the probability that that sequence of visible states is actually generated by this model θ ? So, for that let me take a sequence of visible state.

(Refer Slide Time: 14:17)

$$V^4 = \{v_1, v_3, v_2, v_0\}$$

$$t=0; w_1.$$

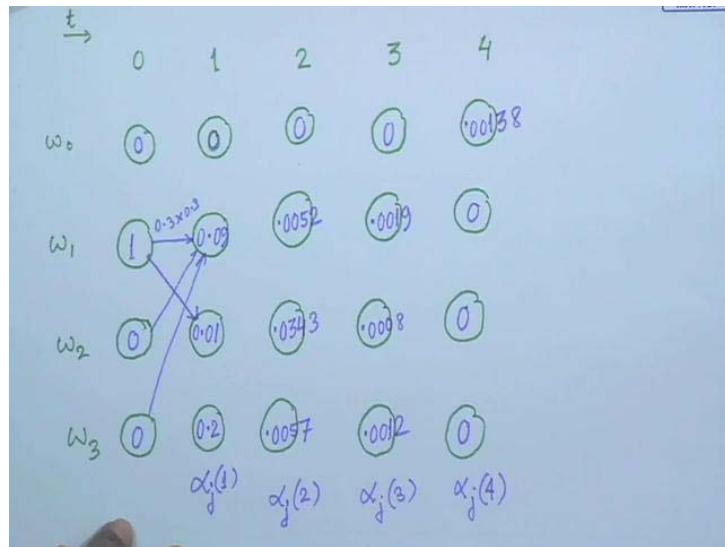
$$p(V^4 | \theta) = ?$$

Say the sequence; we have the sequence of four visible states V^4 , which is given by v_1, v_3, v_2 and v_0 . And as we said that we have to have an initial state, we assume that the machine is initially at t equal

to 0, the machine is in one of the hidden states. So, I assume that at $t = 0$, the machine is in state ω_1 . So, this is my assumption and based on this, I have to find out what is $P(V^4 / \theta)$, where this θ is specified by state of hidden states, the state of visible states, the transition probability tables a_{ij} and b_{jk} . So, this part is a_{ij} .

Let me just have a demarcation between these two. So, this is the a_{ij} and this is the b_{jk} . So, these four quantities, they specify my model θ . So, what I have to find out is that given this sequence of visible states, what is the probability that that machine θ has generated this sequence of visible states? So, this is what I have to find out and that is what is my evaluation problem. So, in order to solve this problem, let us have a diagram, which is called trellis diagram.

(Refer Slide Time: 15:57)



So, at different instants of time, I will put t in this direction at $t = 0, t = 1, t = 2, t = 3$ and $t = 4$. At $t = 1$, I have different states, $t = 0$, I have different states. This is $\omega_0, \omega_1, \omega_2$ and ω_3 . So, let me put ω_0 in the first row, ω_1 in the second row, ω_2 in the third and ω_3 in the fourth row. So, this is the situation I have and as we said that at t equal to 0, we have assumed that the machine is in state ω_1 .

So, naturally $\alpha_1(0), \alpha_1(0)$ that has to be equal to 1 as per our definition because we have defined this α in that form. If you come to this definition at $t = 0$, if j is initial state, then $\alpha_j(t) = 1$. If it is not an

initial state at $t = 0$, then $\alpha_j(t) = 0$. So, just by going, just going by this definition as I have assumed that at $t = 0$, the machine is in state ω_1 . So, I have $\alpha_1(0) = 1$, $\alpha_0(0)$ will be equal to 0, $\alpha_2(0)$ will be 0, $\alpha_3(0)$ will be 0. So, this is the initialization, initial states that I have. Now, let us see what happens at times step $t = 2$. So, here again, I have all these four different states. Now, you find that coming to state ω_0 , I have various possible paths. I can have a transition from ω_1 to ω_0 , I can have a transition from ω_2 to ω_0 , and I can have a transition from ω_3 to ω_0 .

So, if I do this transition, then as we said over here that $\alpha_1(0)$ at time step zero, what is the value of $\alpha(1)$ that will be given by this expression and in this expression, I have multiplied by this transition probability. Where this transition probability is dictated by v_1 and what is the value of v_1 if you look at this $b_{jkv(1)}$, if you look at this and look at transition, the sequence of symbols that I have at times step t 1, the visible state is v_1 . And if you look at this, this is my hidden state 0 and the probability of emission of v_1 in state 0, in state ω_0 is equal to 0.

So, whatever path I choose, the probability that this state will exist, the existence of this state will have certain probability at time step at $t = 1$ that is equal to 0. So, as I am using this color, let me use this color. So, that is equal to 0. Then let us come to this ω_1 . You find that I have various ways in which I can have a transition to state ω_1 . I can have a transition to state ω_1 , from ω_0 , I cannot have a transition because this transition probability is equal to 0, ω_0, ω_1 , and this is equal to 0. So, I cannot have a transition to state ω_1 from 0. I can only have transition to omega from ω_1 to ω_1 . I can have transition from ω_2 to ω_1 . I can have transition from ω_3 to ω_1 .

Now, this transition probability from ω_1 to ω_1 , if you look at this, that is equal to 0.3 multiplied by what is the probability that in state ω_1 at time step 1, it will output a symbol which is v_1 because my first symbol is v_1 . So, the probability that from state ω_1 , it outputs the symbol v_1 is equal to 0.3. So, over here, it will be 0.3 that is the transition probability from ω_1 to ω_1 multiplied by 0.3 again, which is the emission probability of the visible symbol v_1 from state ω_1 . So, this is 0.3 into 0.3. If you come from here, this probability is 0.

So, the contribution to this will be 0. This probability is 0. So, the contribution to this will be equal to 0. So, this $\alpha_1(1)$, the value of this will be 0.09. In the same manner, here what I have is ω_1, a_{12} that is the transition probability from ω_1 to ω_2 , a_{12} . Now, the contribution of these two to this will be equal to 0 because $\alpha_2(0)$ is 0, $\alpha_3(0)$ is 0. So, that contribution from this to this is 0. I will have only

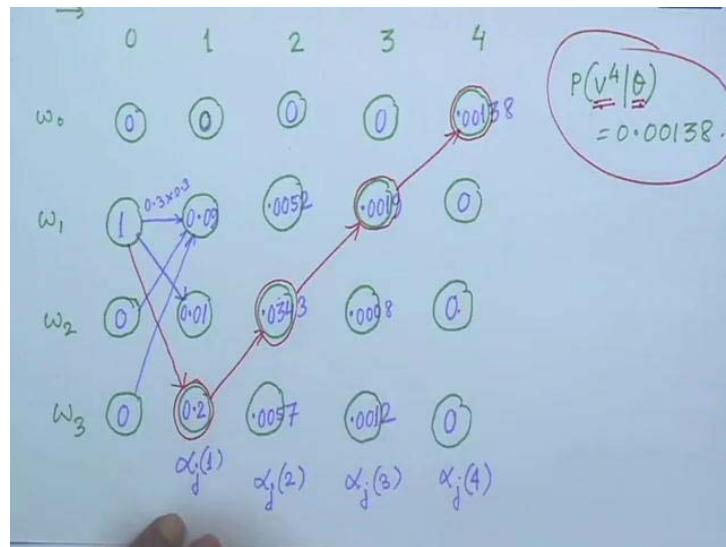
contribution from here to here and this again, if you look at this one, the transition probability from ω_1 to ω_2 , ω_1 to ω_2 that is 0.1 and the emission probability of v_1 from ω_2 , v_1 from ω_2 that is also 0.1.

So, this contribution will be 0.1×0.1 , which is nothing but 0.01. In the same manner, I can compute that this will be 0.2. So, these figures in this circle says that this is the probability that my model will be in hidden state ω_1 at $t = 1$ after emitting the first symbol, which is v_1 . And if you continue like this at different time steps, so I will have $\omega_0, \omega_1, \omega_2, \omega_3$, time step 3, $\omega_0, \omega_1, \omega_2, \omega_3$. This is also $\omega_0, \omega_1, \omega_2$, and ω_3 . If I continue in the same manner, you will find that here also in time step 2, $\alpha_0(2)$ will be equal to 0, in time step 2, $\alpha_1(2)$ will be equal to 0.0052. You can do this computation. This is nothing but this transition probability multiplied by this multiplied by $b_{jkv(t)}$ and this case b_{jk} is nothing b_{13} and my symbol is v_3 .

So, this will be b_{j3} or b_{13} . So, this probability multiplied by a_{11} multiplied by b_{13} plus this probability 0.01 multiplied by a_{21} multiplied by v_{13} , similarly, from here. So, if I add all these defined terms, what I get is 0.0052. If you compute in the same manner, here it will be 0.0343; here it will be point 0.0057. So, over here, what I get is alpha 2 at time step two for all j, this is for j equal to 0, j equal to 1, j equal to 2, j equal to 3. This gives alpha 1 for all j. In the same manner, if you compute at t equal to 3, so here I have $\alpha_j(3)$, for j equal to 0, I get this. For j equal to 1, I get this. For j equal to 2, I get this. For j equal to 3, I get this.

So, here again, this term will be equal to 0. This figure will be point 0.0019. This will be 0.0008 and here it will be 0.0012. Coming over here $\alpha_j(4)$, if you compute, this will be 0.00138, this will be 0, this will be 0, and this will be 0. And you find that now if you look at this algorithm, the forward algorithm that we had written, the final probability that the machine θ has generated this sequence V T is given by $\alpha_0(T)$ that was my algorithm and here $\alpha_0(T)$ is nothing but 0.00138.

(Refer Slide Time: 28:03)



So, the probability that our machine that you have considered here has generated the given V^4 that is this, this sequence for the given machine or hidden Markov model θ , where Markov model θ is defined by these parameters is equal to 0.00138. So, this is the probability that our given model θ has generated that given sequence and here at every state, the term the figure within the circle indicates the probability that the model will be in ω_2 , so here in ω_2 , at times step 2, after generating v_1 and v_3 as per this sequence.

So, it has generated v_1 and v_3 . After generating these first two symbols, the probability that the machine will be in step ω_2 is given by this. Similarly, after generating these two symbols v_1 and v_3 , the probability that the machine will be in state ω_1 is given by this, which is 0.0052. So, here what you are getting is the probability that the machine has generated this V^4 . The model θ has generated this V^4 . This generation may be by any of the paths because when computing these probabilities at every time step, I have considered that the contribution from all the paths.

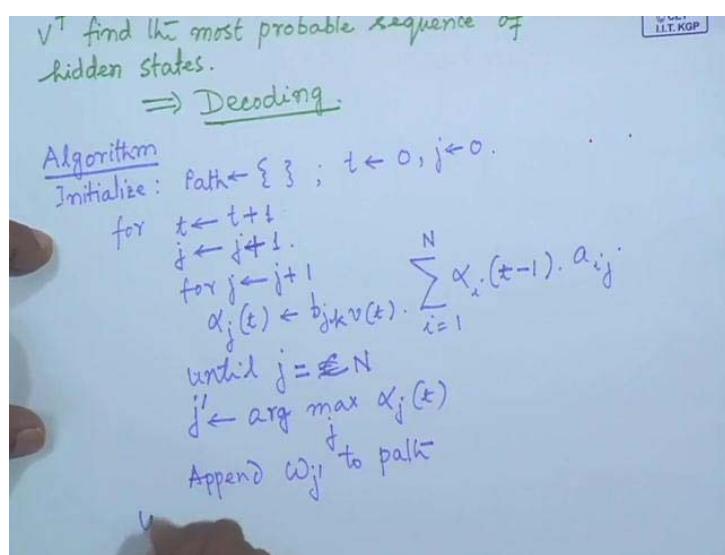
So, this is the probability that this V^4 has been generated by machine θ , by this model θ and while generation of this V^4 , the model can make use any of this paths. So, this is what we said is what evaluation problem. The second problem that we have said is the decoding problem and what is we have said in the decoding problem is that what is the most likely sequence, or most probable sequence of hidden states through which the machine has transited while generating that V^T .

And if I try to find out this most probable sequence of hidden states through which the machine has made the transitions while generating V^T , I can simply say that at every time step, I can only consider that state which is most probable because you come over here, the most probable state where the machine can be, in which the machine can be at $t = 1$ is 0.2 because the probability that the machine will be in state ω_3 after generating the first symbol is 0.2, whereas the same probabilities for the other states is less than 0.2.

So, I can say that in time step one, this is the most probable state. Similarly, in time step 2 at $t = 2$, the most probable state in which the machine can exist is ω_2 because here the probability that the machine will be in state ω_2 after generating first two symbols is 0.034, whereas for ω_1 , it is 0.0052, for ω_3 it is 0.0057, for ω_0 , it is equal to 0. Obviously, here we cannot reach. So, this is the next most probable state after generating the first two symbols. Similarly, over here, the next probable state after generating the first three symbols is this and obviously the final state in the sequence is the absorbing state or the final state which is ω_0 . So, once I have this trellis diagram, the decoding problem is very, very simple. I know my initial state was ω_1 .

So, the sequence of states through which the machine transits while generating this sequence V^4 is equal to v_1, v_3, v_2 and v_0 is the sequence of states $\omega_3, \omega_2, \omega_1$ and ω_0 . So, this is the most probable sequence of states and that is what is my decoding problem. Once from the trellis diagram, I have this simple method to find out, to solve the decoding problem, I can also write an algorithm for this decoding problem and this algorithm is very simple that straight away comes from this diagram.

(Refer Slide Time: 33:52)



So, the decoding problem is we said that given V^T ; find the most probable sequence of hidden states. So, this is what was my decoding problem that was the second issue in our hidden Markov model and obviously, I can write an algorithm for doing this. So, in algorithm, basically what I am trying to do is I am trying to find out the path, the most probable path, which is nothing but the sequence of the hidden states through which the model will make transitions.

So, I will have as before an initializing step. In initialization, I set path to be an empty set and I initialize t to 0. Then I will have a number of iterations. So, I set go for iterations with increment in t and increment in an index j . Then, for j to $j + 1$, I put $\alpha_j(t)$ will get $b_{j|v(t)} \cdot \sum_{i=1}^N \alpha_i(t-1) \cdot a_{ij}$, i going from 1 to N as N is the number of hidden states that I have and this will continue until j , where N is the number of hidden states. Once this condition is reached, then what I have to do is I have to put j' which is nothing but I have to find out the state, which is having the maximum probability. So, this $\underset{j}{\operatorname{argmax}} \alpha_j(t)$, where this maximum has to be computed over j .

(Refer Slide Time: 37:29)

```

Algorithm
Initialize: Path ← {} ; t ← 0, j ← 0.
for t ← t+1
    j ← j+1
    for j ← j+1
         $\alpha_j(t) \leftarrow b_{j|v(t)} \cdot \sum_{i=1}^N \alpha_i(t-1) \cdot a_{ij}$ 
    until  $j = N$ 
     $j' \leftarrow \underset{j}{\operatorname{arg max}} \alpha_j(t)$ 
    Append  $w_{j'}$  to path
until t = T.
Return Path

```

Then, append j' to path and you have to repeat this until t becomes equal to capital T , which is the length of the sequence and at the end of the algorithm, what we have to do is we have to return the path. So, if you run this algorithm for this given problem, we will find that we will come out with this path only. So, this is what is known as the decoding problem that is I find out the most probable

sequence of hidden states through which the machine makes a transition while generating the sequence of the visible symbols v t.

At the end, what we get is what is the probability that the given sequence of the symbols has been generated by the given machine θ , where θ is actually specified by the state of states, the state of visible states and the state of hidden states along with two transition probability tables, probability transition tables. One is the transition probability from the hidden state to the hidden state and the other one is the transition probability from the hidden state to the visible state.

And this is not really a transition. What we can call is that this is probability of emission of with visible states from different hidden states. So, these four quantities define my model θ and given this model θ and given sequence of visible symbols, I want to find out what is the probability that the model θ has generated this sequence of visible symbols V^T . Now, my final goal is I want to classify this sequence of these symbols. So, now you can recollect that when you talked about the Bayes rule, we have said that we have class conditional probability functions.

(Refer Slide Time: 40:09)

The image shows a handwritten derivation on a light blue background. At the top left, it says $p(x|\omega_i)$. Below it, there is a red arrow pointing right with the text $x \rightarrow p(\omega_i|x)$. To the right of this arrow, there is a fraction: $p(\omega_i|x) = \frac{p(x|\omega_i) \cdot p(\omega_i)}{p(x)}$. Below this fraction, there is a red arrow pointing right with the text $p(\omega_i|x) > p(\omega_j|x)$. At the bottom right, there is a red arrow pointing right with the text $\Rightarrow x \in \omega_i$.

That is what we had is something like $P(X/\omega_i)$. This was class conditional probability function. What can be estimated now from an unknown sample say X to classify this in one of the classes, what we needed to compute is $P(\omega_i/X)$ and that particular i for which $P(\omega_i/X)$ is maximum, this unknown sample was classified to that particular class. And we have said in that case, that what Bayes theory

says is, it computes this is this $P(\omega_i/X)$ is called a posteriori probability. And $P(X/\omega_i)$ is called a class conditional probability and along with that, we can have a priory probability, what is the probability of the occurrence of the particular class.

And we can combine this class conditional probability density function with a priory probability by using the Bayes rule, which states that $P(\omega_i/X)$ is nothing but $P(X/\omega_i) \cdot P(\omega_i)$. So, if I have two classes and the same unknown sample X, I will compute $P(\omega_i/X)$. I will also compute for another class ω_j , $P(\omega_j/X)$. So, these are two aposterior probabilities. If $P(\omega_i/X) > P(\omega_j/X)$, then we conclude that X belongs to class ω_i . So, this is what we have done in Bayes classification.

(Refer Slide Time: 42:22)

The image shows a handwritten derivation of Bayes' rule for a Hidden Markov Model (HMM). It starts with the expression $P(v^T|\theta)$. Below it, $P(\theta|v^T)$ is underlined twice. The formula is then written as:

$$\underline{\underline{P(\theta|v^T)}} = \frac{P(v^T|\theta) \cdot P(\theta)}{P(v^T)}$$

Below the formula, θ_i and θ_j are written. A red box encloses the following inequality and conclusion:

$$P(\theta_i|v^T) > P(\theta_j|v^T) \Rightarrow v^T \in \theta_i.$$

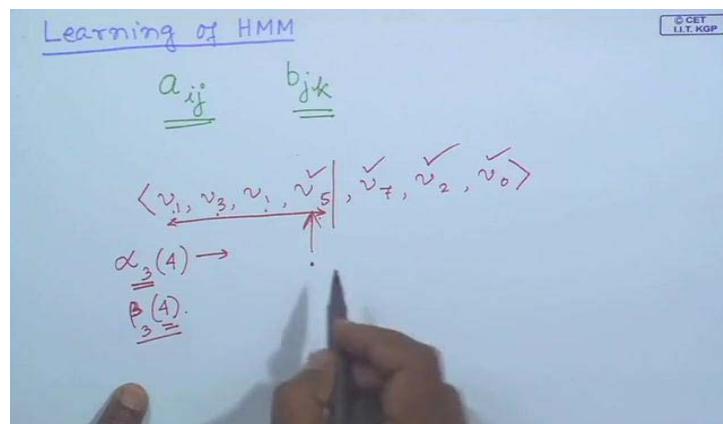
In the same manner, in this particular example or in this particular case of hidden Markov model with a sequence of visible symbols, what we are finding out is $P(V^T/\theta)$ that is what is the probability that V^T has been generated by θ , but what we need for classification purpose is $P(\theta/V^T)$. That is given this visible sequence, what is the probability that this sequence belongs to a class θ .

So, here again, we can apply this Bayes rule. So, this Bayes rule can simply be applied as $P(\theta / V^T)$ to be taken as $P(V^T / \theta) \cdot P(\theta)$, which is the apriory probability upon probability of this sequence V^T irrespective of the models. And once we have this quantity $P(\theta / V^T)$, then I can use this quantity for classification. So, if I have two models, one model is θ_i , the other model is θ_j . I can compute this term for θ_i , I can compute this term for θ_j , I can compute this $P(\theta_i), P(\theta_j)$ which are the apriory probabilities. Then I have this aposteriori probability $P(\theta_i / V^T)$; I also have a aposteriori probability $P(\theta_j / V^T)$. If $P(\theta_i / V^T) > P(\theta_j / V^T)$, then obviously my interpretation will be that V^T belongs to class θ_i .

So, this is how I can classify a sequence, a time sequence of visible symbols. So, this will be my classification. So, out of the three major issues that we have said that three central issues in a hidden Markov model, what are the issues we have said? We have said that first issue is the evaluation, where we try to compute what is a probability that a model has generated in a given visible sequence. The second issue was decoding issue where we have found out, what is the most probable sequence of hidden states through which, the model has made transitions to generate the given sequence of visible symbols.

The third central issue, which is a very, very important issue, is learning of the hidden Markov model, training of the hidden Markov model. Again, as we discussed before, when we talked about the supervised learning and unsupervised learning when you try to train the hidden Markov model, you make use of a number of sequences of visible symbols and you already know to which, what is that visible symbol. As using a number of known visible sequences, you try to train the hidden Markov model. So, the training or learning process of hidden Markov model is actually a supervised learning.

(Refer Slide Time: 46:31)



So, now what we try to discuss is learning of hidden Markov model, which as we have said it is very important issue. So, before learning what we said is the hidden Markov model is coarsely specified that is I have a very coarse representation of the hidden Markov model in terms of I know what is the number of hidden states of the hidden Markov model. I know what are the visible symbols, which are generated by the hidden Markov model?

So, these are the two quantities, which are known that is the hidden states and the visible states or visible symbols. So, by learning of hidden Markov model or by training a hidden Markov model, what we usually mean is that we have to estimate the transition probabilities a_{ij} and b_{jk} . So, I know what are the hidden states? I know, what are the visible states? We have to estimate is the transition probabilities a_{ij} and b_{jk} .

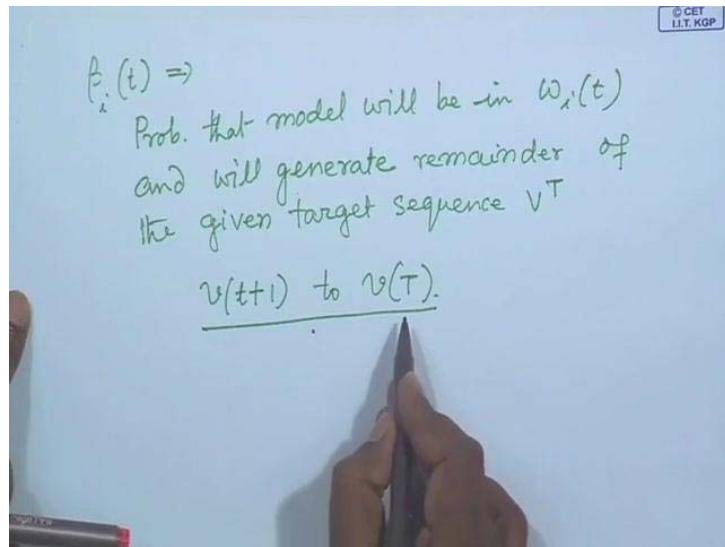
So, these are the two parameters of the hidden Markov model, which needs to be estimated through the learning process. And for the estimation of this a_{ij} and b_{jk} , as we said this is the supervised learning or supervised training process. So, we make use of us number of known sequences, the sequences for which we known that to which of the class those sequences belong. So, using this, we go for the training of the hidden Markov model.

Now, you remember when I talked about this forward algorithm, this one, that time we said for training of the Markov model or for learning, we use a similar algorithm, which is the backward algorithm. So, in case of forward algorithm, we have said that this $\alpha_j(t)$, this actually tells what is the probability that the model will be in state ω_j at time step t after generating first t number of visible symbols in the given sequence of visible symbols V^t , When I go for backward algorithm, the backward algorithm says that what is the probability that the machine or the model will be in state ω_j at time instant t and will generate the remaining set of the sequence of visible symbols? So, what I mean to say is that if I have a set of visible symbols, which are given like this $v_1, v_3, v_1, v_5, v_7, v_2, v_0$, suppose this is the sequence of visible symbols.

So, $\alpha_3(4)$, this gives me what is the probability that the machine will be in state ω_3 after generating v_1, v_3, v_1 and v_4 up to this. So, the machine has generated these symbols in the given sequence and then what is the probability that it will be in the hidden state ω_3 , whereas if I write say $\beta_3(4)$, what it will say is what is the probability that the machine will be in state ω_3 , it will generate the remaining four symbols of the given sequence, that means it will generate v_7 , it will generate v_2 , it will generate v_1

and of course, as I said 4, so v5. So, what is the probability that in this time, the machine will be in state ω_3 and it will generate v_7, v_2, v_1 .

(Refer Slide Time: 51:44)



So, in other words what I can say is that this $\beta_i(t)$, this actually represents the probability that model.

So, this $\beta_i(t)$ gives you what is the probability that the model will be in state of ω_i in time step t and it will generate remainder of the given target V^T , that means it will generate all the visible symbols $v(t+1)$ to $v(T)$. So, all the symbols from $v(t+1)$ to $v(T)$ that will be generated and the machine is in state ω_i at time step t. So, that is what is $\beta_i(t)$. So, in the forward algorithm, we find out that what is the probability machine is in the state ω_i after generating first t number of symbols and $\beta_i(t)$ in the backward algorithm that tells you what is the probability that the machine is in state of ω_i and it will generate the remaining part of the symbols of the target sequence. So, accordingly I can write the definition $\beta_i(t)$.

(Refer Slide Time: 53:48)

$$\beta_i(t) = \begin{cases} 0 & \omega_i(t) \neq \omega_0 \text{ and } t = T \\ 1 & \omega_i(t) = \omega_0 \text{ and } t = T \\ \sum_j \beta_j(t+1) a_{ij} b_{jkv(t+1)} & \text{otherwise.} \end{cases}$$

I can define this $\beta_i(t)$. So, this $\beta_i(t)$ will be equal to 0 if $\omega_i(t) \neq \omega_0$ and t is equal to the last symbol, in the last time state that is $t = T$ because as we said that the last state hidden state has to be ω_0 in which the machine generates the only visible symbol, which has to be 0. So, if this condition is not true, then $\beta_i(t)$ will be equal to 0. And $\beta_i(t)$ will be equal to 1 if $\omega_i(t)$ is the final state or absorbing state of ω_0 and $t = T$. In all other cases, this $\beta_i(t)$ will be $\sum_j \beta_j(t+1) a_{ij} b_{jkv(t+1)}$, where the summation has to be taken over all j .

So, this is the definition of $\beta_i(t)$. Now, if you look at the same diagram that we have used earlier, we can explain that part actually means. Coming over here, in case of forward algorithm, we have taken all possible transitions from the states in time step $t - 1$. In case of backward algorithm, in the other, on the other hand, what we will do is for every step in for every state in step t , I will find that what will be possible transitions to different states in step $t+1$ because all these transitions are supporting the existence of this particular or the probability of existence of this particular state. So, that is what is done in the backward algorithm and for that, this $\beta_i(t)$ has been defined.

(Refer Slide Time: 56:26)

HMM backward.

Initialize: $\beta_j(T)$; $t \leftarrow T$, a_{ij} , b_{jk} , v^T .

for $t \leftarrow t-1$

$\beta_i(t) = \sum_j \beta_j(t+1) a_{ij} b_{jk} v^j(t).$

until $t=1$

Return $P(v^T) \leftarrow \beta_i(0)$ for the known initial state.

end.

So, as we defined this, we can also write an algorithm, this backward algorithm which is, which we will term as HMM backward. So, earlier we have written HMM forward algorithm. Now, I write HMM backward. So, here again we have to initialize, we have to have an initialization state, stage for I out $\beta_j(T)$, t as T , a_{ij} , b_{jk} and V^T . So, all these are initialized. Then we have to have for loop for t to $t-1$

, $\beta_i(t)$ will be simply that expression $\sum_j \beta_j(t+1) a_{ij} b_{jk} v^j(t)$, take the summation over j . And this has to be

as we are moving backwards, so I have to move from capital T to 1. So, until $t=1$ and at the end, you return $P(V^T)$, which is $\beta_i(0)$ for the known initial state and end of the algorithm.

So, we can estimate the probability that the machine will be in state say ω_i or ω_j at a time step t after generating the first t number of sequence or it will be in the same state, at time step t and will generate the remaining symbols from the given sequence by this backward algorithm. So, I will stop here today. In the next class, I will use both this forward algorithm and the backward algorithm for estimation of the model parameters a_{ij} and b_{jk} .

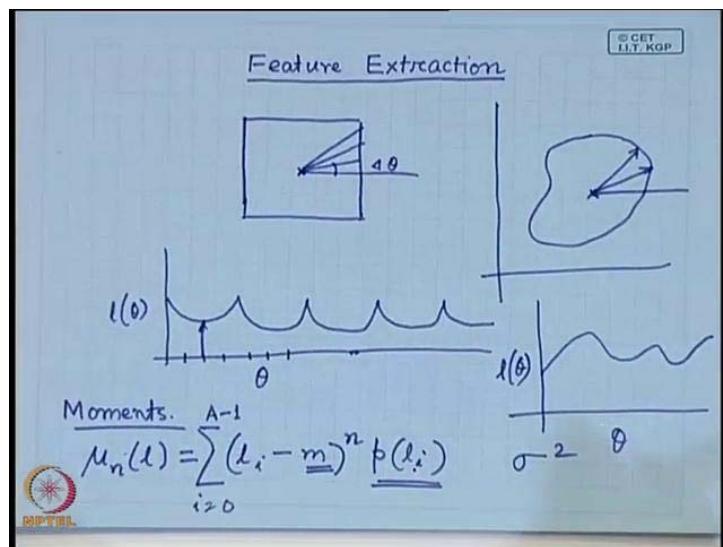
Thank you.

Pattern Recognition and Application
Prof. P.K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 4
Feature Extraction – III

Good morning, so continuing with our lecture on feature extraction, in the last class we have talked about one of the boundary feature, which is, which we have said as signature. And we have said that the signature is obtained something like this.

(Refer Slide Time: 00:38)



That given a two dimensional shape, what you have to do is you have to find out the centroid of this two dimensional shape. And if I take the horizontal line as the reference then I have to find out the distance of the boundary points from this centroid by at different orientations. So, if I take the orientation at an interval of say $\Delta\theta$ something like this. So, this is $\Delta\theta$, and then if I plot θ versus $l(\theta)$ where at every instant of an interval of $\Delta\theta$, I will get some distance value and if I have a square or a rectangular shape like this then this distance with respect to θ will vary in this form. So, it is like this.

In the same manner if I have some arbitrary shape again from the centroid of the shape you take the distance of the boundary points in different directions, where this orientation, this direction will be measured with respect to the line which is horizontal again I will get with respect to θ the distance of the boundary points from the centre which is given as $l(\theta)$. And this $l(\theta)$ will also have some variation of this form.

So, what I get is this 2 dimensional boundary information that is converted to a one dimensional function where it is a distance function distance with respect to orientation that is θ . So, if I take this as the boundary signature then I can generate different types of features or feature vectors from this particular pattern itself and one of them is the moments.

So, a moment of order say n that is given by $\mu_n(l)$ that will be nothing but

$$\mu_n(l) = \sum_{i=0}^{A-1} (l_i - m)^n p(l_i), \text{ where capital } A \text{ is the number of samples that I have within this pattern.}$$

And what I have to compute is $p(l_i)$. So, how do I get this $p(l_i)$. If this distance values that we are getting that we quantize into a number of distance values. Usually this distance value will be a continuous value.

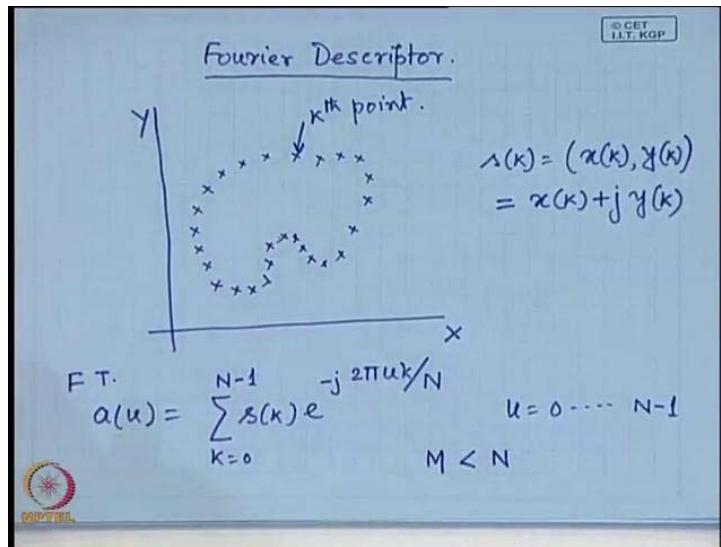
So, what I have to do it, is do is I have to quantize this distance values into a number of bins. And l_i is the i^{th} bin and once I quantize it then I have to find out that in i^{th} bin how many distance values actually occur. So, for every i^{th} bin I get the number of distance values occurring in that i^{th} bin. So, effectively what I get is histogram of this quantized distance values. And if I normalize this histogram with respect to the number of samples that I have which in this case is a then what I get is some estimate of the probability of occurrence of the l_i .

And then by using this expression I get the n^{th} moment of the distance values are and m in this case is the mean of those distance values. And obviously you find that for $n = 2$ that is the second moment that has got a spatial significance because if I put $n = 2$ this simply

$$\text{becomes } \mu_2(l) = \sum_{i=0}^{A-1} (l_i - m)^2 p(l_i), \text{ which is nothing but the statistical variance that is } \sigma^2. \text{ So, if}$$

I take the moments of different orders 2, 3, 4 and so on each of those moments becomes a feature which is a feature of this signature. So, these features can be used to represent this signature or it forms if I get a number of such features concatenated 1 after another I get a feature vector which represents this particular signature. So, this is also one of the boundary based features that can be used for recognition purpose.

(Refer Slide Time: 05:43)



Similarly, I can have other features the other one is Fourier descriptor. So, what is Fourier descriptor? Let us assume that we have a number of boundary points, say like this is in our 2 dimensions. So, I have this x dimension and y dimension. So, every kth point in this boundary say this is my kth boundary point is actually a coordinate. So, I can say that this is s_kth boundary point s_k is actually having the coordinates x_k and y_k. So, for different values of k, I get set of different boundary points. I can assume that this coordinate can be represented by a complex number. So, I can represent this as s_k = x(k) + jy(k). So, it is simply my y coordinate I am assuming to be the y axis, I am assuming to be the imaginary axis. So, this coordinate I represent by a complex number.

So, once I do this, I can find the Fourier transform of this particular sequence of boundary points. So, this Fourier transform I will have. So, if I find out the Fourier transform, I will get

$$a(u) = \sum_{k=0}^{N-1} s(k) e^{-j 2\pi u k / N} \text{ where } k \text{ varies from } 0 \text{ to } N-1. \text{ And } N \text{ is the total number of samples I}$$

have in this boundary representation. So, what I am simply doing is I have a set of 2 dimensional points which are nothing but the boundary points, these points are represented as complex numbers. So, I have a set of complex numbers or a complex sample, I take the Fourier transform of that set of complex numbers.

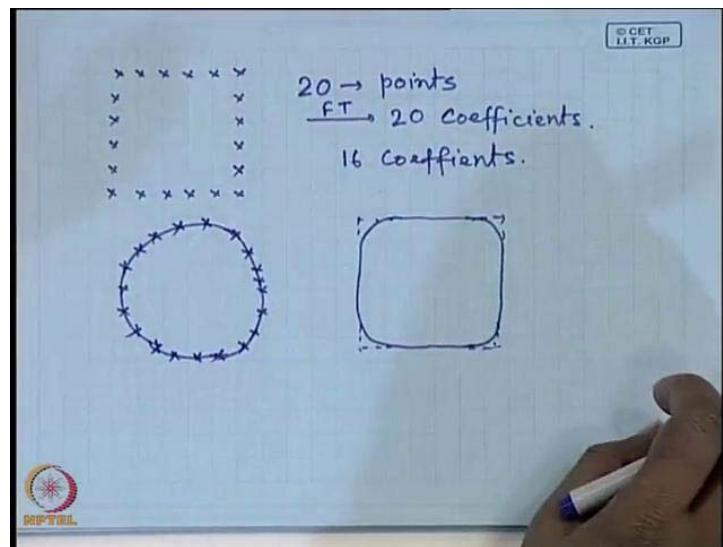
So, I get this a(u), where u will also vary from 0 to capital N-1 because if I take the Fourier transform of N number of points I get N number of coefficients. So, this u also varies from 0 to capital N-1 that means I get capital N number of coefficients. So, this capital N number of coefficients can be represented as a vector. So, I have an n dimensional vector however we know that when I take a Fourier transformation, the higher order coefficient

gives us the detailed information of the pattern. And the lower order coefficient gives us an average information of the pattern.

So, it is possible that if I am not much interested in the higher order coefficients or much detailed information within the pattern then I can truncate the vector. So, instead of considering all the n values of a u, I may decide that I will take only m values of a u where m is much less than n, so that I can reduce the dimensionality of the feature vector. So, what is the effect if I reduce the dimensionality of the feature vector is that once I have taken the Fourier transformation and got the Fourier coefficients. If I take the inverse Fourier transformation then I should get back my original pattern or the original shape, but as the higher order coefficients I am removing from my feature vector.

So, whatever the number of features that I have in my feature vector, if I take the inverse Fourier transformation of that in that case the detailed information present within my original shape they will be lost, but the total number of points will remain the same. Because when I truncate and I want to take the inverse Fourier transformation to take the inverse Fourier transformation all the coefficients which I have removed I set those coefficients values equal to 0. And take the inverse Fourier transformation, so that my total number of points remain the same. Then how does it affect when I in the shape.

(Refer Slide Time: 10:56)



So, for example, if I have boundary points something like this. See it is a square boundary and there are 20 boundary points. So, square boundary having 20 boundary points. So, when I

take the Fourier transformation, Fourier transformation will give me 20 coefficients. So, out of this 20 if I decide that some of the higher order coefficient values I will remove.

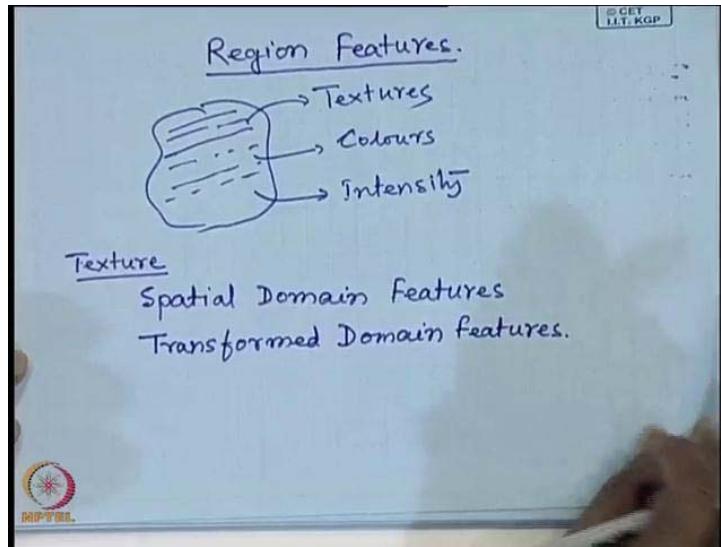
So, if for example, I decide that I will retain only 1 or 2 coefficients to represent, to give me the Fourier descriptor, and using those 1 or 2 coefficients if I take the inverse Fourier transformation by setting the coefficients which are removed equal to 0. Then the kind of boundary after reconstruction what I will get is a circular boundary something like this. Though the number of points on the circular boundary will be equal to 20. I will have 20 points on this boundary.

So, all these 20 points will be there, but you find that all the detailed information which are there at the corners those detailed information are lost. The reason is very simple that if I take only 1 coefficient that simply gives you the d_c value. So, your boundary will be an uniform boundary. Whereas, if instead of just 1 or 2 if I decide that I will retain say 15 or say 16n coefficients and I will take the inverse Fourier transformation by using those 16 coefficients, the remaining 4 coefficients I want to make equal to 0.

So, reconstructed boundary in that case will be something like this, though the total number of points on this boundary will remain the same as 20, but you find that these corner information which are not there anymore. So, over here this detailed information are lost, but overall shape is retained. So, in some cases we may feel that even this kind of description is sufficient for all recognition purpose. So, instead of taking all that 20 coefficients we may decide that I will take 16 coefficients or I will take 10 coefficients or even 8 coefficients as representation of this boundary.

And using those 10 coefficients I get feature vector of dimension 10. And this feature vector I will use for recognition purpose. So, these are the various boundary based description techniques. Now, let us go to the inside the boundary that what are the different kind of region descriptors or region based features that we can extract which can help us in the recognition process.

(Refer Slide Time: 14:44)



So, let us take some region features, so suppose I have a close boundary something like this. Now, far the descriptors that we have discussed those descriptors are based on the boundary information. That means it makes use of only the information which is given by the boundary I have not made use of any information which is their inside.

Now, there are different other type of features also which are shape features or boundary based features and there are n number of such different features. So, I am not going to discuss all of them, but just the concept that I can obtain features only from the boundaries and this simply gives you the shape information it does not tell you anything what is the inside the shape.

Now, coming to the region features this region can have different kinds of textures. So, this bounded region can have different types of textures or it can have different types of colours or different regions if it is a black and white image un-textured that is a uniform region can have different intensity values. So, this intensity value itself can be 1 feature the colour information of the region can be 1 feature as well as texture of that region can be another feature.

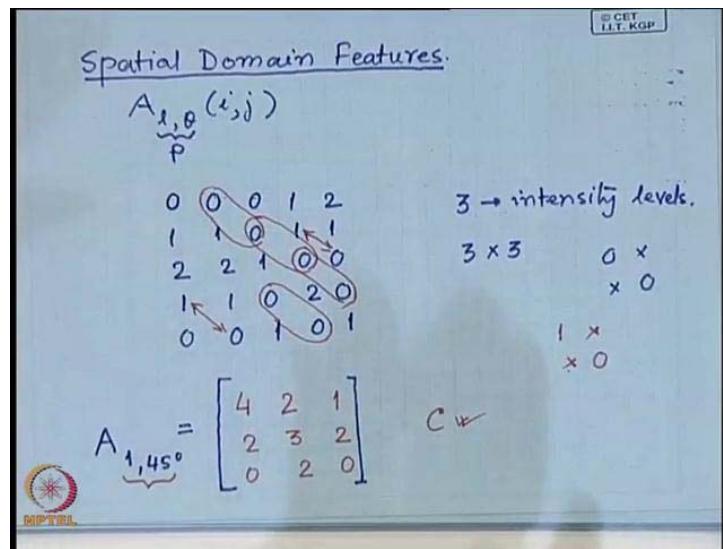
So, I can obtain the descriptors or the feature vectors corresponding to either feature corresponding to colour or even corresponding to the intensity value. So, if it is intensity value it is a scalar function. So, I get a single feature it is not a feature vector, but that feature can be combined with other features to give a feature vector.

So, let us first consider what are the different types of texture features, that I can obtain from a textured region. So, and I will talk about the other extraction techniques like colours later on, but first let us talk about the texture feature. So, when I go for region feature extraction I can have 2 types of approaches 1 is I can extract the features using the spatial domain information itself or even I can do some sort of transformation. And I can take the transform based feature

Say, like this what I have shown in this case with respect to signatures over here, the signature is a pattern. So, if I take the samples on this pattern that itself becomes a feature vector or I can take the different moments of this that becomes a feature vector the moments become a feature vector. If I take the Fourier transformation of this pattern the Fourier transformation coefficients become a feature vector.

So, either I can take the features from the spatial domain that is from the raw data or I can also extract the features after taking transformation of the raw data. So, I can extract both spatial domain features as well as transformation domain features. So, I will have either spatial domain features or I can have transformed domain features. So, let us first see that what kind of spatial domain features that we can be obtained from a texture.

(Refer Slide Time: 19:01)



So, when we talk about spatial domain features particularly in case of texture, the features are obtained from a matrix which is called co-occurrence matrix. So, co-occurrence matrix is defined like this say $A_{l,\theta}(i,j)$, where l, θ taken together defines a parameter which is called a position parameter p , l tells you what is the distance and θ tells you what is the direction?

And i and j are 2 intensity values as we have saying that it is a co-occurrence matrix it tells you that what is the frequency at which an intensity value i and an intensity value j will occur together, where the position of i is defined by this position parameter p with respect to intensity value j , is that okay?

That means I have in the image intensity value i or i^{th} intensity value which is at a distance one in the direction θ from another pixel having an intensity value j . So, I have to count that how many times this (i, j) pair following this position parameter p occur within the image. So, that count, that number gives me an element $A(i, j)$ in the matrix $A_{1,\theta}(i, j)$. So, after obtaining this matrix the number of counts in which this intensity pair i and j following this position parameter p occur within a given image.

And if I normalize that, then what I get is the frequency of occurrence or the probability of occurrence of the intensity pair (i, j) within the image following this position parameter p . So, let us take very simple example suppose I have a sample image something like this. So, suppose this is my sample image and I want to determine this particular matrix A and suppose the value of 1 is 1 that means I have to find out the pixel pairs intensity value pairs at distance 1 and at an angle of 45° . So, how will be this matrix.

Now, here you find that there are 3 different intensities levels 0 1 and 2. So, these are the 3 different intensity levels. So, naturally this matrix A will be of dimension 3 by 3 because it is indexed by intensity values i and j , i and j are nothing but the intensity values. So, the matrix A will also be of dimension 3 by 3. Now, to compute the 0eth element $A[1 \theta 0 0]$ both i and j they are 0. So, I have to find out the 0 0 pair occurring like this at a distance of 1 and inclined by an angle of 45° . So, when you come to this particular image, sample image you find that this kind of combination I have over, one over here, one over here right, one over here, is there anymore and of course this one.

So, I have 4 such occurrences where this 0 0 intensity value they appear following our position parameter 1, 45° . So, $A(0, 0)$ will have a value 4 over here, then coming to 0, 1 I have to have a pattern of this form 1, 0 I do not bother about these 2 locations because my position parameter is 1 direction is 45° and 0, 1 means the occurrence of 0 with respect to 1 following this position parameter. So, the kind of intensity pairs that I have to have is 1, 0 like this I do not bother about these 2 whatever be the value of these 2 that is immaterial to me.

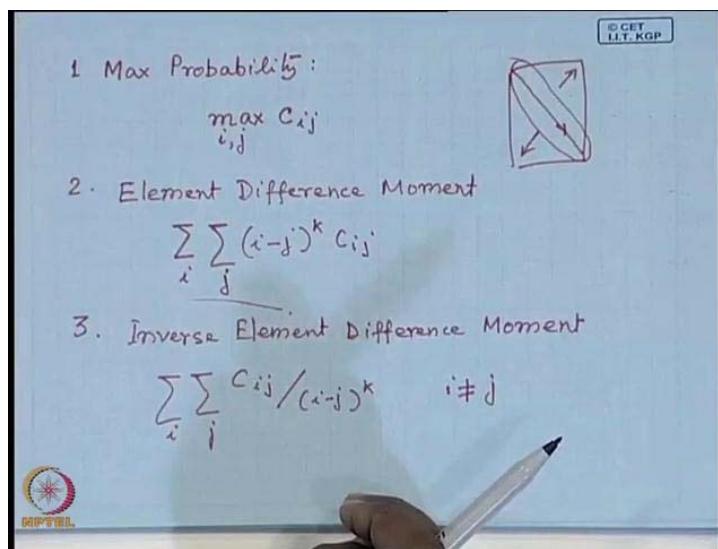
So, you find that how many such occurrences I have within this image, 1 0, 1 is over here, 1, 0 and 1 is over here. So, this value will be 2. So, if you compute like this all the pairs where

the intensity values are 1 of 0, 1. And then you get this matrix as $A_{1,45^\circ} = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 3 & 2 \\ 0 & 2 & 0 \end{bmatrix}$.

So, this is matrix A, $A_{1,0}(i, j)$ where this is the kind of occurrences of the intensity pair that I will have. Now, from this matrix I compute the co-occurrence matrix C which is nothing but normalization of this matrix $A_{1,0}(i, j)$ with respect to total number of occurrences. So, if I take the sum of all these elements in the matrix θ that gives you the total number of occurrences of the intensity pairs. So, divide this A by total, by the sum of the elements in this matrix A I get this co-occurrence matrix C.

So, this co-occurrence matrix C simply tells me, what is the frequency of occurrence of different intensity pairs following this parameter, position parameter p. So, I can have multiple numbers of such co-occurrences, co-occurrence matrixes for multiple position parameters because I can define any position parameter. And for every such position parameter I will get a co-occurrence matrix. So, find what do you get the information that you will get from this occurrence matrix because you are looking for the intensity pairs which occur within the image following some position parameter. So, it gives you various information like, what is the regularity, what is the interval at which the different intensity pairs they occur. So, various such information can be obtained from the occurrence matrix. And the kind of features which are actually obtained which are computed from the co-occurrence matrix are given by this.

(Refer Slide Time: 28:01)



One is the maximum probability. This maximum probability is nothing but $\max_{i,j} c_{ij}$, where

the maximum is taken over i and j. So, this maximum value tells you that what the pattern predominant within the texture. So, if I take different samples of the same texture then this max value will remain the same more or less same. So, this gives me an indication that how I can identify 2 patterns or 2 samples of the same pattern or 2 samples of the same texture. If the texture are different, textures are different then this $\max_{i,j} c_{ij}$ will also be different I will get a maximum, but for different values of i and j not for the same values of i and j. So, this is 1 of the features that can be obtained from the co-occurrence matrix.

The second type of feature that can be obtained from the co-occurrence matrix is element difference moment which is defined as $\sum_i \sum_j (i-j)^k c_{ij}$. So, what does this feature tell you find

that you are taking $(i-j)^k c_{ij}$.

So, naturally these value will be minimum, whenever i and j are same that is whenever $i=j$ means I have this matrix $i=j$ means the elements on the main diagonal of the matrix.

So, when I compute this the value will be very low if most of the C_{ij} or the maximums of the C_{ij} , they appear along the main diagonal. If most of C_{ij} are away from the main diagonal, for all the elements away from the main diagonal the values of i minus j is high. So, the value will be more the summation, sum value will be more whereas, if C_{ij} non most of the non-zero C_{ij} or maximum values of C_{ij} they appear along the main diagonal then this value will be more less. So, this is another feature describing the textures.

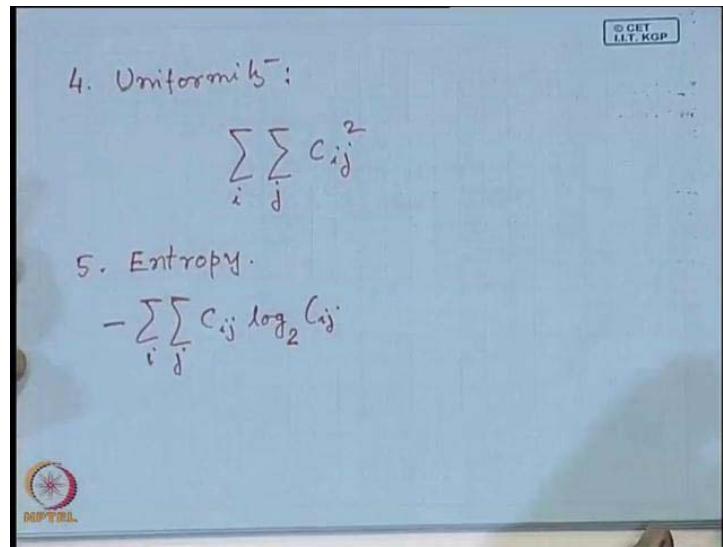
The other can be just inverse of this which is called inverse element, difference moment,

where it is defined as $\sum_i \sum_j \frac{c_{ij}}{(i-j)^k}$. And naturally this is not defined for $i=j$, because i

equal to j, $(i-j)^k$ becomes 0, so this term $\sum_i \sum_j \frac{c_{ij}}{(i-j)^k}$ becomes undefined. So, it will have

this inverse element difference moment will have just an inverse effect of element difference moment, wherever element difference moment is high inverse element difference will be low and wherever the other one is low this one will be high. So, this gives you an indication of how the co-occurrence matrix of a given texture look like. And if I have samples from the same co-occurrence matrix, from the samples of the same texture the co-occurrence matrices will also be similar similarly, the other feature.

(Refer Slide Time: 32:44)



Similarly, the other feature that can be obtained from the co-occurrence matrix is what is called uniformity. And uniformity is simply defined as summation of $\sum_i \sum_j c_{ij}^2$. And you will

find that this value will be high if all the C_{ij} are similar or same. That means all the intensity pairs they are equally likely within the given image. So, that becomes a regular pattern. And the other features which is typically obtained from this C_{ij} because C_{ij} gives you the joint probability of occurrence jointly i and j, how do they occur following the position parameter. And whenever I have a probability measure I can define something called entropy.

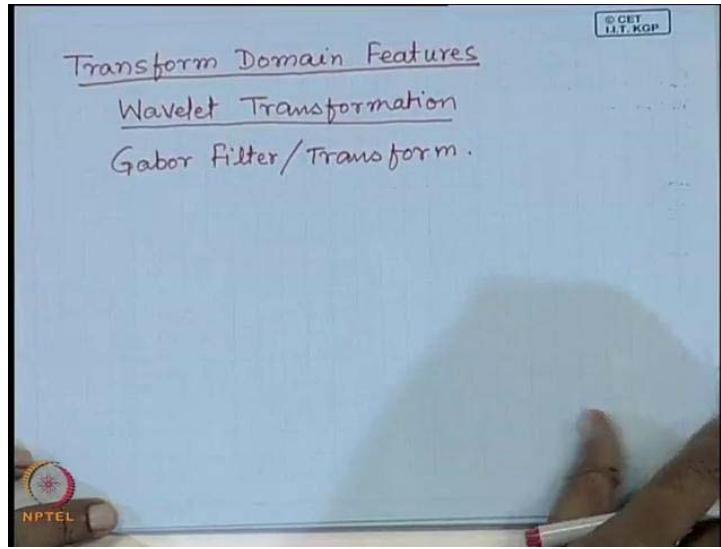
So, I can also find out what is the entropy and as you know that the definition of entropy is simply given by $-\sum_i \sum_j c_{ij} \log_2 c_{ij}$. So, if every C_{ij} value is random the entropy will be very

high that means this value will be quite high if the C_{ij} values are random whereas, if the C_{ij} values are more or less same the entropy value will be very low.

So, the entropy of a source which generates random symbols is quite high, the entropy of a source which generates the same symbols or the symbols which can be predicted from the previous symbols that is very low. So, these are different kinds of features we can obtain in the spatial domain by making use of the co-occurrence matrix of the texture. There are various other ways to generate the spatial domain features also, but I am not going into details all of them in fact there are numerous ways in which, the feature vectors can be generated.

And the way you generate the feature vector that depends on what kind of application, you have maybe that for a kind of application you will be, you may find out, that you can generate some sort of feature which is not really even in textbooks, that is also possible. So, what kind of feature you use for a particular application that depends on the application or that depends on the kind of objects that you have. Now, let us see that, what are the different kinds of transform domain features that we can obtain for textures.

(Refer Slide Time: 35:44)



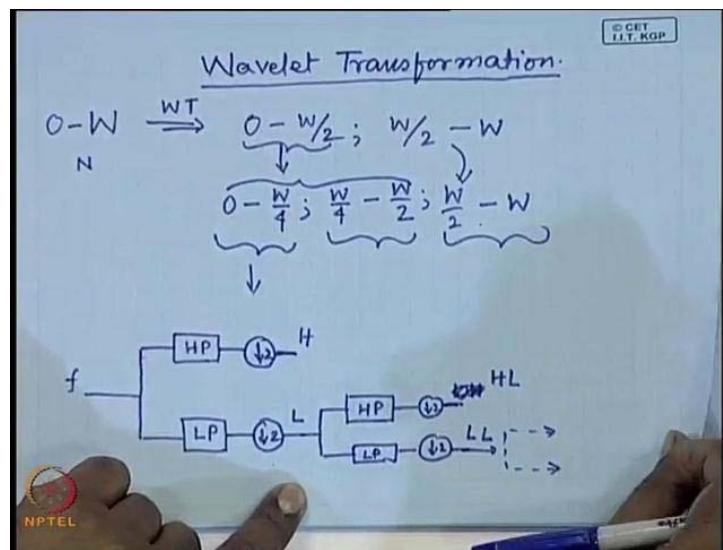
So, let me go for the transform domain features. So, whenever we talk about transformation the first kind of transformation that comes into mind is the Fourier transformation, or that cosine transformation, the difference between Fourier transformation and the cosine transformation is that Fourier transformation gives you the complex coefficients. So, you also have phase information embedded into the Fourier coefficients, the cosine transformation or the discrete cosine transformation gives you the real coefficients. So, you lose the phase information. However both Fourier transformation and cosine transformation they are very popular in signal processing.

Now, the problem with textures is that textures are some sort of random pattern, they have both high frequency components as well as low frequency components. So, if I take the Fourier transformation or the discrete coefficient or the discrete cosine transformation of the textures. The Fourier coefficients or the DCT coefficients they do not give you much of discriminating power because I have all high frequency components as well as low frequency components appearing in different degrees in different types of transformations.

So, the kind of transformation which has become very, very popular for characterizing or for describing textures are, one is wavelet transformation. And the most popular to represent textures or to describe textures is Gabor filter or Gabor transformation. So, let us first talk about the wavelet transformation. For those who have done my image

processing course wavelet transformation was not covered, I do not know whether others have done it, have you done wavelet transformation? No.

(Refer Slide Time: 38:15)



So, let us see what is wavelet transformation? When you take the Fourier transformation or cosine transformation, what is it that you do effectively? Anybody have the answer what does these coefficients mean whether it is Fourier coefficients or DCT coefficients, that is what you effectively get, but what does the transformation give you?

Whatever is the set of data you have you represent that set of data by a set of sinusoidal waves. So, if you have one dimensional signal you have one dimensional sinusoid, if you have two dimensional signal like images you have two dimensional sinusoid. For Fourier transformation I get two coefficients one is the cosine coefficient other one is the sine coefficient, the cosine coefficient gives you the real part and the sine coefficient gives you the imaginary part.

These two taken together gives you the phase information. When you take the cosine transformation I get only cosine coefficients, I do not get the sine coefficients. So, effectively I lose the phase information. So, when I take Fourier transformation or cosine transformation

as I represent the data by a set of sinusoidal waves, these sinusoidal waves are infinitely extended. So, what I have is I can equivalently say that I have wave Transformations, given a set of samples, I am representing them by a set of sinusoidal waves of different frequencies and different magnitudes, different amplitudes.

In case of wavelet transformation, it is not wave it is wavelet, that means part of wave or a small wave. So, in case of wavelet transformation the same signal is represented by tiny wave portions. Now, how, I do not want to go into details of that because wavelet transformation will itself will be full time semester course, but I will simply touch up on how you implement wavelet transformation and what you get out of wavelet transformation.

So, when we talk about the wavelet transformation it is something like this, suppose I have a signal let us assume I have one dimensional signal initially. And suppose the signal has a bandwidth of say w , right? What wavelet transformation does is, wavelet transformation breaks this signal into sub bands, the total bandwidth is w . So, let us assume that it is 0 to w when I apply wavelet transformation wavelet transformation breaks this bandwidth into two sub bands one is 0 to $\frac{w}{2}$ and other sub band is $\frac{w}{2}$ to w .

When I continue further this 0 to $\frac{w}{2}$ is further converted into 0 to $\frac{w}{4}$, half of this. And the other sub band will be $\frac{w}{4}$ to $\frac{w}{2}$ and this one is retained $\frac{w}{2}$ to w . So, I am always dividing the signal into a number of sub bands. And the same operation is repeated in the lower sub band. So, effectively what I get is a tree kind of structure.

Now over here you find that I have the original signal of bandwidth 0 to w , when I convert this into a signal of bandwidth of 0 to $\frac{w}{2}$ naturally the detailed information present in the signal is lost because high frequency components I am removing. So, this 0 to $\frac{w}{2}$ it is a coarser version or a signal, same signal at a lower resolution. Whereas, the higher sub band $\frac{w}{2}$ to w that gives you the detail information which is present in the image.

So, I can represent this kind of filtering in the form of like this. So, I have 2 filters one is a high pass filter of cut off frequency $\frac{w}{2}$ and I have a low pass filter of the same cut off frequency $\frac{w}{2}$. And this is my input signal say f , I retain the output of the high pass filter. And this output of the low pass filter that again I want to sub divide into 2 sub bands, as has

been d1 here 0 to $\frac{w}{2}$ has been divided into this sub band whereas, the higher sub band $\frac{w}{2}$ to w that has been retained.

Now, there is information here that my original bandwidth was 0 to w. And from sample rate you know, that when you have a bandwidth of w, the minimum sampling frequency has to be twice of w, that is what is your sampling, Nyquist sampling rate otherwise you lose the information.

So, if I have total n number of samples within this to represent this particular signal. Now, because I am reducing the bandwidth to $\frac{w}{2}$, so following the sampling rate my sampling frequency comes down by a factor of 2. So, I have to retain total n number of samples, but by using $\frac{N}{2}$ numbers of samples I retain the same information. So, similarly here. So, there is a concept of sub sampling by a factor of 2.

So, we find that when I have total n number of samples if I simply do the filtering low pass filtering and high pass filtering the output of this band will be n number of samples, this band will also be n number of samples. So, total from n samples I generate 2N samples, but I can reduce the size by using this concept of sub sampling by a factor of 2. So, this also reduces to $\frac{N}{2}$ numbers of samples this also reduces to $\frac{N}{2}$ numbers of samples.

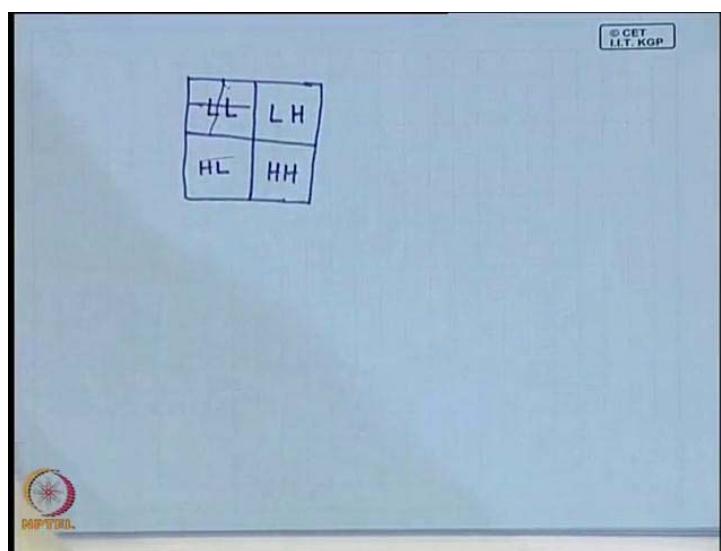
So, the total number of samples that I retain is again n. So, I go for this $\frac{N}{2}$ sub sampling and this one I called as high frequency band and this is the low frequency band. Now, I apply the same filtering operation over here high pass filter over here low pass filter over here because I am going to further subdivide the signal into different sub bands. So, this is low pass filter this will again be sub sampled by a factor of 2, this will again be sub sampled by a factor of 2. So, that way your total numbers of samples remain the same.

And this sub band now I call as LH, because this was, let me reverse this notation I call it HL because this sample was, this sub band has been obtained by first low pass filtering of the original signal. And that sub band further high pass filtered. So, first low pass filtered then high pass filtered. And this sub band I call as LL this is low pass filtered low pass filtered.

So, I can continue with this further to give me different levels of decomposition, is that okay? And this set of coefficients that I get in different sub bands this is what my wavelet

transformation coefficients are. So, this is what I have in case of one dimensional signal what do I have in case of images because image is a 2 dimensional signal. So, when I have images and that has two dimensional, that is a two dimensional signal I can do high pass filtering horizontal direction I can do low pass filtering in horizontal direction, I can perform high pass filtering in vertical direction, I can also perform high pass filtering in vertical direction. So, I actually have 4 different combinations low pass filtering horizontally low pass filtering vertically high pass filtering horizontally low pass filtering vertically low pass filtering horizontally high pass filtering vertically high pass filtering horizontally high pass filtering vertically.

(Refer Slide Time: 48:01)

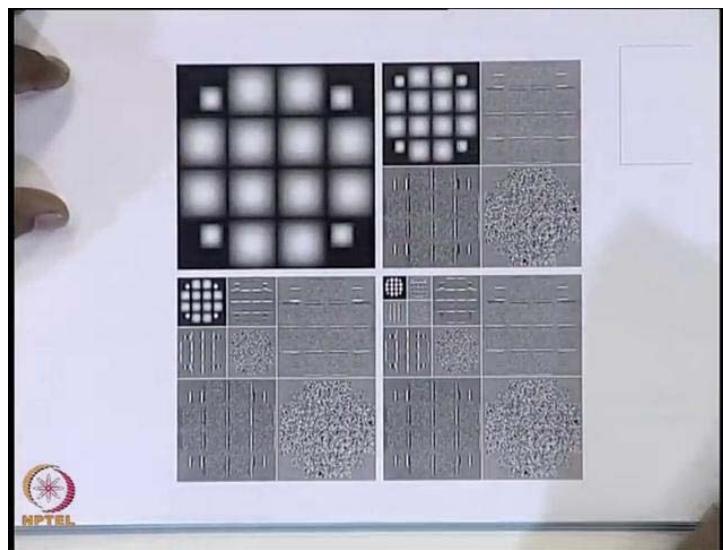


And all this different sub bands if I put in this form because every time I decompose by using sub sampling I reduce the image size by 4, half in the horizontal direction half in the vertical direction. So, over here I put the sub band which is low pass filtered horizontally, low pass filtered vertically. So, this is nothing but a reduced resolution version of the same image. Over here I can put that particular sub band which is high pass filtered horizontally, low pass filtered vertically. Here I can put the sub band which is low pass filtered horizontally, high pass filtered vertically and here I put HH band.

Now, coming to different sub bands, as I said this is nothing but lower resolution of the same image. What is this? I am doing high pass filter horizontally low pass filter vertically high pass filtered horizontally means it will try to enhance all the vertical edges, you are taking differentiation in the horizontal direction. So, all the vertical discontinuities will be highlighted. So, mainly vertical edges will be highlighted in this LH sub band.

Similarly, HL sub band will highlight all the horizontal edges, right? HH which is high pass horizontal high pass vertical that will mostly highlight all the diagonal edges, right? Then I perform further subdivision on this LL sub band go continue with every time I get sub bands at different resolutions. So, that is what wavelet transformation? There is another concept which is called wavelet packet. Wavelet packet is, I go for decomposition of each of these sub bands, in case of wavelet only the LL sub band you go on decomposing in case of wavelet transformation packet. Wavelet packet transformation, you sub divide each of the sub bands. So, I get a set of different sub bands, let me show an example of this.

(Refer Slide Time: 50:33)



Say this is an image suppose this is my original image after first level of decomposition I get this lower resolution version of the same image. Over here I have all the horizontal edges, here the combination was different high pass filtered horizontal low pass filtered vertical it does not matter whether I put that sub band here or I put that sub band here. It is simply position and this is high, high HH sub band. So, this is what I have after first level of decomposition. After second level of decomposition this is what I get, because then this LL sub band is further sub divided after third level of decomposition this is what I get. This LL sub band is further sub divided. Similarly, in case of wavelet packet transformation what I get is something like this.

(Refer Slide Time: 51:32)



This is a finger print image. So, you find that every sub band has been sub divided further. So, the size of every sub band is same after wavelet packet decomposition whereas, after wavelet decomposition as I go higher up in this tree, the sub band size becomes lower and lower every time it is 1 fourth of the sub band size of the previous level. So, effectively what I do is given a signal I divide the signal, I broken the signal into a number of sub bands.

Now, depending upon the type of the signal the energy of the coefficients in the energy sub bands will be different, all the signals will not have the same energy in the same frequencies. So, if I compute the energy of these different sub bands put them in a particular order. Then this ordered arrangement of the energies of different sub bands that itself gives me a feature vector, right, which can be used for pattern recognition purpose. Similarly, the other transformation that I was talking about is Gabor filtering. A Gabor filter is given by an expression like this.

(Refer Slide Time: 53:05)

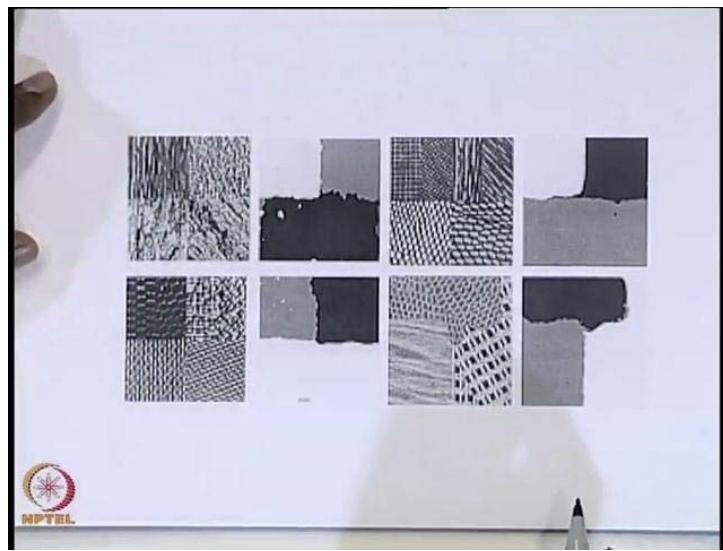
$$g = \exp\left[-\left(\frac{x'^2}{\sigma_x^2} + \frac{y'^2}{\sigma_y^2}\right)\right] \cos 2\pi w x'$$
$$x' = x \cos \theta + y \sin \theta$$
$$y' = -x \sin \theta + y \cos \theta$$

$$g = \exp \left[-\left(\frac{x'^2}{\sigma_x^2} + \frac{y'^2}{\sigma_y^2} \right) \right] \cos 2\pi \omega x'. \text{ So, this is my expression of the Gabor filter. So, here you}$$

find that this term is nothing but a Gaussian envelope in 2 dimensions centered at 0. And this is a cosine term which actually modulates this Gaussian envelope. And this x' and y' are given by $x' = x \cos \theta + y \sin \theta$ and $y' = -x \sin \theta + y \cos \theta$, where θ is the orientation of this Gaussian envelope.

So, if I convolve my image if I do filtering of my image using this Gaussian filter for different orientations for different values of ω and different values of this spread σ_x and σ_y , I get different sets of coefficients. And that different set of coefficients can give me the feature information. And this particular filter is very, very popular for texture classification as well as texture segmentation.

(Refer Slide Time: 55:14)



Here, I want to show you just some result, texture classification result that has been obtained using this sort of Gabor filter coefficients. So, here you find that this is a combination of three textures 1 texture over here, one texture here and though here it appears to be two different textures, but it is the same texture, one is the rotated version of the other. And after classification using this Gabor feature you find that this part has been put into one class this part has been put into another class, and this entire part has been put into another class. Similarly, in other examples. So, this clearly shows that Gabor filter coefficients have high discriminating power of the textures and. So, this Gabor filter coefficient can be used for texture descriptor and as well as for texture recognition purpose. So, we will stop here today.

Thank you.

Pattern Recognition and Image Understanding
Prof. P.K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 40
Hidden Markov Model
(Contd.)

Hello. So, we are discussing about the temporal pattern recognition, that is the patterns which unfolds in time. And we have started discussion on the tool, which is used for recognition or classification of such temporal patterns and this tool as we are discussing is what is known as hidden Markov model. We have seen earlier that this hidden Markov model has three central issues.

The first issue is evaluation problem where we have said that evaluation problem is nothing but if we are given a particular hidden Markov model say theta and you are given a sequence of visible symbols, then we have to find out what is the probability that the model theta has generated this sequence of visible symbols V^T by any path. So, when I say path, it is the sequence of hidden states through which the model makes transition while generating the sequence of visible symbols say V^T .

(Refer Slide Time: 01:50)

A handwritten derivation of the forward-backward algorithm for HMM evaluation. The formula shown is:

$$P(V^T | \theta) = \sum_{\gamma=1}^{\max} P(V^T | \omega_\gamma^T) \cdot P(\omega_\gamma^T)$$

Below the formula, there is a diagram illustrating the transition between hidden states ω_i and ω_j :

a_{ij} . b_{jk}
↓
 $\omega_i \rightarrow \omega_j$

At the bottom left, there are labels ω_0 and v_0 .

And we have seen that how to estimate this probability and this probability we have said that

it is $P(V^T \mid \theta)$, and we have estimated this as $\sum_{r=1}^{r_{\max}} P(V^T \mid \omega_r^T) \cdot P(\omega_r^T)$, where this r will vary

from 1 to r_{\max} , where r_{\max} is the total number of possible paths or total number of possible sequences of the hidden states through which the model can have transition while generating this sequence of visible symbols or sequence of visible states called V^T .

And we have seen that there are two other major issues. The second major issue that we have already discussed is what is called the decoding problem. We have said that the decoding problem is, the aim of the decoding problem is to find out the most probable sequence of hidden states through which the machine will make transition to generate this sequence of visible symbols V^T . And the third issue that we are now discussing, we have just started that in the next class is the most important issue, which is learning of the hidden Markov model. So, by learning what we mean is suppose we have a course structure of the hidden Markov model that is we know that what are the hidden states and we also know what are the visible states or visible symbols.

So, given these two and given a set of sequence of visible symbols supposed to be generated by this hidden Markov model, we have to find out the transition probabilities a_{ij} and b_{jk} , where a_{ij} we know that this is the transition probability from state ω_i to state ω_j , where both ω_i and ω_j they are hidden states and the transition probability or symbol emission probability b_{jk} is the probability that the machine generates a symbol v_k when it is in state ω_j .

So, these are the two transition probabilities that we have to estimate from a set of known sequences, which are generated by this hidden Markov model and we have the course description, course structure of the Markov model in terms of that hidden states and in terms of the visible states. As we have already said that out of these hidden states, one of the hidden states say ω_o , this is called the final state that is once the hidden Markov model enters this state ω_o ; it cannot come out of this state ω_o . Any further transition it will make, it has to make within state ω_o only and while the machine is in state ω_o , it will generate only one visible symbol, which is represented by v_o .

So, learning problem is to estimate a_{ij} and b_{jk} , these two transition probabilities or the transition probability matrices from a sequence of set of visible symbol sequences. And in order to do that, we have used two types of probabilities (Refer Slide Time: 06:10)

$$\alpha_j(t) = \begin{cases} 0 & t=0 \text{ and } j \neq \text{initial state} \\ 1 & t=0 \text{ and } j = \text{initial state} \\ \left[\sum \alpha_i(t-1) a_{ij} \right] b_{jk} v_k(t) & \text{otherwise} \end{cases}$$

One of the probabilities which we have estimated from using the forward algorithm that is we have defined like this that is $\alpha_j(t)$ and we have said that this $\alpha_j(t)$ actually tells the probability that the model will be in state ω_j at time step t by generating t number of visible states, the first t number of visible states given in our sequence of visible states v . The definition of $\alpha_j(t)$ is it is like this that $\alpha_j(t)$ will be equal to 0 if t equal to 0, and ω_j is not an initial state and $\alpha_j(t)$ will be will be equal to 1 if t equal to 0, and ω_j is an initial state otherwise it will be defined like this. So, this will be otherwise.

Similarly, using the backward algorithm, we have defined $\beta_i(t)$ as given by this and this $\beta_i(t)$ tells what is the probability that the model θ will be in state ω_i and will generate the remaining symbols of sequence of visible symbols V^T . That means, what is the probability that it will be in state ω_i and will generate the remainder of visible symbols starting from $t+1$ to T . So, these remaining symbols of all visible sequence V^T will be generated by this machine and under that situation, what is the probability that the machine will be in state ω_i .

So, that is what is given by $\beta_i(t)$ and we will see in a short while from now that both this forward probability and backward probability will be used for estimation of the probability,

transition probabilities a_{ij} and b_{jk} . Now, you find whether this $\beta_i(t)$, which is obtained by backward algorithm or this $\alpha_j(t)$, which is obtained by forward algorithm both of them uses a_{ij} and b_{jk} . So, $\beta_i(t)$ uses a_{ij} and b_{jk} , $\alpha_j(t)$ also uses a_{ij} and b_{jk} , but the problem is this that this a_{ij} and b_{jk} that we are going to estimate that is what is the learning of hidden Markov model. So, initially I do not know what are the proper values of a_{ij} and b_{jk} .

So, the estimates of this $\alpha_j(t)$ or $\beta_i(t)$, they are not the correct estimates because I do not know what are the exact values of a_{ij} and b_{jk} . So, these estimates are only some approximation, they are not exact as, a_{ij} and b_{jk} are not exactly known. So, what is the way out? We will do it this way that we define a probability of transition from state ω_i to state ω_j in the time step $t-1$ to time step t .

(Refer Slide Time: 10:13)

$$v_{ij}(t)$$

$$\omega_i(t-1) \xrightarrow{V^T} \omega_j(t)$$

$$v_{ij}(t) = \frac{\alpha_i(t-1) a_{ij} b_{jk} \beta_j(t)}{P(V^T | \theta)}$$

So, I define $v_{ij}(t)$ that is the probability of transition from state ω_i at time step $t-1$ to state ω_j at time step t , I define this. And this is defined for a particular tuning sequence say V^T . So, what I can do is I can define this $v_{ij}(t)$, this will be simply $\frac{\alpha_i(t-1) a_{ij} b_{jk} \beta_j(t)}{P(V^T | \theta)}$, such $\alpha_i(t-1)$ that

is the probability that the machine will be in state ω_i at times step $t-1$. It will make a transition to state ω_j . So, I have this transition probability a_{ij} it will emit a symbol v^t . So, that

will be b_{jk} , which will be effective only for the k symbol emitted at t at time step t , which is equal to v_k times $\beta_j(t)$.

So, this is the probability of transition from state ω_i to state ω_j in the time, in the time step $t - 1$ to t while generating the sequence V^T and this divided by $P(V^T|\theta)$. So, this will be my probability of transition from ω_i to ω_j in the time instant t and you find that this $P(V^T|\theta)$, this is the probability that the model θ has generated this sequence V^T following any path. And that is quite obvious because when we have defined $P(V^T|\theta)$, definition of $P(V^T|\theta)$ was something like $\sum_{r=1}^{r_{\max}} P(V^T|\omega_r)P(\omega_r^T)$.

So, it is summed over all possible paths ω^T , ω_r is one of the possible sequences of length capital T . And I am summing over all such possible sequence of hidden states. So, this $P(V^T|\theta)$ is actually the probability that the model θ has generated this V^T following any path. This includes only that path where a transition from ω_i to ω_j from time step $t - 1$ to time step t is involved.

So, this is an estimate of the probability of transition in the t^{th} step from ω_i to ω_j at time step t . Now, here again you find that this a_{ij} and b_{jk} , they are still not exactly known. So, how this algorithm is going to work? So, the usual practice is initially you choose the transition probabilities a_{ij} and b_{jk} are arbitrarily at random.

So, I choose some random values, some arbitrary values for the transition probabilities a_{ij} and the transition probability b_{jk} . Using that and using training sequence V^T , you try to estimate what is $\gamma_{ij}(t)$. Now, you can use this estimated $\gamma_{ij}(t)$ to refine or to improve the values of a_{ij} or the values of b_{jk} . So, how that improvement can be done? You find that the expected number of transitions from ω_i at $t - 1$ to ω_j at t at any time in the sequence is given by.

(Refer Slide Time: 14:56)

Expected no. of transitions
 $\omega_i(t-1) \rightarrow \omega_j(t)$
at any time in sequence V^T

$$\sum_{t=1}^T \gamma_{ij}(t)$$

Total expected no. of transitions from ω_i :

$$\sum_k \sum \gamma_{ik}(t)$$

So, I will write that expected number of transitions ω_i in time step $t-1$ to ω_j at time step t at any time in the sequence, in sequence V^T , it will be simply sum of $\sum_{t=1}^T \gamma_{ij}(t)$, where this summation has to be taken over $t = 1$ to capital T . So, in this entire sequence V^T wherever, whenever I have a transition from state ω_i to state ω_j , I have to sum all of them.

So, that is what is giving me the expected number of transitions from ω_i to ω_j . And the total expected number of transitions from state ω_i , so against this, I have total expected number of transitions from state ω_i that will be given by $\sum_k \sum \gamma_{ik}(t)$ where I have to take summation over the index k and this whole this thing has to be summed again over $t = 1$ to capital T . So, you find that I have two quantities. I have the total expected number of transitions from state ω_i to state ω_j given this sequence V^T and I also have the total number of expected transitions from state ω_i to any state. So, once I have these two quantities, then I can have the transition probability a_{ij} .

(Refer Slide Time: 17:40)

© CET
I.I.T. KGP

$$\hat{a}_{ij} = \frac{\sum_{t=1}^T \gamma_{ij}(t)}{\sum_{t=1}^T \sum_k \gamma_{ik}(t)}$$

Forward-
Backward
Algorithm/
Baum-Welch
Algorithm.

$$\hat{b}_{jk} = \frac{\sum_{t=1}^T \sum_l \gamma_{jl}(t) \quad v(t) = v_k}{\sum_{t=1}^T \sum_l \gamma_{jl}(t)}$$

So, I can have a refined transition probability a_{ij} . I will put it as \hat{a}_{ij} , which will be sum of

$\frac{\sum_{t=1}^T \gamma_{ij}(t)}{\sum_{t=1}^T \sum_k \gamma_{ik}(t)}$. So, this is our refined value of transition probability a_{ij} and in the same

manner, I can have a refined value of the transition probability b_{jk} . So, I will have \hat{b}_{jk} , which

will be given by $\frac{\sum_{t=1}^T \sum_l \gamma_{jl}(t) \quad v(t) = v_k}{\sum_{t=1}^T \sum_l \gamma_{jl}(t)}$. And I have to consider only those cases, where the emitted

visible symbol is v_k .

So, this has to be computed wherever $v(t) = v_k$ because I am interested in the transition probability b_{jk} and this denominator is irrespective of any symbol that is emitted from the state ω_j . So, the numerator is only considering those transitions wherever the emitted symbol is symbol v_k and denominator is irrespective of whether the emitted symbol is v_k or not. So, this ratio gives me a refined estimate of b_{jk} . So, the procedure I have is initially you choose the arbitrary values of a_{ij} and b_{jk} . Using those arbitrary values of a_{ij} and b_{jk} , you estimate what will be your $\alpha_i(t)$ and what will be $\beta_j(t)$.

So, using this arbitrary estimate, the initial estimate of a_{ij} and b_{jk} , you have an estimate of $\alpha_j(t)$ and an estimate of $\beta_i(t)$. Using this estimate of $\alpha_j(t)$ and $\beta_i(t)$ what you go for, you go for an estimation of the transition probability from ω_i to ω_j and that transition probability is this.

So, you find that I have $\alpha_i(t-1)$ or $\alpha_j(t-1)$, whatever subscript I use that does not matter. Similarly, I have an estimate of $\beta_j(t)$; I have an arbitrary estimate, initial estimate of a_{ij} and b_{jk} . So, I can estimate again using this a_{ij} and b_{jk} , I estimate what is $P(V^T / \theta)$ and from here, I can estimate what is $\gamma_{ij}(t)$.

Using this $\gamma_{ij}(t)$, I go for refinement of the transition probabilities a_{ij} and b_{jk} . In the second iteration, I use this refined values of a_{ij} and b_{jk} again to estimate what is $\alpha_j(t-1)$, what is our $\beta_i(t-1)$ or $\beta_j(t)$. I go for estimate of what is gamma i j. Again, using that refined value of gamma i j, I go for further refinement of a_{ij} and b_{jk} .

This entire step starting from refinement of gamma i j, refinement of b_{jk} and refine, refinement of a_{ij} , this entire process repeats a number of times unless I reach to a situation when the difference in a_{ij} or the change in a_{ij} and the change in b_{jk} , they are within the tolerance limit. That is when I assume that my algorithm has converged.

So, when I reach that state, that a_{ij} and that b_{jk} can be used for evaluation of my model θ for any sequence VT. And that is how hidden Markov model is trained and because this process is using both forward estimation and the backward estimation, the algorithm is called forward backward algorithm. Also, this is also known as Baum-Welch algorithm.

So, let us try to recapitulate what we have done during this entire course on pattern recognition and applications. Initially, we said that whenever we want to classify or we want to recognize a pattern, for the pattern, we have to have some descriptors or some features and when we take a large number of features put in a particular order that is what is called as feature factor.

So, when a pattern is represented by feature vector, suppose the feature vector is of dimension d, then this entire pattern is represented by a point or by a vector in a d

dimensional space. The advantage that I have by representing a pattern by a feature vector is as the pattern is represented by a point in my d dimensional space, if I have two patterns p 1 and p 2, the pattern p 1 will be represented by a point in d dimensional space and pattern p 2 will also be represented by a point in my d dimensional space.

And now, what I can do is I can find out the distance between these two points. If I find that the distance is very large, immediately I can infer that these two patterns p1 and p2, they are not similar, they are widely different. Whereas if the distance between the corresponding points is in my d dimensional space is small, ideally if the distance is 0, that is both the points are mapped the same points in my d dimensional space, then immediately I can infer that the patterns p1 and p2, they are same.

If the distance is small, I can infer they may not be same, they may not be identical, but they are very much similar because their corresponding feature vectors are similar. They are very close to each other. So, this is the advantage I get whenever I represent a pattern by feature vector and when I say feature vector having d number of components, each of these d number of components capture certain aspects of the pattern. And these aspects or these properties of the patterns can be estimated in the time domain or in the spatial domain; in the time domain if it is the time domain signal, in the spatial domain if it is the special domain signal.

So, I say it is time domain say for example a speech signal; a speech signal varies with time right or the signal coming out of a microphone that is also a signal, which varies with time. So, whenever I have a time domain signal, I can extract the features in time domain itself. I say spatial domain signal say for example in case of an image; the image is defined over a two dimensional space. I can have combined spatial domain and temporal domain signal like in case of a video sequence. In a video sequence, as you know that the video sequence is nothing but a number of frames which are played one after another at certain intervals. So, usually when we have commercial videos, they are played at an interval of say 25 frames per second, at a rate of 25 frames per second and or at the rate of 30 frames per second.

So, every individual frame is a spatial domain signal and when you take the number of frames, which are played one after another, we move into temporal domain. So, I can have signals, which will have both spatial property as well as temporal property. So, when I talk about features, I can extract the features in spatial domain, I can also extract the features in the temporal domain. So, whichever way I extract the feature or I can also extract the features in the transformed domain.

So, this signal can be transformed to the transform domain using the various types of transformations. I can use Fourier transformations, I can use discrete cosine transformation, I can use Wavelet transformations, I can use Gabor transformations, and many such transformations. So, by using these transformations, what we do is we simply transform the signal into the transformed domain and the features can be extracted in transformed domain also.

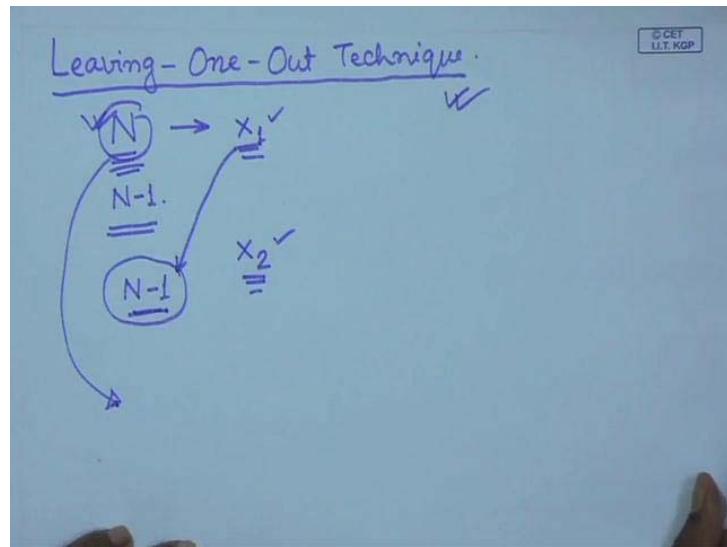
So, I can have spatial domain signal, I can have spatial domain features, I can have time domain features, I can also have transformed domain features. Each of these features as I said captures certain aspect or certain property of the pattern or certain property of the signal, when these features are put in a particular order, what I get is a feature vector and the advantage of converting a pattern or a single or a signal to a feature vector is as I said that it can be represented by a point in my feature space or in my d dimensional feature space. And we have talked about various types of classifiers starting with our statistical classifier like Bayes classifier; we have talked about the parametric classifier where the probability density function is having a parametric form.

We also have talked about the non-parametric classifier when we still estimate the probability density function at a particular instance of the feature vectors, but we do not assume that the probability density function has any parametric form. We have also talked about other types of non-parametric classifiers like nearest neighbor classifier, minimum distance classifier, and all these different sorts of classifiers. Then among the other types of classifiers, we have talked about using the discriminating functions for different classes. We have also talked about what are the decision boundaries with different classes.

So, using that, we can classify the patterns or we can recognize the patterns. We have also talked about the neural networks for pattern classification or pattern recognition. We have talked about hyper box classifier. We have talked about the fuzzy min max neural network for pattern classification. So, these are different types of pattern recognition techniques or pattern classification techniques that we have done throughout, we have discussed throughout this course.

Now, the question is as we have so many different types of classifiers or as we have so many types of recognizers, how to find out or how to estimate the performance of different types of classifiers?

(Refer Slide Time: 31:47)



Before that, there is another problem that how we design a robust classifier, so for the design of classifier, there is one of the techniques we will talk about is what is called leaving one out. So, what is this leaving one out technique? It is quite intuitive that when we train the classifier using the supervised techniques that is when you design the classifier using a large number of feature vectors for which the class belongingness is known, then it is quite intuitive that if I increase the number of such samples used for the training of the classifier, the classifier will be more robust.

Similarly, for testing, if I use a large number of samples for testing the classifier, the tested result will be more accurate. But, practically, it is difficult to obtain such large number of training samples or such large number of test samples because only after proper training and only after getting the test results, I can put the classifier into the actual job. So, the leaving one out technique is basically a virtual technique by which even limited number of samples, training samples or test samples can be posed as a large number of samples. So, suppose you had been given capital N number of training samples.

So, what this leaving one out technique does is you set aside one of the training sample for testing purpose. So, if I set aside one of the training samples, I am left with N minus 1 number of training samples. You design your classifier using N-1 number of training samples and the sample say x, which you have left aside, you use this sample x for testing the classifier, which has been designed using N-1 number of training samples.

So using, suppose this is x_1 , which I had left aside and using all the samples other than x_1 , I train my classifier and use x_1 to test the classifier. Next, I set aside another sample say x_2 and use this x_1 for training. If I set aside this training sample x_2 , I am again left with N minus 1 number of samples for training, where this N minus 1 number of samples include this x_1 and excludes x_2 . So, using this $N-1$ number of samples, again you design the classifier and using x_2 , you test the classifier.

So, if every time I set aside one of the samples out of total number of samples N , and every time I design the classifier using the remaining N minus 1 number of samples and the sample which was set aside, I use that for testing purposes. So, effectively what I am doing is, I am designing N number of classifiers. When this x_1 or this x_2 is used for testing the classifier, which has been designed using the other N minus 1 number of samples, sometimes this test will result positive.

That means, the sample may be correctly classified, sometimes the test result may be negative that is the sample may not be correctly classified. So, what I can do is, I can estimate a classification error rate that is out of so many test samples, how many times I have got misclassification of the sample? So, I have a classification error rate.

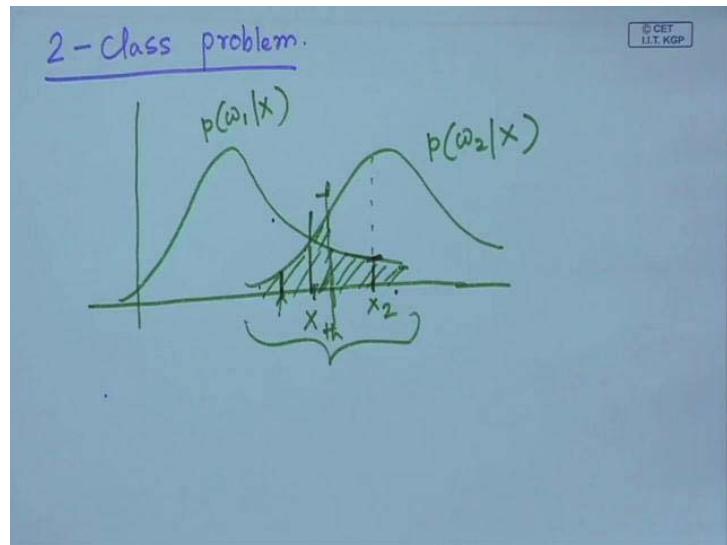
Then for designing your final classifier, you use all the N number of samples to design your final classifier and when all these N number of samples are used to design the final classifier, I can say that the error rate of the classifier will be at least less than the error rate that has obtained when I had designed N number of classifiers leaving aside one, so because I am using more number of samples to design my classifier. So, when I design the classifier, I know that what is the expected error rate? This is the method of designing the classifiers, which is called leaving one out technique.

Now, this can be slightly relaxed in the sense instead of every time leaving one sample, I can say set aside a 10 percent of the samples and I design the classifier using the remaining 90 percent of the samples. This classifier will be tested using the 10 percent samples, which was set aside. So, every time I set aside 10 percent, using the remaining 90 percent, I design the classifier and test this classifier with this 10 percent samples.

So, here you find as I am setting aside 10 percent of the samples, I will be designing 10 different classifiers. For every classifier, I find out what is the error rate and then when I finally, design the classifier using all the N number of samples, all the samples, I know that the maximum error rate of this classifier will be the error rate of the individual sample that I

have got. So, this is one way of designing technique. Now, suppose I have designed my classifier. Then how do I present the performance of this classifier or how do I estimate that what is the sensitivity of this classifier? Now, when I say sensitivity, typically I am talking about a 2 class problem.

(Refer Slide Time: 38:43)

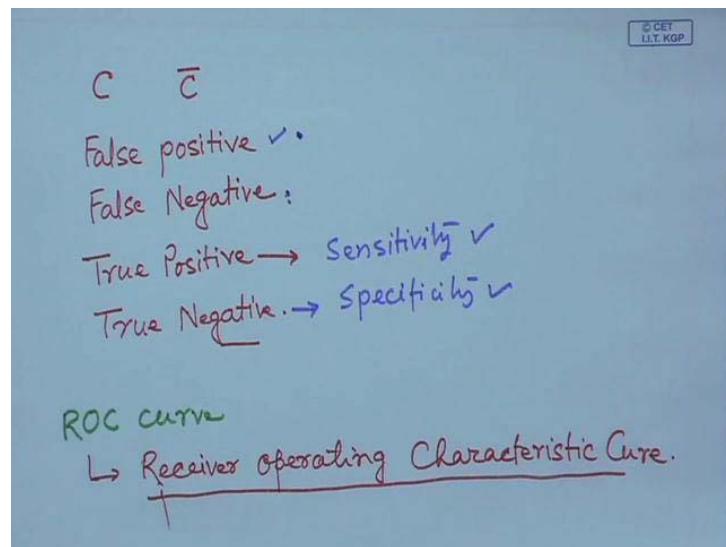


So, to discuss about this 2 class problem, I will take our old methods, which we have used earlier that I have say these two probabilities densities. So, this is $P(\omega_1/X)$ and this is $P(\omega_2/X)$ and for a minimum error rate classifier, we know that our threshold, our decision bound was somewhere over here. So, this is X_{th} . So, in case of minimum error rate classifier, we had assumed that if the value of X , if the vector is to the left of X_{th} , then the feature vector is classified to class ω_1 . If it is to the right of X_{th} , then the feature vector is classified to class ω_2 .

So, this is my decision boundary and while doing this because there is a finite probability, that X over here may belong to X_2 . So, I have an error, which is given by this amount. Similarly, when I decide feature vector say X_2 falls on this side because $P(\omega_2/X_2) > P(\omega_1/X_2)$, I decide that this feature vector belongs to class ω_1 , but there is a finite probability given by this that the feature vector may also belong to class ω_1 . I have decided that it belongs to class ω_2 , but there is a finite probability that the feature vector may belong to class ω_1 .

So, the probability of error over here is this much, the probability error over here is this much and the total probability is nothing but the area under these two curves. Now, if I shift my decision boundary to say this side, then for one of the classes, the probability of error will increase; for the other class, the probability of error will decrease. So, if I shift my decision boundary towards class ω_2 , then the probability of error for class ω_2 will increase; probability of error, or total error for class ω_1 will decrease. So, for all such decision processes, I have one type of probability error, one type of error rate which increases and the other type of probability which decreases with the shifting of my decision boundary.

(Refer Slide Time: 41:58)



So, a particular case of this supposes we consider a 2 class problem where I am interested in samples belonging to a class c and the samples, which do not belong to class C . So, I represent them as C and \bar{C} . So, this represents the samples belonging to class c and this represents the samples which do not belong to class C . Now, because of our decision making process or because of the classification process, I may have a situation that a sample, which actually belongs to class \bar{C} is classified as belonging to class c . So, this is a kind of situation which is expressed as false positive. I can also have a situation that the sample actually belongs to class C , but it has been classified to be belonging to class c complement. So, from c , you have put it to \bar{C} . That is what the classifier has done and such cases are known as false negative.

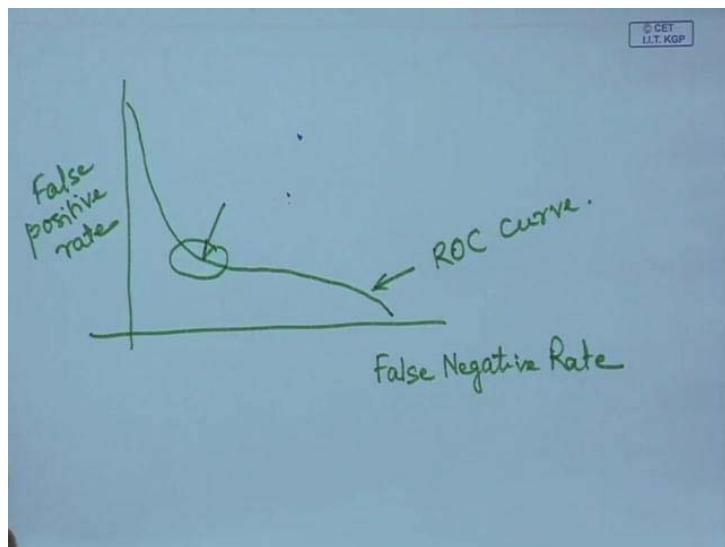
There are correct classification cases when a sample actually belongs to class C and it has been classified to belong to class C , which is called true positive and a case when a sample

actually belongs to class \bar{C} and it has been classified to belong to class c complement and that is the case, which is known as true negative. So, I have true positive case and I have true negative case. So, these are the different cases that I can have. This true positive or rate of true positive is also called the sensitivity of the classifier and the true negative this is what is called specificity of the classifier.

So, I have these two terms, one is sensitivity, other one is specificity. As we said that by varying the position of the decision boundary, I will increase one error rate. At the same time, the other type of error rate will decrease that is the number of false positives, the number of false negatives, in one case the false positive can go on increasing, the false negative go on decreasing. In the other side, the false negative may decrease, but false negative may increase, but false positive will decrease.

So, naturally the position of the boundary leads to a trade-off and the situation of best trade-off is actually described by a curve, which is called a characteristic curve or which is also known as ROC curve. This ROC curve, this ROC actually represents receiver operating characteristic curve. So how do you plot this ROC curve? ROC curve is you plot one of the false cases against the other that is I can plot false negative versus false positive.

(Refer Slide Time: 46:23)



So, a curve something like this false negative rate versus false positive rate for different positions of my decision boundary. This curve may be say something like this and this is called an ROC curve or characteristic curve. The advantage of this ROC curve is, it simply says that how sensitive your classifier is against shifting of the decision boundary. And from

this ROC curve, I can choose the best trade off, may be something somewhere over here I can say that total number of false positive and false negative that will be minimum. So, whatever, whichever position of the boundary I am at this position, that position in can use as my decision boundary position.

(Refer Slide Time: 47:56)

Confusion Matrix.

	A	B	C
A	w	x	x
B	x	w	x
C	x	x	w

The other way of representing the performance of a classifier is, by means of what you call a confusion matrix. The confusion matrix says that suppose I have got three different classes, class A, class B, class C and the confusion matrix is plotted in this manner A B C. So, it simply says that how many samples belonging to class A has been classified as class B, how many samples belonging to class A has been classified as class C. So, these are actually wrong classifications or false classification. The sample actually belongs to class A, but it has been classified as class B or class C. So, these are false classes and it also says that how many samples belonging to A has actually been classified as class A.

So, these are the false classifications. This is the true classification. Similarly, how many samples belonging to class B has been falsely classified to class A, how many samples have been correctly classified as class B, how many samples have been falsely classified as class C. Similarly, for class C, how many samples belonging to class C has been classified to A, falsely classified to B, how many samples have been correctly classified as class C. So, you find that the diagonal of this confusion matrix, this actually gives you that how good your classifier is because this is what represents the number of correct classifications. So, these are

the defined techniques. There are many other techniques, which can be used for, to measure the performance of the classifier.

So, in this course, we have talked about different classifier techniques, classification techniques or recognition techniques. We have also talked about different types of clustering, where similar patterns or the similar feature vectors are put into the same group. We have talked about the classification or recognition of temporal patterns and towards the end, we have concluded our course on pattern recognition, and applications using the design techniques like leaving one out and the different types of performance measures of the classifiers. So, with this, we come to end of this course. I hope you have all enjoyed this course and you got benefitted by this course.

Thank you.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 5
Bayes Decision Theory

Good morning. So, today we will start our discussion on pattern recognition problems. And today particularly will talk about Bayes decision theory which is the basis of statistical pattern recognition. Now, before I come to the pattern recognition problem let us just have a quick recapitulation of what we have done over last three or four classes. So, our last few classes what we have said is given an object we can find out different types of features of that object.

The features may be derived from the boundary features which are shaped descriptors or shape features. Similarly, the features may also be derived from the region enclosed by the boundary and those features can be say texture features, or they can be color features, they can be intensity features and so on. We have also said that none of this feature on its own can describe an object or a shape.

Rather many of the features taken together they can describe an object to some degree of accuracy. So, instead of considering a single feature we have to consider a feature vector, where the components of this feature vector will be from different features may be from boundary features may be from shape features may be from region features, like colour texture and all that. They had to be concatenated in a particular order and whichever order we concatenate them throughout our problem that is modelling of the pattern as well as recognition of the pattern we have to make use of the same order.

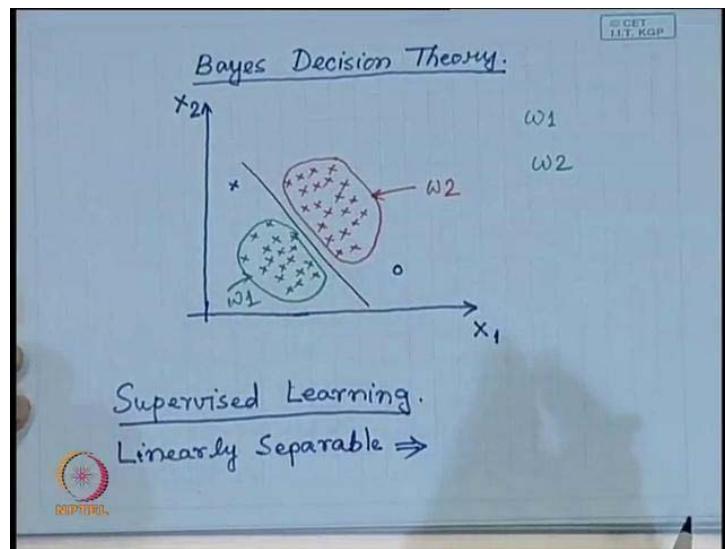
So, when I put all these different features in a particular order what I get a feature vector. So, the dimensionality of the feature vector will be dependent upon the kind of problem that we have. In some case, may be just two dimension sufficient in some cases three dimension, in some cases the dimensionality of the feature vector can be more than 100 maybe even 500 or so.

So, as more and more dimension or more and more features, you add to the feature vector the description becomes more or more unique. There is to a great extent the

accuracy of the description increases as we increase the dimensionality of the feature vector. So, to what dimension of the feature vector we have to go for in order to tackle a pattern recognition problem that depends upon what is the problem that you have attacked. What is the complexity of the pattern that you have?

Now, whatever it is, once we describe or once we represent a pattern by a pattern vector by a feature vector then this entire pattern is mapped to a single point in our feature space. So, if I have a two-dimensional feature then a pattern will be mapped to a point in the two dimensional feature spaces, if we have three dimensional feature vectors for a pattern and the pattern is mapped to a point in the three-dimensional features space. If we have n dimensional feature vector then the same pattern will be mapped to a point in our n dimensional feature space. So, it is nothing but whenever we are finding out the feature vector we are basically mapping that pattern to a point in the feature space. So taking a very simple example suppose.

(Refer Slide Time: 04:01)



We have two-dimensional feature vectors. I am taking this example of two-dimension illustration rather in two dimensions, because I can very easily plot or demonstrate what happens in two dimensions. As the dimensionality increases the complexity also increases. So, I may not be able to plot them on a two-dimensional space. So, that is the

only reason I am illustrating this with the help of two dimensional feature space. So, suppose my two-dimensional feature vector has got two components one is X_1 and other one is X_2 . So, these are the two components of my feature vector. Now different patterns will be mapped to different points in this two-dimensional feature space.

So, suppose I have got patterns belonging to two different categories. One of the categories I may call as ω_1 that is one category and the other category maybe ω_2 , now because I am taking patterns from these two categories ω_1 and ω_2 . So, all the patterns which are taken from class ω_1 , they will be put in, they will be mapped in the number of points in this two-dimensional features space while the points will be very close to each other.

Similarly, the patterns which are taken from this particular class ω_2 , those will also be placed in points, mapped to points in this two-dimensional features space while these two points will also be very close to each other. Whereas, the two points corresponding to the patterns from class ω_1 and the points corresponding to the patterns from class ω_2 , they are likely to be wide apart. So, effectively what I will have for all the patterns which are taken from class ω_1 , they will form a point cloud in this two-dimensional space.

Similarly, all the patterns which are taken from class ω_2 they will also be mapped to point clouds in this two dimensional feature space, where these two different clouds are likely to be wide apart if the patterns are wide apart. So, I will assume that all the points corresponding to the patterns in class ω_1 , maybe they will be mapped to points like this. Similarly, all the patterns belonging to class ω_2 they may also be mapped to points or point clouds something like this. So, this is the set of points which corresponding to feature vectors taken from class ω_1 and this is the set of points representing feature vectors corresponding to the patterns taken from class ω_2 .

Now, in this simple kind of situation this pattern recognition problem is nothing but something like this that I have to find out a decision boundary. So, if the points lies on one side of the boundary I will say that point belong to one class if it lies on the other side of the boundary I will say that the pattern belongs to some other class. So, in this simple example you find that I can draw a straight line separating these two different sets of point clouds.

So, if I have an unknown pattern or a feature vector corresponding to an unknown pattern and suppose the feature vector falls on this point, somewhere over here. Now, the

since the feature vector is on the left side of this decision boundary, I will say that this feature vector belong to class ω_1 . If the feature vector happens to be somewhere over here, it is on the other side of the decision boundary, I will say the feature vector belongs to class ω_2 .

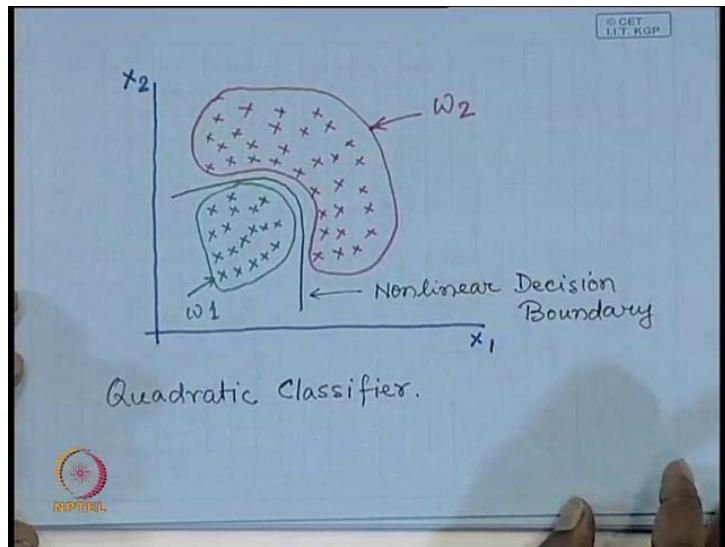
So, here designing of this classifier or training on the classifier means, I have to find out the equation of this decision boundary from the set of training samples that are given to us. So, what are those training samples, this is the set of training samples taken from class ω_1 and this is the set of training samples marked in red taken from another class that is ω_2 .

So both, for these two sets of training samples I already know what the last belongingness is? So, that is why this is called supervised learning, that means for designing this classifier or for training of this classifier I take a set of training samples for which the class belongingness is known. I take a number of training samples from class ω_1 , I also take a number of samples from class ω_2 and making use of these two training samples I have to find out this linear decision boundary.

And because in this case the two classes can be separated by a linear boundary this is also known as linearly separable classes. So, classes in this case are linearly separable. So, only when the classes are linearly separable I can find out a straight line separating the two classes in two dimensions. If it is in three dimensions that is I have three dimensional feature vectors in that case, I will have a plane separate separating the two classes. If the dimensional is more than three, I can neither have a line, a straight line nor a plane, but what I have is a hyperplane.

So, a hyperplane of dimension 4, hyperplane of dimension 5, in case of five dimensional vector feature hyperplane of dimension 6 in case of six dimensional feature vector hyperplane of dimension n in case of n dimensional feature vector, but in all these cases the equation of the decision boundary that will be linear. So, this is a very simple case why the classes are linearly separable. If the classes are not linearly separable in that case what we do?

(Refer Slide Time: 11:32)

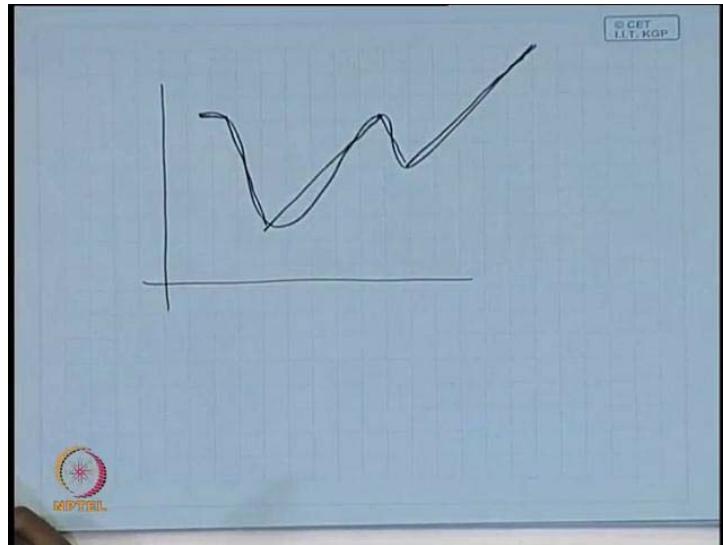


So, let us take another illustration say something like this again I have two-dimensional feature vectors having components X_1 and X_2 . I have, say these are the samples which belong to class ω_1 , that is one of the two classes. So still I have, I am taking the concept of supervised learning that means for samples I already know to which class this sample belongs to.

Similarly, I have another set of training samples taken from class ω_2 which are distributed like this. So I am putting these samples in red and so you find that I have this set of training samples which are taken from class ω_2 , so when the samples belonging to class ω_1 and the samples belonging to class ω_2 they are distributed like this. Now, you find that I cannot separate these two classes by a straight line. Similarly, in three dimensions, if the samples are distributed like this, I cannot separate the classes by plane or in higher dimension I cannot separate the classes by hyperplane.

So, in such cases I have to have a non-linear decision something like this. So, I have to have non-linear decision boundary and we say that the classes are not linearly separated. So, among these non-linear decision boundaries the most popular one and the most common one is a quadratic classifier, or at the most we can go for a cubic classifier. Classifiers of higher order more than quadratic or cubic are not very common because designing such classifier is not of that simple. Now, if I have a decision boundary as simple as this possibly I can separate the boundaries by quadratic classifier or using cubic classifier, but if the decision boundary is much more complicated than this.

(Refer Slide Time: 15:09)

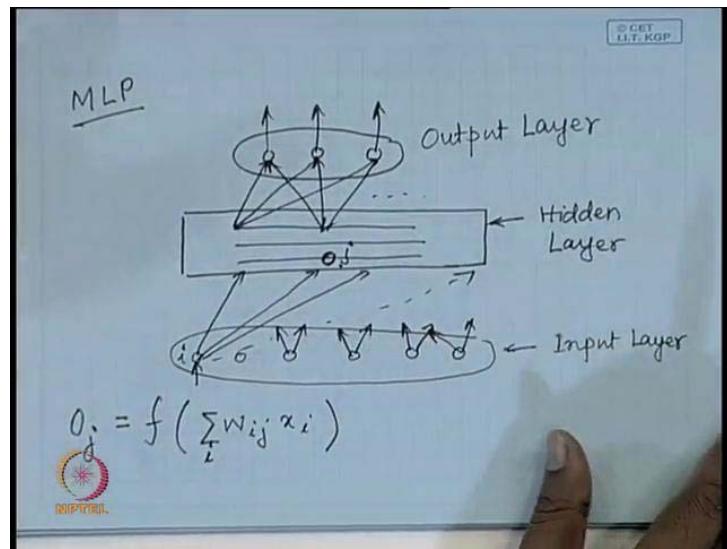


Suppose, I have decision boundary like this, something like this, so you find that such boundaries decision boundary are so complicated. It is not very easy to design such decision boundaries analytically, or to have an analytical expression for such a decision boundary. So, this is the case with, when I am considering only two class problem that is I have only two classes that is ω_1 and ω_2 . The decision boundary becomes much more complicated if the number of classes are more than two. So, I can have 3 classes, I can have 4 classes, 5 classes, I can have even 10, 15 or 20 number of classes. So, as the number of classes goes on increasing and the boundaries are non-linear having analytical expression for such decision boundary is almost impossible.

So, the kind of approach that people take in such complicated cases is making use of neuron network classifiers. Where the decision boundary, the information of the decision boundary is actually encoded in the weight vectors, or the weight matrices of a neural network. How many of you have done neural network classifier only 1, 2, 3, 4, 5, 6, 7. So, let us see, I mean we will come to neural network classifiers later on, but what does neural network do.

So, this sort of decision boundary though it is a complicated non-linear decision boundary, but I can have a linear approximation of this decision boundary something like this. A neural network in the simplest form actually tries to form a collection of such straight line boundaries or piecewise straight line boundaries and that collection of straight lines actually model a complicated decision boundary like this. So, what you have in neural network let us see.

(Refer Slide Time: 17:25)



A neural network has one input layer, one output layer and none or more hidden layers. This is the output layer and this is input layer. In every layer you have a number of neurons. So here I can have one or more number of hidden layers, right? Now in absence of any hidden layer if I have just an input layer and outputs layer this what is single layer perceptron. If I have hidden layers within that I can have just one hidden layer I can have two hidden layers and three hidden layers and so on.

More and more hidden layers you incorporate more and more complicated decision boundary it can encode. So, if I have one or more hidden layers it is called a multilayer perceptron or MLP. Right now what you have is on each of the nodes of one layer you have connections to every node of the upper layer. So, from every node in the input layer I have connections to every node in the upper layer.

So, it continues like this and each of these connections have a connection weight. Similarly, from the i th layer to $i+1$ st layer every node has a connection. Similarly, over here from every node on this layer I have connections to every node in the output layer, so this way it continues. And each such connection has an associated weight with it, right? So when I take any node in let us say j .

So, let us call it a j th node and it gets connection from every node in the i th layer or every node in the previous layer. So, let us take a particular node say i th node to the j th node the total sum of inputs coming from the layers in the i th node will be given by w_{ij} . This is the connection weight from the i th node to the j th node times input to this, which let us

say X_i . Then summation of this over all i because this node is getting the inputs from every node in the previous layer.

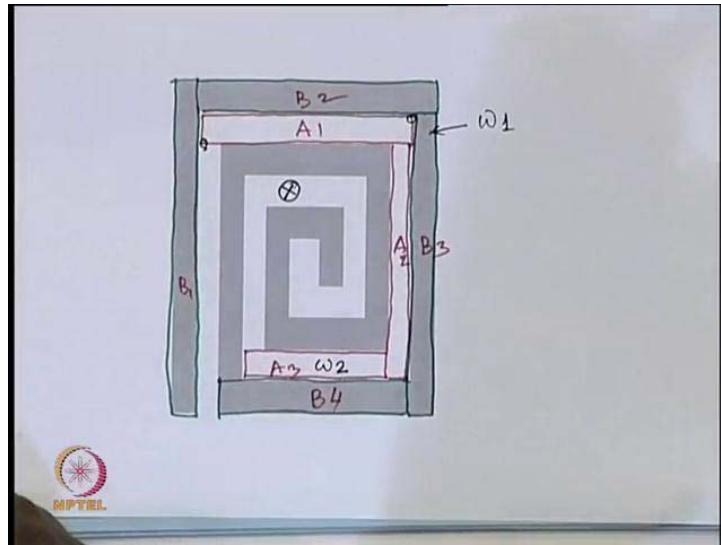
So, this i corresponds to i^{th} node in the previous layer. So, this is the total input to this j^{th} node, right, then you apply a non-linear function of this f and this gives you the output of the j^{th} node which is o_j . Right? Now if I forget about those non linearity, what is output of the j^{th} node o , that is nothing but $o_j = f\left(\sum_i w_{ij}x_i\right)$. And you find that it is nothing but a linear equation. It is linear combination of all these inputs.

Similarly, for every j in this node I will get a linear equation. Right? Coming to the next layer there also feeding these inputs o_j to the nodes above it which also combines them by a linear equation and that continues till top up to this. So, finally what I get is output of every node on the output layer is nothing but a linear combination of the inputs which are coming to the inputs nodes. And when you take the linear combination, the coefficients of this linear equation, they are different for different outputs, which is nothing but a combination of the connection weights coming from all these different layers.

So, effectively what I have is, I have a number of linear equations which are actually encoded within this weight vectors or weight matrices. So, more and more number of connections or weights I have within this network, more and more number of linear equations I can have. So, as a result such a kind of complicated boundary is actually modelled by a number of linear equations by such a simple kind of neural network which is multilayer perceptron or MLP.

So, will come to details of this later on and how this is related to some sort of analytical linear classifier that we can design following a criterion, which is called a perceptron criterion. Now, you find that in all these cases the decision boundary are such that either the classes are linearly separable or the classes are non-linearly separable.

(Refer Slide Time: 23:56)



Now, let us consider a situation something like this. So, you find that it is a spiral shaped object and if I assume that all the points which are belonging to this grey shades, suppose this is the distribution of two-dimensional feature space, right? So if I assume that all the points which belong to this grey shades they belong to class ω_1 and the points belonging to the white region they belong to class ω_2 .

So, here you find that the points are not coming as cluster of points, but the points are distributed following a structure, but the structure is so complicated that the points belonging to one class is totally intermixed with the points belonging to other class. Is it possible to have decision boundaries in such cases so that the classification will still be successful?

So, you find that neither linear classifier nor quadratic classifier. Nor even this simple multilayer perceptron that you have discussed just now will be able to give me, will be able to model a decision boundary which can classify the points belonging to two classes which is the simplest problem, two class problem that we talked about in pattern recognition.

So, I cannot have a decision boundary so easily if the points are distributed like this, so one of the options that we can go in this particular case. Suppose, I define number of rectangles like this, so suppose this is one rectangle or boxes, this is another box, this is another box, this is another box and so on. Similarly, for points belonging to the other

class I can also have number of boxes like this, so it continues like this. So, I actually mark these boxes as class ω_1 , I mark these boxes as class ω_2 .

Now, once I have a box in two-dimension or a hyper box in n dimension, I can represent these boxes by something called mean point and max point or whichever way I represent this boxes. So, what I actually have is I have multiple number of such boxes or multiple number of such hyper boxes. Now, if somehow I can put collect this hyper boxes under the same umbrella. So, all these boxes B_1, B_2, B_3, B_4 they are put under the same umbrella named as ω_1 .

Similarly, these boxes let me call as A_1, A_2, A_3 and so on these boxes I put under another umbrella named as ω_2 . So, actually I have a collection of boxes put under two umbrellas one umbrella is ω_1 other umbrella is ω_2 , right? Now, if get an unknown pattern suppose the unknown pattern is falling over here. If I have a representation of the box, I can easily find out in which of the boxes this unknown pattern is falling, this unknown feature vector is falling. Then I look at the umbrella under which this box belongs and accordingly, I can say whether this unknown sample belongs to class ω_1 or it belongs to class ω_2 .

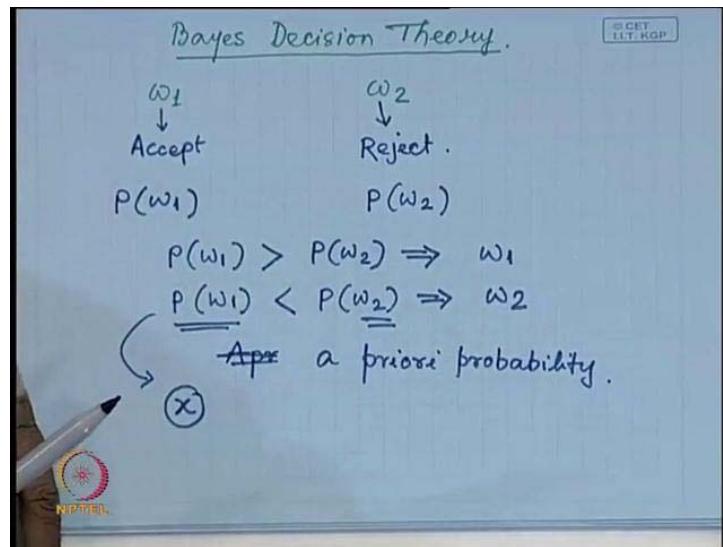
So, this is what is known as hyper box, the classification based on hyper boxes. Such hyper boxes can also be represented by neural network, in which gives the neural network will represent this boxes in terms of two points. One is min point and one is max point. So, I can say this is the min point, this is the max point because this is having the minimum X_1, X_2 feature values, this is having the maximum X_1, X_2 feature values. So, once I have this mean point and max point it is something like left bottom corner and top right corner if I have these two then immediately I can draw a rectangle I do not need any other information provided.

So, if this boxes represented by this min point and this max point it is what is called min max classifier. On top of this if I incorporate some idea of fuzzy set, this is what fuzzy min max hyper boxes. So, if time permits will come to all these details later on. So, far I have talked about all these different concepts just to tell you that what is the domain of pattern recognition problems?

And how complicated the pattern recognition problems can be starting from the similar, very, very simple linearly separable classes to non-linearly separable classes to classes,

which can neither be separated by linear boundary nor by simple and non-linear boundary rather the classes will be represented as collection of chunks of datasets, which have some class belongings. Is that okay? So, these are the different problem domains or dimensions of the problem that we can deal with when we talk about pattern recognition. Now, let us come back to what I said that, will be my topic of discussion today that is Bayes decision theory.

(Refer Slide Time: 30:57)



Now, to discuss about this Bayes decision theory let me take a very, very simple example. The example is, suppose I have a manufacturing industry that manufactures machine parts. Now, you know that in any industry there is section called quality control or quality inspection. What is the job of that particular section, they go for inspection of the products produced in that firm? After inspection if they find that the products are acceptable there meeting in the norms and specifications set for that particular product. It will report under accepted category. If the product is not acceptable there is some defects it will be put under reject category.

So, I was talking about two class problem one class I said ω_1 and the other class I said ω_2 . Here, let me assume that this class ω_1 actually means the category accept and class ω_2 actually means category reject. Right? So, when this quality control department then put an object under accept category or under reject category, they look at some of the features or something of that particular object to decide about this.

Now, out of this if I want to automate the process let me assume that I want to form a decision rule to decide whether the object will be rejected or the object will be accepted. So, for that suppose I want to go for the supervised learning mode. So, I have to take the previous history that how many objects have been rejected and how many objects have been accepted by the quality control department. And based on that I generate two probabilities one is $P(\omega_1)$, that is the probability that the object will be accepted or the probability that the object belongs to class ω_1 . And the other probabilities $P(\omega_2)$ that is the probability that the object will be rejected, or the probability that the object belongs to ω_2 .

So, once I have these two probability I can form very simple decision rule. The decision rule can be something like this, that if $P(\omega_1) > P(\omega_2)$ then you decide in favour of ω_1 , or if $P(\omega_1) < P(\omega_2)$ then you decide in favor of class ω_2 . So, here you find that though here we have been able to form a decision rule, but this decision rule is not really logical.

The simple reason is if from the history I have found out that $P(\omega_1) > P(\omega_2)$, then all of the new coming objects I will always decide in favour of ω_1 even if it should actually belong to class ω_2 , or if $P(\omega_2) > P(\omega_1)$, I will always decide in favour of ω_2 . $P(\omega_1) < P(\omega_2)$, I will always decide in favour of ω_2 even if the object may actually belong to class ω_1 . That means an objects will always be rejected or it will always be accepted based on or a priori probability $P(\omega_1)$ and $P(\omega_2)$.

So, this is not at all a logical decision that we are taking. So, to make our decisions more logical what we have to do is, along with this a priori probability, so these are called a priori probabilities. So, to make our decisions more logical along with this a priori probability, we have to combine some feature let us say feature x. And in the simplest form what this feature x can be. Suppose my decision whether the object will be accepted or whether the object will be rejected is based on the finishing or the polish given to that particular object.

So, it is the quality of the polish if I can quantize that if I can measure that, then that quality of polish be represented by this variable x. It is good, very good, excellent, bad,

and so on. So I can have different sorts of measuring this. And suppose it is measurable then that becomes an observation, and this observation is represented by this feature x . By feature x can have various values.

So, what I would like to do is, along with this a priori probability, I will also make use of this observation x or feature x to decide whether the object showed belong to class ω_1 or the object showed belong to class ω_2 , is that okay? So again I go for the supervised learning mode that means using some objects for which the decision has already been taken.

So, I take some objects from class ω_1 that is the objects which are accepted and I also take some samples of the objects from class ω_2 that is the objects which are rejected. And I measure this feature x for those objects which belong to class ω_1 and I also measure the same x for the objects which belong to class ω_2 . That means I can find out a probabilistic measure or probability density function of variable x for the objects which belong to class ω_1 . I can also find out the probability density function of the same observation x for the objects belongs to class ω_2 .

(Refer Slide Time: 38:24)

The whiteboard contains the following handwritten text:

$p(x|\omega_1)$ $p(x|\omega_2)$
 Class conditional PDF
 $p(\omega_1|x)$ $p(\omega_2|x)$

$$p(\omega_i, x) = p(\omega_i|x) \cdot p(x)$$

$$= p(x|\omega_i) p(\omega_i)$$

$$\Rightarrow p(\omega_i|x) \cdot p(x) = p(x|\omega_i) p(\omega_i)$$

$$p(\omega_i|x) = \frac{p(x|\omega_i) p(\omega_i)}{p(x)}$$

That is, I can find out what is $P\left(\frac{x}{\omega_1}\right)$. So, this is nothing but the probability density function of x taking the objects from class ω_1 . I can also find out $P\left(\frac{x}{\omega_2}\right)$ by taking the

objects from class ω_2 . So, I can find out $P(x/\omega_1)$, I can find out $P(x/\omega_2)$. So, these are the probability density functions which are called class conditional probability density function. So, class conditional pdf. Right? So, I have $P(x/\omega_1)$, I have $P(x/\omega_2)$.

Now, my recognition problem is, the decision problem is for an unknown object I can measure x . And from this measurement x , measurement of the future x I have to decide whether I should put this object in class ω_1 or I should put this object in class ω_2 . That means what I am interested in is that is my decision should be based on $P(\omega_1/x)$ because I have this observation x . And based on that I have to take decision ω_1 or I have to take decision $P(\omega_2/x)$. Right?

So, if I find that if $P(\omega_1/x) > P(\omega_2/x)$ then I will decide in favour of class ω_1 . If $P(\omega_2/x) > P(\omega_1/x)$ then I have to decide in favour of ω_2 . And this decision appears to be more logical than our simplest decision that if the a priori probability are more then I will put them in one class and the more logical will be if this probability density function that is $P(\omega_1/x)$ or $P(\omega_2/x)$ can be combined with a priori probabilities $P(\omega_1)$ and $P(\omega_2)$. So, if I can combine these two then I will have a more logical decision rules.

So, let us see that how we can combine these two. Now from the preliminary probability theory you might know that the joint probability distribution, the joint probability density function that is an object belongs to class to class say ω_i . Let me generalize this instead of calling ω_1 and ω_2 , let me call it ω_i , I can have a value one or two.

So joint probability that an object belongs to class ω_i and at the same time has the feature x , this is not a class conditional probability this is joint probability. So, an object having, is taken from class ω_i and at the same time it will have the feature x . So, this joint probability density function is given by in terms of class condition probability. This is nothing but $P(\omega_i, x) = P(\omega_i/x).P(x)$ or this is same as $P(x/\omega_i).P(\omega_i)$. Is that okay?

So this joint probability can be written in terms of conditional probability that $P(\omega_i/x)$.

That is the joint probability that an object is taken from class ω_i and at the same time it has a feature x , is nothing but $P(\omega_i/x)$ that is the conditional probability into $P(x)$ and which is nothing but $P(x/\omega_i) \cdot P(\omega_i)$. This is again a conditional probability and this is the a priori probability that you have already studied. Now, from here you find that I get a very simple expression that $P(\omega_i/x) \cdot P(x)$ is same as $P(x/\omega_i) \cdot P(\omega_i)$.

So I get this expression from this preliminary probability theory. Now, from here what I get is, I already know what is the of ω_i . That is a priori probability based on what is the history of classification in that particular form, how many objects have been rejected how many objects have been accepted out of the total number of objects that has been produced in that form. I take objects belonging to different classes that mean those objects which have been rejected. I also take those objects which have been accepted. And based on that I find out the class conditional probability density function of x that is $P(x/\omega_i)$.

So, this $P(x/\omega_i) \cdot P(\omega_i)$ they are known to you. For an unknown object, I measure the feature x and what I have to find out. I have to find out $P(\omega_i/x)$ because then only I can say whether or I will be able to say that whether this particular object having this feature x should be classified into ω_1 or it should be classified into ω_2 , right? So, from here I get $P(\omega_i/x)$ is nothing but $P(x/\omega_i) \cdot \frac{P(\omega_i)}{P(x)}$. So, from here I can have my decision rule.

(Refer Slide Time: 45:22)

$p(\omega_i|x) = \frac{p(x|\omega_i) p(\omega_i)}{p(x)}$
 a posteriori probability

$$p(x) = \sum_{i=1}^2 p(x|\omega_i) \cdot p(\omega_i)$$

$$p(\omega_1|x) > p(\omega_2|x) \Rightarrow \omega_1.$$

$$p(\omega_1|x) < p(\omega_2|x) \Rightarrow \omega_2$$

So, as I have $P\left(\frac{\omega_i}{x}\right)$, let me repeat this expression $P\left(\frac{\omega_i}{x}\right)$ is nothing but $P\left(\frac{x}{\omega_i}\right) \cdot \frac{P(\omega_i)}{P(x)}$, where what is this $P(x)$, $P(x)$ is nothing but $P\left(\frac{x}{\omega_i}\right)$ that is class conditional probability into $P(\omega_i)$ take the summation because I have got only two classes ω_1 and ω_2 . So, $i = 1$ to 2 is that okay? So, what I have is, I have class conditional probability density functions of the feature x , I have a priori probability that is $P(\omega_i)$.

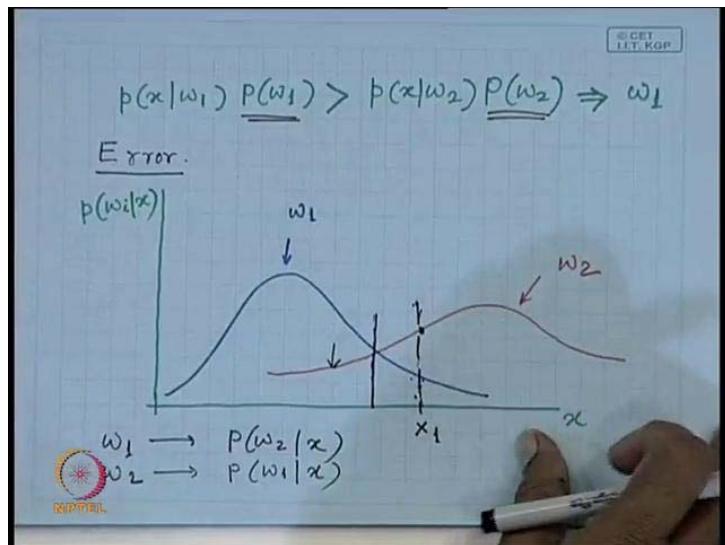
And using these two I am computing $P\left(\frac{\omega_i}{x}\right)$, which is called a posteriori probability.

Now, over here my simple decision rule. So, this is what Bayes theory actually, right? And from this Bayes theory I can have simple Bayes decision rule which will be that if $P\left(\frac{\omega_1}{x}\right) > P\left(\frac{\omega_2}{x}\right)$ then you decide in favour of class ω_1 . Or if $P\left(\frac{\omega_1}{x}\right) < P\left(\frac{\omega_2}{x}\right)$ then you decide in favour of class ω_2 .

So, this is my simple decision rule, but when I perform this decision when I take this decision I make use of a posteriori probability. And this a posteriori probability actually combines the class conditional probability and the a priori probability. So, I make use of both that is class conditional probability and a priori probability, to make, to take this

sort of decision. Now, here you find that over here my condition simply becomes because $P(x)$ will appear at the denominator for both the classes ω_1 and ω_2 .

(Refer Slide Time: 48:30)



So, if I expand this expression my expression will be $P\left(\frac{x}{\omega_1}\right) \cdot P(\omega_1)$, this is nothing but

$P\left(\frac{\omega_1}{x}\right)$ I am not taking into consideration $P(x)$ because that appears in the

denominator for both classes ω_1 and ω_2 . So, this is my $P\left(\frac{\omega_1}{x}\right) = P\left(\frac{x}{\omega_1}\right) \cdot P(\omega_1)$.

So, if $P\left(\frac{\omega_1}{x}\right) > P\left(\frac{\omega_2}{x}\right)$ then I take decision in favour class ω_1 that means I say this object belongs to class ω_1 . Right?

So, over here if $P(\omega_1) = P(\omega_2)$ that means, the particular firm produces objects which are equally likely to be rejected or to be accepted. In that case my decision is based on the $P\left(\frac{x}{\omega_1}\right)$ and $P\left(\frac{x}{\omega_2}\right)$ that is class conditional probability.

Whereas if this is same that means for a given x if belonging to class ω_1 or ω_2 they are equal, then my decision is based on the a priori probability $P(\omega_1)$ and $P(\omega_2)$. So, if I cannot take a decision based on the observation I make use of a priori probability. If I cannot take a decision based on a priori probability, I make use of observation. In other

cases, you consider both to take the decision. Is that okay? So, this particular Bayes decision rule combines both a priori probability and the class conditional probability to give you a decision rule which is more logical, than simple rule based on a priori probability only.

Now, once we have this, what is the error that will encounter? So, what is the probability of error or what is the total error that will have in such cases? Now, let us see suppose

this is my x and along the vertical axis I plot the posteriori probability $P(\omega_i/x)$. And

suppose the posteriori probability is something like this. This is for say ω_1 , that is $P(\omega_1/x)$. Similarly, this is for ω_2 , that is $P(\omega_2/x)$. Is that okay? So over here my

decision rule was whenever $P(\omega_1/x) > P(\omega_2/x)$ given x , I decide in favour of class ω_1

and when $P(\omega_2/x) > P(\omega_1/x)$, I decide in favour of class ω_2 .

So, my decision boundaries actually the point where $P(\omega_1/x)$ is same as $P(\omega_2/x)$. So

this is my decision boundary. However, $P(\omega_1/x) > P(\omega_2/x)$, but still you find that for a

given x if I decide in favour of ω_1 still there is a finite probability that the object of may belong class ω_2 . Because here P of ω_2 given x is non zero. If it was 0 then I would have said that there is no error, but there is a nonzero probability that the object may belong to class ω_2 .

So, there is a finite probability of error. And what is the probability of error, if I decide in favour of class ω_1 , the probability of error is, the probability that the object may belong to class ω_2 whereas as if I decide in favour of class ω_2 then the probability of error is the probability to that the object may belong to class ω_1 . So, it is very simple that if I decide in favour of ω_1 then the error probability is $P(\omega_2/x)$.

Whereas if I decide in favour of ω_2 , then the probability of error is $P(\omega_1/x)$, but

whatever is the probability of error will that is the minimum possible that I can have, is it not? Because if I decide for an object for which the value of x is here say x_1 over here

$P(\omega_1/x) < P(\omega_2/x)$. But if I decide the object to belong to class ω_1 my probability of error is $P(\omega_2/x)$ which is quite high.

Whereas, if I decide in favour of ω_2 then the probability of error is $P(\omega_1/x)$ which is less than $P(\omega_2/x)$. So, whatever decision that we take based on a particular observation x , the Bayes decision rule to ensure that the probability of error is minimized, is that okay? So, given a situation like this, what is the total error that we can have, that is probability of error?

(Refer Slide Time: 55:04)

$$\begin{aligned}
 P(\text{error}) &= \int_{-\infty}^{\infty} p(\text{error}, x) dx \\
 &= \int_{-\infty}^{\infty} p(\text{error}|x) \cdot p(x) dx \\
 p(\text{error}|x) &= \min \left\{ p(\omega_1|x), p(\omega_2|x) \right\}
 \end{aligned}$$

$P_{\text{error}} = \int_{-\infty}^{\infty} P(\text{error}, x) dx$, because over here you find that asymptotically the error value extends up to $+\infty$ on the positive side and extends up to $-\infty$ on the negative side.

So, the total error, P_{error} will be given by $P_{\text{error}} = \int_{-\infty}^{\infty} P(\text{error}, x) dx$, which is nothing but

$P_{\text{error}} = \int_{-\infty}^{\infty} P(\text{error}/x) \cdot p(x) dx$. And what is this $P(\text{error}/x)$, if I decide in favour of ω_1 , this $P(\omega_2/x)$ and if I decide in favour of ω_2 it is $P(\omega_1/x)$. So, this $P(\text{error}/x)$ is nothing

but $P(\text{error} \not\propto x) = \min \left\{ P(\omega_1 \not\propto x), P(\omega_2 \not\propto x) \right\}$. So, for a particular value of x whichever is minimum whether it is $P(\omega_1 \not\propto x)$ or $P(\omega_2 \not\propto x)$ whichever is minimum $P(\text{error} \not\propto x)$ is that only because I am taking the decision in favour of the other, So, let us stop here today.

Pattern Recognition Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 6
Bayes Decision Theory
(Contd.)

Good morning, so we will continue with our discussion on Bayes theory, the discussion that we started in our last class.

(Refer Slide Time: 00:33)

The image shows handwritten notes on a whiteboard. At the top, it says "Bayes Decision Theory." Below that, there is a diagram with two vertical arrows pointing downwards from $p(\omega_1)$ and $p(\omega_2)$. The left arrow is labeled "Accepted" and the right arrow is labeled "Rejected". Below the diagram, there are two probability expressions: $p(x|\omega_1)$ and $p(x|\omega_2)$. Below these, the formula for the posterior probability is given as $p(\omega_1|x) = \frac{p(x|\omega_1)p(\omega_1)}{p(x)}$ and $p(\omega_2|x) = \frac{p(x|\omega_2)p(\omega_2)}{p(x)}$. At the bottom, the normalization factor is shown as $p(x) = \sum_{i=1}^2 p(x|\omega_i)$. The IIT Kharagpur logo is visible at the bottom left.

So, we were discussing about Bayes decision theory. So, what we talked about yesterday is that, suppose in a particular classification or pattern recognition application domain, we have a set of objects where we have for that particular object, we have an observation say x . So, we have said that x is an observation, and based on this observation we have two classify the object in one of the two classes. So, taking that particular example of a manufacturing industry, we have to classify that object either into the accepted category or in the rejected category.

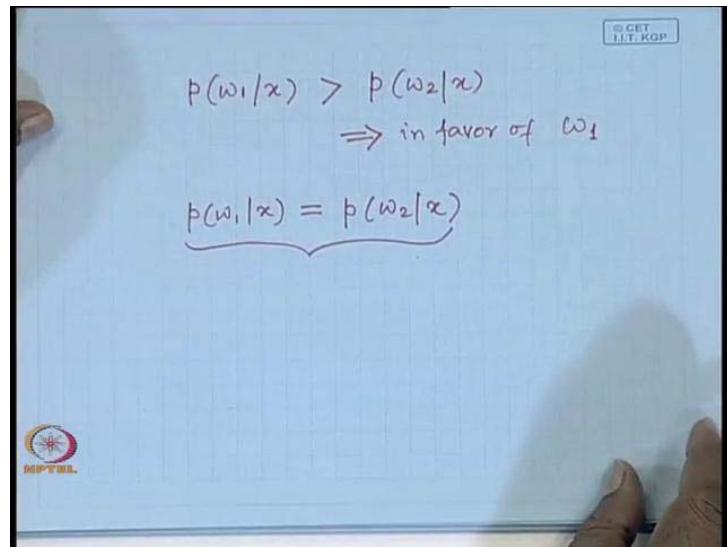
So, for doing this job we have said that we have some a priori probability that is, $P(\omega_1)$ and $P(\omega_2)$. Where, ω_1 this category you have said that let us assume that this is the category of accepted objects. And ω_2 is the category of rejected objects. And $P(\omega_1)$ is the a priori probability that the object will be accepted and $P(\omega_2)$ is the a priori probability that the object will be rejected.

Now, in addition to make our decision we have said that, we have an observation x . And for this observation x , also we have class conditional probability density function that is, probability $P(x/\omega_1)$ and also probability $P(x/\omega_2)$. So, these two are the class conditional probability density function which we have to estimate based on the measurements on x or objects which are unknown to belong to class ω_1 , and the measurement of x from the objects which are known to belong to class ω_2 .

So, from this two we have two estimates this $P(x/\omega_1)$ and $P(x/\omega_2)$. But finally, our classification problem is that we have an observation x . And based on this observation x we have to put this object either in category ω_1 or in category ω_2 or effectively what we have to find out is, $P(\omega_1/x)$ and $P(\omega_2/x)$. So, these are the two probability measures, which we say of the posteriori probability. And based on these two probability measures we have to decide whether the object has to be classified to class ω_1 or the object is to be classified to class ω_2 .

So, here $P(\omega_1/x)$, we have said from Bayes rule that this is nothing but $P(x/\omega_1) \cdot \frac{P(\omega_1)}{P(x)}$ x. Similarly, $P(\omega_2/x)$ is nothing but $P(x/\omega_2) \cdot \frac{P(\omega_2)}{P(x)}$ where you find that this p of x appears in the denominator of both these expressions, where this $P(x) = \sum_{i=1}^2 P(x/\omega_i)$. So, this is what we get from Bayes rule.

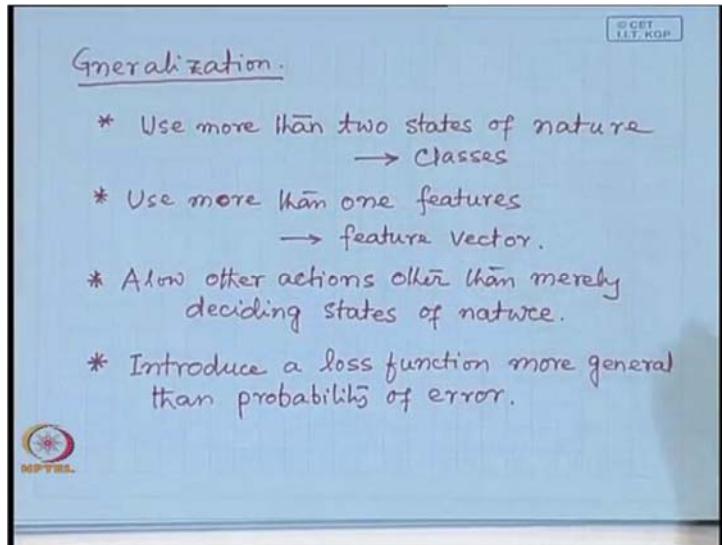
(Refer Slide Time: 05:01)



And then our decision was like this that, if $P\left(\frac{\omega_1}{x}\right) > P\left(\frac{\omega_2}{x}\right)$ then we decided in favour of class ω_1 . So, in favour of ω_1 that means we decide that the object having this observation x belong to class ω_1 . If it is otherwise, $P\left(\frac{\omega_2}{x}\right) > P\left(\frac{\omega_1}{x}\right)$ then we will decide that the object belongs to ω_2 .

However, you have find that we have one condition that if $P\left(\frac{\omega_1}{x}\right) = P\left(\frac{\omega_2}{x}\right)$. So, this is the case when we cannot take any decision because the object lies on the decision boundary between the class ω_1 and ω_2 . So, it may be both in class ω_1 as well as in class ω_2 . So, this is a case where decision cannot be taken. So, this was our basic Bayes decision theory where, we have taken that we have two class ω_1 and ω_2 . And our decision was one of the two decisions either the object has to be put in class ω_1 or the object has to be put into class ω_2 . Now there can be a generalization of this Bayes theory.

(Refer Slide Time: 06:47)



So, the generalised Bayes theory can be put something like this. So, instead of classes let us call it the states of nature. So, earlier in the example that you have taken there are only two states of nature. Now, we can generalise it having multiple states of nature. So, the first generalisation is like this that, use more than two states of nature. And then this state of nature in our case, in this classification problem is nothing but the classes. So, that will be our understanding when we talk about when we say that the states of nature it is nothing but the classes. In the earlier case we have taken a single observation x . So, based on this observation x we have tried to decide whether we have to put the object in class ω_1 or we have to put into class ω_2 .

Now, in the generalization we allow more than one observations that means instead of having a single feature we have allowed feature vector. So, use more than one feature so that means we are going for feature vector. The other generalization is, in the earlier case we had only two actions that is either decide about class ω_1 or decide about class ω_2 . So now, we can allow a number of actions instead of just deciding whether this belongs to class ω_1 or this belongs to class ω_2 . So just this case I have shown that if

$$P(\omega_1/x) = P(\omega_2/x), \text{ I cannot take any decision.}$$

So, this particular fact that I cannot take any decision, I can also define this is as action that I cannot decide to which class does the object belongs that also I can call as an action. This no decision is also action. So, I can allow other actions, other than merely

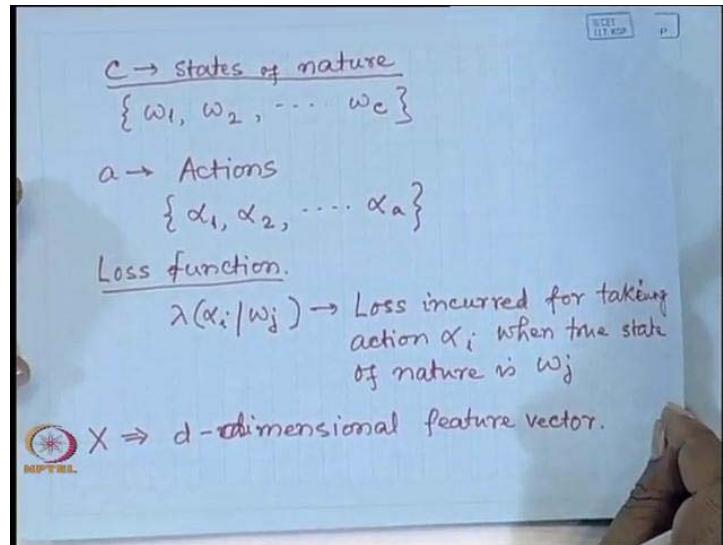
deciding states of measure. So, when as I said the states of measure in our cases are classes. So, merely deciding whether the objects belong to class ω_1 or the object belongs to class ω_2 . Apart from that I can take other actions as well.

And the fourth generalisation is this that in our case our decision was based on a posteriori probability that if $P(\omega_1/x) > P(\omega_2/x)$ then we decided to be in favour of ω_1 .

If it is other way round that is $P(\omega_2/x) > P(\omega_1/x)$ then we decide in favour of class ω_2 .

So, in this case we can have a more generalized criteria based on which we can decide about the states of nature which is called a loss function. So, introduce a loss function, which is more general than probability or let us say probability of error P_{error} . Because as we said that, if we decide in favor of ω_2 then the probability of error is $P(\omega_1/x)$. And the purpose was to reduce the probability of error in simple Bayes decision theory. So, as I said that will have more than one states of nature.

(Refer Slide Time: 12:22)



So, let us assume that there is C numbers of states of nature or C number of classes to be more specific in our application. And those classes let us call a set of classes ω_1, ω_2 up

to say ω_c . So, there are C states of nature. Now, about the actions as you also said that will also allow actions rather than merely deciding about the states of nature.

So, suppose there are a number of actions. So, I have this state of actions which I represent as $\{\alpha_1, \alpha_2, \dots, \alpha_a\}$. So, this is the state of actions that I have apart from simply deciding that whether the object belongs to class ω_1 or it belongs to class ω_2 and so on. So, I have this number of actions and I said that we introduce a loss function which is more general than probability of error. So, this loss function I represent as $\lambda(\alpha_i / \omega_j)$. So this means that if the actual state of nature is ω_j whereas we take an action α_i .

Then, the loss incurred while taking this action α_i when, the actual state of nature is ω_j is $\lambda(\alpha_i / \omega_j)$. So this is loss incurred for taking action α_i when true state of nature is ω_j . And the fourth generalisation is said that instead of considering a single feature, we will consider a feature vector. So, here let us assume that we have a feature vector that instead of a single feature we have a feature vector that is a multiple number of features of multiple observations of various parameters, which is a feature vector x .

And this feature vector x is d -dimensional. So, these are things that we have c number of states of nature given by ω_1 to ω_c . We have a number of actions from α_1 to α_a . We have a general loss function which is given by $\lambda(\alpha_i / \omega_j)$ that means the loss incurred for taking an action α_i , when the true state of nature is ω_j . And we consider a feature vector x which is a d -dimensional feature. Now, let us see that how this decision rule in this generalised based theory has to be taken.

(Refer Slide Time: 16:37)

Action α_i

$$R(\alpha_i | x) = \sum_{j=1}^c \lambda(\alpha_i / \omega_j) P(\omega_j | x)$$

Risk function / Conditional Risk /
Expected loss.

So, suppose we have an

⇒ Minimum Risk Classifier.

object and for that object we have made an observation vector or the feature vector given by x . So x is the feature vector, which is of dimension d . And for this feature vector we take an action α_i . So, as we said earlier that the loss incurred for taking an action α_i , while the true state of nature is ω_j , is given by $\lambda\left(\frac{\alpha_i}{\omega_j}\right)$. Right?

So, here for this feature vector x from this observation x , we have taken an action α_i , but we do not know what is the true state of nature. It may be ω_1 , it may be ω_2 , it may be ω_3 and so on. The true state of nature may be anything. So, for each of these I will incur a loss function. Right? So, if the true state of nature is ω_j , I will incur a loss which is given by $\lambda\left(\frac{\alpha_i}{\omega_j}\right)$. And if the probability that the true state of nature is ω_j given the feature vector x then the average loss or the average risk can be computed like this.

This average risk or expected loss can be $R\left(\frac{\alpha_i}{x}\right) = \sum_{j=1}^c \lambda\left(\frac{\alpha_i}{\omega_j}\right) P\left(\frac{\omega_j}{x}\right)$, x is my observation vector, take the summation over $j = 1$ to c . Is that okay? Because I do not know, what is true state of nature. So, if the true state of nature is ω_j then my loss function is $\lambda\left(\frac{\alpha_i}{\omega_j}\right)$. Then I have to multiply this with the probability of true state of nature being ω_j given my observation vector x that is $P\left(\frac{\omega_j}{x}\right)$.

So this is $R\left(\frac{\alpha_i}{x}\right) = \sum_{j=1}^c \lambda\left(\frac{\alpha_i}{\omega_j}\right) P\left(\frac{\omega_j}{x}\right)$ that gives you the expected loss. So, which we are calling as $R\left(\frac{\alpha_i}{x}\right)$. So, this expected law is also called a Risk function or we can also call it conditional risk because it depends upon x . It is also the expected loss. So, any decision or any action α_i that I had to take that particular α , that particular action for which this risk is minimum or the expected loss is minimum.

Unlike, in the previous case where we used only two classes our decision was taken in favour of that class, which gives us minimum error. That is if I decide in favour of ω_1 , I make sure that the error is minimum which nothing but $P\left(\frac{\omega_1}{x}\right)$. Similarly, if I decide in favour of ω_2 my error $P\left(\frac{\omega_2}{x}\right)$ which is minimum in that case.

In the generalised case, I had to take that action α_i for which this risk $R\left(\frac{\alpha_i}{x}\right)$ is minimum. So, accordingly this is also called minimum risk classifier. So, in this generalised case, the kind of classifier that I have is a minimum risk classifier. Now, later on we see various derivatives of this minimum classifier, it is this minimum classifier which under different conditions leads to different kinds of classifier which are actually in use.

So, let us see two category cases. Suppose, I have two classes ω_1 and ω_2 or two states of nature ω_1 and ω_2 . So, in this case and if I assume that the action means saying whether the object belongs to class ω_1 . So, α_1 means that decision that object belongs to class ω_1 , α_2 means decision that object belongs to class ω_2 .

(Refer Slide Time: 22:23)

$$\begin{aligned} \omega_1 & \quad \omega_2 \\ \alpha_1 & \quad \alpha_2 \\ \lambda(\alpha_i|\omega_j) & \Rightarrow \lambda_{ij} \\ R(\alpha_i|x) & = \sum_{j=1}^2 \lambda(\alpha_i|\omega_j) P(\omega_j|x) \\ R(\alpha_1|x) & = \lambda_{11} \cdot P(\omega_1|x) + \lambda_{12} \cdot P(\omega_2|x) \\ R(\alpha_2|x) & = \lambda_{21} \cdot P(\omega_1|x) + \lambda_{22} \cdot P(\omega_2|x) \\ (\lambda_{21} - \lambda_{11}) P(\omega_1|x) & > (\lambda_{12} - \lambda_{22}) P(\omega_2|x) \end{aligned}$$

So, I have two states of nature one is ω_1 other one is ω_2 . As I have this ω_1 and ω_2 and also I have action the α_1 and α_2 so these are the actions. So, action α_1 means the decision that the object belongs to class ω_1 , action α_2 means deciding that object belongs to class ω_2 , so over here now if I write $\lambda\left(\frac{\alpha_i}{\omega_j}\right)$ as say λ_{ij} , just for simplicity of expression. So, λ_{ij} means $\lambda\left(\frac{\alpha_i}{\omega_j}\right)$. That is the loss incurred for taking an action α_i when the true state of nature is ω_j .

So, for this two class problem I can have risk function $R(\alpha_i/x)$, as we said is nothing but

$$R(\alpha_i/x) = \sum_{j=1}^c \lambda_j(\alpha_i/\omega_j) P(\omega_j/x).$$

We took the summation for $j = 1$ to c for all possible true state nature. So, in our two class problem this expression simply becomes that if I take action α_1 , so I will have $R(\alpha_1/x)$.

This will be λ_{11} that means $\lambda(\alpha_1/\omega_1)$, is that okay, into $P(\omega_1/x) + \lambda_{12}$, that means

$$\lambda(\alpha_1/\omega_2) \text{ into } P(\omega_2/x).$$

If I expand the lost function the expected loss for taking an action α_i on an observation vector x then this is the expansion of the loss function, expected loss function or the risk function.

Similarly, I have the other option of taking action α_2 , I can take one of this two actions. So, the risk involved for taking action α_2 on observation vector x is nothing but $R(\alpha_2/x) = \lambda_{21}P(\omega_1/x) + \lambda_{22}P(\omega_2/x)$. Now, I said that I have to take that action for which the risk is minimum.

So, if I find after computation of $R(\alpha_1/x)$ and $R(\alpha_2/x)$ that $R(\alpha_1/x) < R(\alpha_2/x)$ then have

to take action α_1 . If it is otherwise that $R(\alpha_2/x) < R(\alpha_1/x)$ then I have to take action α_2 .

So, under that condition for deciding in favour of class ω_1 or for taking an action α_1 , where $R(\alpha_1/x) < R(\alpha_2/x)$. You can find that this leads to a condition that

$$(\lambda_{21} - \lambda_{11})P(\omega_1/x), \text{ this has to be greater than } (\lambda_{12} - \lambda_{22})P(\omega_2/x).$$

So, this is the condition if I have to take this decision in favour of class ω_1 or if I have to take action α_1 . Now, over here you find that this λ_{11} means taking an action α_1 when the true state of nature is ω_1 . And as we are saying in our case taking an action in α_1 means deciding that object belongs to class ω_1 so we are taking the correct decision.

Similarly, λ_{22} this is the loss incurred for taking an action α_2 when the true state of nature ω_2 . So, this is also a case when we are taking a correct decision so this is loss involved

for taking correct decision. Right? Whereas, this λ_{21} and λ_{12} these are the loss for taking wrong decisions because we taking action ω_1 , where the true state of nature is actually ω_2 . Or we are taking an action ω_2 when the true state of nature is actually ω_1 .

So, naturally this λ_{21} will be much larger than λ_{11} because this is loss for taking correct action and ideally this should be equal to zero because that action is correct. Similarly, that λ_{12} is much greater than λ_{22} because this is also the loss incurred for taking wrong decision, whereas λ_{22} is the loss incurred for taking correct decision. So, you find that both of these that $\lambda_{21} - \lambda_{11}$ and $\lambda_{12} - \lambda_{22}$ both of them will be positive or greater than zero. Now I can compare this with the two case or Bayes decision that we have taken earlier or minimum error classification method that we have done earlier.

(Refer Slide Time: 29:04)

$$P(\omega_1|x) > P(\omega_2|x) \Rightarrow \omega_1$$

$$(\lambda_{21} - \lambda_{11}) \cdot P(\omega_1|x) > (\lambda_{12} - \lambda_{22}) P(\omega_2|x)$$

$$\Rightarrow \omega_1.$$

So over there, our condition was $P\left(\frac{\omega_1}{x}\right)$ if it is greater than $P\left(\frac{\omega_2}{x}\right)$ then we have decided in favour of class ω_1 . Now, in case of this generalised one by incorporating the risk function that is in minimum risk classifier, what you have do is, this decision rule that we had taken in simple Bayes decision that actually has to be weighted by the loss difference. Because this lambda $\lambda_{21} - \lambda_{11}$ is nothing but a loss difference for taking wrong decision and for taking a correct decision.

Similarly, $\lambda_{12} - \lambda_{22}$ this is also a loss difference between taking a wrong decision and taking a correct decision. So, the difference between our generalised case and the specific cases is that here, we had the simple expression $P\left(\frac{\omega_1}{x}\right) > P\left(\frac{\omega_2}{x}\right)$ leads you to the decision of ω_1 . In the present term action α_1 , if it is the reverse then I had to take action α_2 whereas, in this minimum classifier simply becomes $\lambda_{21} - \lambda_{11}$ which is weighting this a posteriori probability $P\left(\frac{\omega_1}{x}\right)$ that is greater than $(\lambda_{12} - \lambda_{22}) \cdot P\left(\frac{\omega_2}{x}\right)$. This actually initiates action α_1 or deciding in favour of ω_1 . So, this is what we are getting following the minimum risk classification. Now, as I said that there are derivatives of this minimum risk classification. Under different situations I can have different types of classifiers which are actually derived out of this minimum risk classifier.

(Refer Slide Time: 31:45)

Minimum - Error - Rate
classification.

$\alpha_i \rightarrow$ True state of nature is ω_i

$$\lambda(\alpha_i | \omega_j) = \begin{cases} 0 & i=j \\ 1 & i \neq j \end{cases} \quad i, j = 1, \dots, c$$

$$R(\alpha_i | x) = \sum_{j=1}^c \lambda(\alpha_i | \omega_j) \cdot P(\omega_j | x)$$

$$= \sum_{i \neq j} P(\omega_j | x) = 1 - P(\omega_i | x)$$

So, let us see one such classifier which is minimum error rate classifier. You have any questions so far.

Any question. Those are predefined the loss function are predefined. Depending on the kind of application that you have, you have to define, what is the amount of loss that you will impose for different wrong decision or for a correct decision?

Student: Sir, what means loss incurred for correct decision?

Ideally it is zero, if I take a correct decision then the loss is actually zero. So, ideally it should be zero. However, for more generalisations I can still put a loss function which may be very low. Because as we said in our previous class that, if any I take a decision ω_1 , but the probabilistic point of view there is also a finite probability that the object may actually belongs to class ω_2 . Though, that probability value is very small so ideally the loss involved for taking a correct decision is zero.

But to take care of such cases I can impose a loss even in a correct decision, but that loss value may very low. Is that okay? Because there is always a finite probability that my decision even though I am confident that I am taking a correct decision, but there is a finite probability however, small it is, that my decision can be wrong.

So, that is they can take care of by that λ_{ii} or λ_{jj} . Even there is a situation where I may incur a loss, even if I am confident that I am taking a correct decision from other circumstances. So, what is this minimum error rate classification? So, we said that if I take an action a_i that means I am taking a decision that the true state of nature is ω_i , that is a_1 is true state of nature ω_1 , a_2 is true state of nature ω_2 and so on.

And if I define the loss function like this say lambda a_i given ω_j . So, taking an action a_i means deciding that state of nature is ω_i . So, as I said that if my decision is correct that means if I take action a_i and the true state of nature is also ω_i then ideally I should incur a zero loss.

So, if I define this loss function like that, so I define this $\lambda \left(\frac{a_i}{\omega_j} \right) = 0$, whenever $i = j$, that means I am taking the correct decision. And the loss function I make equal to one whenever $i \neq j$. So, this is how I define my loss function. And this is true for all i and j equal to one to c for all different values of i and j . So, whenever i and j are same that means I have made correct decision, I am taking a correct decision, the loss involved is zero. Whenever, I take a wrong decision the loss involved is one.

So, this is how I define my loss function. So, by this definition of loss function let us see what will be the expected loss or the risks, that is $R \left(\frac{a_i}{x} \right)$ that will be nothing but as

you have already said $R\left(\alpha_i \middle/ x\right) = \sum_{j=1}^c \lambda\left(\alpha_i \middle/ \omega_j\right) P\left(\omega_j \middle/ x\right)$. So, this is the loss involved or

the risk involved for taking an action α_i . Right? This either equal to zero or equal to one. So, it is equal to zero whenever $i = j$ and it is equal to 1 whenever $i \neq j$.

So, this term the summation gets simplified to $R\left(\alpha_i \middle/ x\right) = \sum_{i \neq j} P\left(\omega_j \middle/ x\right)$ Because only

when $i \neq j$ this $\lambda\left(\alpha_i \middle/ \omega_j\right)$ that was equal to 1 and wherever i was be equal to j this was equal to zero. So this summation will be wherever $i \neq j$. Right?

Now, for this α_i , only one value of j which is equal to i , total submission of all these probability values $P\left(\omega_i \middle/ x\right)$ or $P\left(\omega_j \middle/ x\right)$ for all values of j is equal to one. Out of that this is the summation where $i \neq j$. Right? So, this is nothing but $R\left(\alpha_i \middle/ x\right) = 1 - P\left(\omega_i \middle/ x\right)$.

Right? So, if I want to maximise or minimise this risk function that means this term has been minimised. And if I want to minimise this then this p of ω_i has to be maximised.

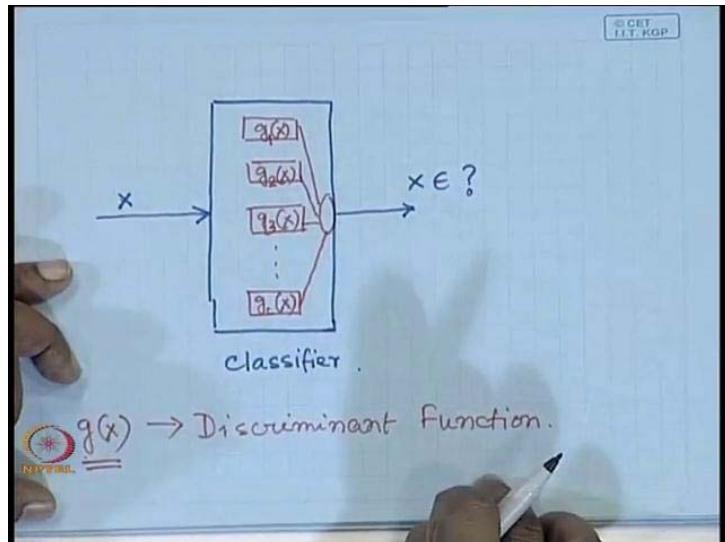
So, you find that I come back to the same decision, similar decision in the generalised case that $P\left(\omega_i \middle/ x\right)$, for whichever i this is maximum I have to take that decision. And that ensures minimum risk, that ensures minimum error rate. So, this minimum risk classifier in this particular case boils down to minimum error rate classifier.

So, that is why I said that starting from there, I can have various derivatives various types of classifiers, but finally, all of them will turn out to be equivalent. But under different situations I can use them differently depending of convenience how we can model a problem in a particular domain. Now, let us come to another concept which is called a discriminant function. So, effectively what are the classifiers you have? Say it is something like this that, i equal to j .

Probability is not zero, $\lambda\left(\alpha_i \middle/ \omega_j\right)$ that is why the term is absent from here.

Here we are taking this probability $P(\omega_i/x)$, $P(\omega_i/x)$ is not zero. But in this expression this $P(\omega_i/x)$ was to be multiplied by λ_{ii} . This λ_{ii} is zero, that is why in this expression this $P(\omega_i/x)$ the term corresponding to this is absent. It is only because of this $P(\omega_i/x)$ is not zero. So, when I have this c class classifier I can put it something like this.

(Refer Slide Time: 41:54)



Say, I have a classifier let us assume that it is a black box. So, this is my classifier block. Input I have an observation vector or feature vector x of dimension d . Then the classifier has to give me a decision that what is the class belongingness of this object having this feature vector x . Now, inside this black box all these different types of calculations are to be done, are to be made. If I go for minimum risk classifier, then for every different action this classifier has to find out that what is the corresponding risk. And it has to take, give me that action for which the risk is minimum.

If I go for Bayes rule for every different class, it has to find out what is the a posteriori probability then whichever class gives me the maximum a posteriori probability the classifier will decide in favour of that class. In case of minimum error rate classifier for every class the classifier has to decide that what the error for taking a particular action. And it will decide in favour of that action which gives the minimum error. Right? And

you find that I have to compute the number of functions which is equal to number classes or which is equal to number of actions that I have in my classifier.

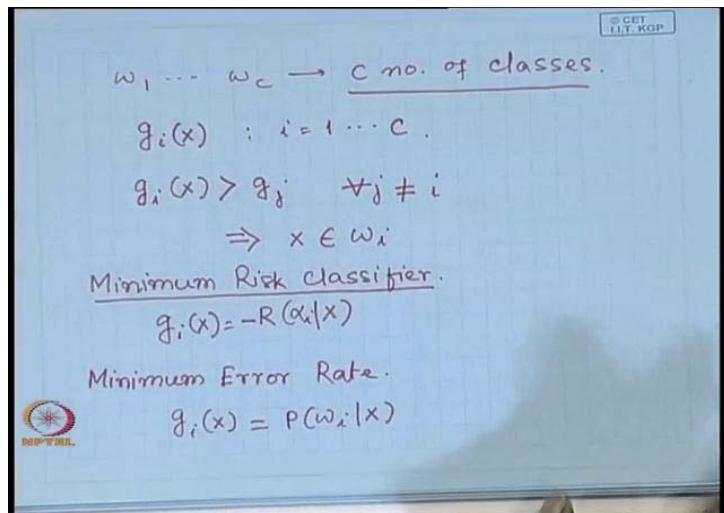
So, many functions are to be computed. Say for every action α_i , for every action α_i in my classifier I have to compute this for every $R\left(\frac{\alpha_i}{x}\right)$. For every class ω_i I have to compute this $P\left(\frac{\omega_i}{x}\right)$ like that. So, it is the number of classes of the number of actions that I defined with my classifier where I have to complete. So many functions then whichever function gives either maximum or minimum. I take that particular action or decide in favour of that particular class. So, I can say that in this classifier box, I have different modules or functional units which computes a function $g(x)$.

So, this computes a function of $g(x)$, $g_1(x)$, this computes $g_2(x)$, this computes $g_3(x)$, this computes $g_c(x)$ on the same feature vector x . Then you take a decision either following the maximum criteria or following the minimum criteria that to which of these c number of classes this feature vector x should belong?

Now, if I put a generalisation that I will always compute the maximum of these functional values. So out these c number of functional values which ever functional value is maximum, I will put x into the corresponding class. So, over there these functions are what are called as discriminant functions, discriminant function. So, this $g(x)$ is called discriminant function.

So, your function is same I will call it $g_1(x)$ when it is computed for class ω_1 or when it is computed for class ω_2 , $g_2(x)$ is the same function when this $g(x)$ is computed for class ω_2 or for action α_2 and so on. So, this function is called $g(x)$ is called discriminant function. And whichever class gives you maximum value discriminant function I put this object into that corresponding class. Now, let us see what will be the nature of this discriminant class

(Refer Slide Time: 46:50)



So, when I have c number of classes w_1 to w_c . So, there is c number of classes. So, I will have c number of the functional value of the discriminant function, so $g_i(x)$, for i varying from one to c . So, if I feel the decision rule in general that $g_i(x) > g_j(x)$ for all $j \neq i$. Then I decide that this x belongs to class w_i .

So, this means that whichever $g_i(x)$, whichever class i gives the value of this discriminant function $g_i(x)$ maximum, because this $g_i(x) > g_j(x)$ for all $j \neq i$ means this one is maximum. So, for whichever class this discriminant function gives the maximum value I put x into that corresponding class. So, what will be the nature of this discriminant function under different conditions.

If I go for minimum risk classifier, my risk is given by $R\left(\frac{a_i}{x}\right)$ for taking an action a_i .

And the distance to be minimum to take that action a_i , but in terms of discriminant $g_i(x)$ the value of the discriminant function has to be maximum. So, naturally if I want to relate this risk function with the discriminant function I have to make $g_i(x)$ which is negative of $R\left(\frac{a_i}{x}\right)$, because whenever this is maximum this is minimum because it is

negative. And whenever this $R\left(\frac{\alpha_i}{x}\right)$ is minimum by negating this $g_i(x)$ because maximum.

Similarly, for minimum error rate classification my condition was that this $R\left(\frac{\alpha_i}{x}\right) = 1 - P\left(\frac{\omega_i}{x}\right)$ that has to be minimum, that means $P\left(\frac{\omega_i}{x}\right)$ has to be maximum. So, I can simply equate $g_i(x)$ to $P\left(\frac{\omega_i}{x}\right)$. So, for minimum error rate classification $g_i(x)$ is simply $P\left(\frac{\omega_i}{x}\right)$. So, I can have the discriminant functions like this and for multiple number of classes for which ever class the value of the discriminant function is maximum I put x into that particular class. Now, you find that I can define $g_i(x)$ like this, but the choice of the discriminant function $g_i(x)$ is not unique.

(Refer Slide Time: 50:48)

$g_i(x)$
 $f(g_i(x))$
 $f(x) \rightarrow \text{monotonically increasing}$.

$$g_i(x) = \frac{p(x|w_i) \cdot p(w_i)}{\sum_{j=1}^c p(x|w_j) \cdot p(w_j)} \rightarrow p(x)$$
.

$$g_i(x) = p(x|w_i) \cdot p(w_i)$$

$$g_i(x) = \ln p(x|w_i) + \ln p(w_i)$$

The reason is if I take a function f say I have this $g_i(x)$. And I take a function which is function of $g_i(x)$. Now, if this function f which is function of $g_i(x)$ this is monotonically increasing. Then also this $f(g_i(x))$ will serve the same purpose as $g_i(x)$, because it is monotonically increasing. So, for whichever i , $g_i(x)$ is maximum for the same I , $f(g_i(x))$ will also be maximum because the function f is monotonically increasing. So, if I can

identify $g_i(x)$ then for any monotonically increasing function f , $f(g_i(x))$ that will also serve the same purpose of discriminant function. So this discriminant function that I said $g_i(x)$ it is not really unique. I can have various types of discriminant function.

So, only care I have to take is this function f that I have to choose that must be monotonically increasing function. And that gives us an advantage in the sense that if somehow I can identify $g_i(x)$ but $g_i(x)$ in its original form if it is not mathematically tractable. I can take another functional of this $g_i(x)$ which can be mathematically tractable, that can be used as a discriminating function.

Say coming to a very simple example. Say coming to this minimum error rate classification, we have said that this $g_i(x)$ is nothing but $P\left(\frac{\omega_i}{x}\right)$. If I expand this, it

simply becomes $\frac{P\left(\frac{x}{\omega_i}\right) \cdot P(\omega_i)}{\sum_{j=1}^c P\left(\frac{x}{\omega_j}\right) \cdot P(\omega_j)}$. Now, find that this term $\sum_{j=1}^c P\left(\frac{x}{\omega_j}\right) \cdot P(\omega_j)$, this is nothing but our $P(x)$.

This is nothing but our $P(x)$. And because this term is appearing in the denominator of all the discriminant functions for every value of i , this will be there in the denominator. So, I can simply remove this when I design my discriminant function. So, I can say that my discriminant function will simply be $g_i(x) = P\left(\frac{x}{\omega_i}\right) \cdot P(\omega_i)$. This will be my $g_i(x)$.

Now, you find that one I define $g_i(x)$ like this, there is a product term. And whenever I have a product, it is more difficult to implement as well as analyse rather than if I have a summation. Right? So, as I have in my original formation $g_i(x)$ like this. We know that logarithmic function is also monotonically increasing function. Right? So, instead of using this I can use log of this. And that can be my discriminant function. So, instead of using $g_i(f)$ as this, I can use $g_i(x)$, as this can also be my discriminant function. And here I have avoided this product by summation. So, it becomes mathematically more convenient. Right?

(Refer Slide Time: 55:29)

© CET
IIT-KGP

Two Category.

→ ω_1, ω_2

$g_1(x) > g_2(x) \rightarrow \omega_1$

$< g_2(x) \rightarrow \omega_2$

$\curvearrowleft g_1(x) - g_2(x) = 0.$

$\underbrace{g(x)}_{=} =$

$$g(x) = P(\omega_1|x) - P(\omega_2|x)$$

$$= \ln \frac{P(x|\omega_1)}{P(x|\omega_2)} + \ln \frac{P(\omega_1)}{P(\omega_2)}$$

Now, using this there is, if I go for two categories case that I have the classes ω_1 and ω_2 , that means that I have classes ω_1 and classes ω_2 . So, when I have these two classes that means I have two discriminant functions. One is $g_1(x)$ other one is $g_2(x)$. And my decision rule is if $g_1(x)$ is greater than $g_2(x)$, I decide in favour of ω_1 , if $g_1(x)$ is less than $g_2(x)$ I decide in favour of ω_2 .

So, what is the decision boundary between the classes ω_1, ω_2 ? Decision boundary simply where $g_1(x) = g_2(x)$. So, $g_1(x) - g_2(x) = 0$, that gives me the decision boundary.

So, if $g_1(x) - g_2(x) > 0$, I put it in class ω_1 , if it is less than zero, I put it in plus ω_2 . So, I can say that instead of taking these two discriminant functions particularly in a two category case, I can have a single discriminant function which is given by $g(x) = g_1(x) - g_2(x)$ and if this is equal to zero that gives me the decision boundary.

And from here by applying the same concept of logarithm you will find that this discriminating function can now be written as. And if I use $g_1(x)$ to be $P\left(\frac{\omega_1}{x}\right)$ and

$g_2(x)$ to be $P\left(\frac{\omega_2}{x}\right)$ then $g(x)$ becomes $P\left(\frac{\omega_1}{x}\right) - P\left(\frac{\omega_2}{x}\right)$. And using the concept

of logarithm and by expanding this in terms of a priori probability and class conditional

$$\text{probability, this will simply be written as } g(x) = \ln \frac{P(\omega_1/x)}{P(\omega_2/x)} + \ln \frac{P(\omega_1)}{P(\omega_2)} .$$

So, here I have these priori probabilities as well as class conditional probabilities. So, when the priori probability is same you find that this term become equal to zero. So, only decision is based on your class conditional probability. So, I will stop here today will continue with this discussion in the next class.

Thank you.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 7
Normal Density and Discriminant Function

Good morning. So, our today's topic of discussion will be normal density and discriminant function. So, in the last class, we have talked about different types of classifiers, which were actually derived from a Bayes classifier called Bayes minimum risk classifier. In Bayes minimum risk classifier, we said that I have to have a number of functional units, which will compute the risk involved for taking any particular action.

(Refer Slide Time: 00:58)

Normal Density & Discriminant Function

$$R(\alpha_i/x) = \sum_{j=1}^c \lambda(\alpha_i/\omega_j) P(\omega_j/x)$$

$$\approx 1 - P(\omega_i/x)$$

$$P(\omega_i/x) \rightarrow \text{Maximum.}$$

$$P(\omega_i/x) > P(\omega_j/x) \quad \forall j \neq i.$$

So, the risk function that we said was of this form $R(\alpha_i/x) = \sum_{j=1}^c \lambda(\alpha_i/\omega_j) P(\omega_j/x)$ that means given a feature vector x , the risk involved in taking an action α_i which was of this form, $R(\alpha_i/x) = \sum_{j=1}^c \lambda(\alpha_i/\omega_j) P(\omega_j/x)$. Take the summation for $j = 1$ to c where c is the total number of classes. So, for each action α_i , when I have say, P number of actions in my system, then there will be P number of such risk functions computed and that corresponding action alpha will be taken for which this risk function is minimum. So, that is what Bayes minimum risk classifier.

Then, we have said that we can derive another classifier, which is minimum error rate classifier from this Bayes minimum risk classifier. Where this risk function, we have said that it can be reduced in the form of $R(\omega_i/x) \approx 1 - P(\omega_i/x)$. So, when this is minimum, we have to put x in the corresponding class ω_i . As this has to be minimum, correspondingly what we get is this probability $P(\omega_i/x)$ that has to be maximum. So, this is nothing but the Bayes classification rule that we discussed about when I started talking about this decision theory that if $P(\omega_i/x) > P(\omega_j/x)$ for all $j \neq i$.

So, this is the posterior probability that given of feature vector x , what is the probability that x will belong to class ω_i . And this is what is the probability that x will belong to class ω_j . So, for that particular ω_i , where $P(\omega_i/x)$ is maximum, we have to put or we have to classify x as belonging to that particular class. Then we have talked about the discriminant functions.

(Refer Slide Time: 03:50)

Discriminant Function.

$\omega_i : i = 1, 2, \dots, C$

$g_i(x) \rightarrow$ Discriminant Function

$f(g_i(x))$

$f(\cdot) \rightarrow$ Monotonically Increasing .

$g_i(x) = P(\omega_i|x)$

$g_i(x) = \ln P(\omega_i|x)$

$= \underbrace{\ln p(x|\omega_i)}_{\text{Part 1}} + \underbrace{\ln P(\omega_i)}_{\text{Part 2}}$

$\frac{P(\omega_i|x)}{P(\omega_j|x)} = \frac{p(x|\omega_i) \cdot P(\omega_i)}{p(x|\omega_j) \cdot P(\omega_j)}$

So, there what we said is for every class i or every ω_i , for i varying from 1 to c , as we have considering a generalized case that as if we have c number of classes, so for every class, we define a function say $g_i(x)$. So, this $g_i(x)$ for a feature vector x , will be computed for every i th class and for whichever value of i , the $g_i(x)$ is maximum, we put x or classify x to that

particular class ω_i . So, this is what we are calling as discriminant function. We have also said that discriminant function $g_i(x)$ is not really unique.

So, if I can identify any function f , which is monotonically increasing, then the functional $f(g_i(x))$, where this function f this is a monotonically increasing function. So, whenever this f is monotonically increasing, then $f(g_i(x))$ that also serves the same purpose as discriminating function because for whichever value of i , $g_i(x)$ is maximum, it is for the same value of i , $f(g_i(x))$ will also be maximum. Then why do we put this functional f ? It is only for convenience of application, in different applications may be $g_i(x)$ itself is not very convenient, but if take a function of $g_i(x)$, then that will be more convenient for our application.

So, coming to this particular case of Bayes minimum risk classifier or minimum error rate classifier, while we have said that our $g_i(x)$ can be the posterior probability $P(\omega_i/x)$ because as per our minimum error rate classification, it was $1 - P(\omega_i/x)$. This is because whenever this $1 - P(\omega_i/x)$ has to be minimum, $P(\omega_i/x)$ has to be maximum. So, I have to put this x into the class ω_i for which $P(\omega_i/x)$ is maximum. So, this itself, I can take as my discriminant function $g_i(x)$.

Then, we said that logarithm being a monotonically increasing function, so I can also define

$$g_i(x) = \ln P(\omega_i/x) \text{ and because } P(\omega_i/x), \text{ this is nothing but } \frac{P(x/\omega_i) \cdot P(\omega_i)}{P(x)}. \text{ Where, we said}$$

that $P(x)$, we can remove because for every value of i , $P_i(x)$ will always appear in the denominator. So, that does not give you any discriminating power. So, I simply put this, $P(x/\omega_i) \cdot P(\omega_i)$, so there this function will be expanded to $\ln(P(x/\omega_i) + P(\omega_i))$. So, this becomes our discriminant function to be used for classification of an unknown sample x into one of the classes or one of ω_i .

So, naturally here you find because the probability density function is involved or the a priori probability is involved, so the structure of this Bayesian classifier will depend upon what kind of probability density that we are making use of. So, we can have various types of probability densities, we can have normal density, we can have Laplacian density, we can

have exponential density and so on, or Poisson density. So, depending upon what kind of density we make use of, for a particular application, what kind of probability density function is more appropriate according to that, our structure of the classifier will be different.

(Refer Slide Time: 09:27)

$$p(x) = \frac{1}{\sqrt{2\pi} \sigma} \exp\left[-\frac{1}{2} \left(\frac{x-\mu}{\sigma}\right)^2\right]$$

$$\mu = \int_{-\infty}^{\infty} x p(x) dx$$

$$\sigma^2 = E[(x-\mu)^2]$$

$$= \int_{-\infty}^{\infty} (x-\mu)^2 p(x) dx$$

$$\sim N(\mu, \sigma^2)$$

So, the most common probability density function, which is in use, is the normal or Gaussian density. So, this is the most common probability density function. So, this normal and

Gaussian density for a single variable is simply given $P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$. So, this

is the extension for normal or Gaussian density. Where the μ , μ is nothing but the expected

value of x or mean value of x , which is simply given as $\mu = \int_{-\infty}^{\infty} x p(x) dx$. Where the integral has

to be taken over the limit $-\infty$ to ∞ and σ^2 , which is the variance or σ , which is the standard deviation is simply given by the expectation value of $(x-\mu)^2$. Which is nothing but

$\sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 p(x) dx$. Again, you take the integral from $-\infty$ to ∞ . So, this is what the

expression of the normal or Gaussian density in a single variable case. And you find that this particular probability density function is specified only by two parameters. One is the mean value μ and the other one is the standard deviation σ or the variance σ^2 .

So, if I know only the mean value or the variance, then I know what this probability density function is. So, in short, this pdf, normal pdf is also written as $N(\mu, \sigma^2)$. So, this means that it

is normal density with the mean value of the variable as μ and variance of the signal as σ^2 , but we are talking about feature vectors that mean multiple numbers of components or multiple numbers of features. So a normal density of single variable is not much useful for us, but what is useful for us is multivariate normal. So, let us see what that multivariate normal density.

(Refer Slide Time: 12:56)

Multivariate Normal Density

$$p(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu)\right]$$

$$\mu = E[x] = \int x p(x) dx$$

$x \rightarrow d\text{-dimensional}$

$$\Sigma \rightarrow \text{Covariance matrix}$$

$$= E[(x-\mu)(x-\mu)^T]$$

$$= \underbrace{\int (x-\mu)(x-\mu)^T p(x) dx}_{d \times d}$$

(d×1) (1×d)
↓
d×d

So here instead of a single feature x , we have a feature vector and the feature vector is again, we use it in x , but instead of writing it as lower case x , we write it as capital X . So, this $P(X)$ in this case for a multivariate case will be simply given by

$$P(X) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (X-\mu)^T \Sigma^{-1} (X-\mu)\right].$$

So, this is the expression for the multivariate

probability density function, where X is the feature vector, a variable representing the feature vector. μ is the mean vector. So, μ is nothing but as before, the expected value of the feature vectors X , where X is a d dimensional vector because we have assumed that our feature vector is d dimensional that is d number of individual features are concatenated together to give you the feature vector.

So, X is a d dimensional feature vector. So, accordingly the mean vector μ vector will also be of dimension d . So, this $(2\pi)^{d/2}$, this d is nothing but the dimension of the feature vector and this Σ is what is called the covariance matrix. So as μ is the expectation value of X , so as before as in case of single variable case, we can write this as $\mu = E(X) = \int_{-\infty}^{\infty} X P(X) dX$ and

this covariance matrix because this is the expectation value of the covariance of the different components.

So this Σ covariance matrix, it is nothing but the expectation value of $(X-\mu)(X-\mu)^t$. So you note carefully that it is not $(X-\mu)^t(X-\mu)$ in which case you get a scalar because X is a d dimensional vector, μ is also a d dimensional vector. So, if make it $(X-\mu)^t(X-\mu)$ that becomes a scalar quantity or dot product of two vectors, rather it is $(X-\mu)(X-\mu)^t$. So, this becomes a d dimensional vector. So, $(X-\mu)$ is of dimension $dx1$ because it is a column vector and $(X-\mu)^t$ is a vector or a row vector of dimension $1xd$.

So, this is actually the outer product of two vectors. When I take the outer product of two vectors, the result is a dxd dimensional matrix. So, when I take the expectation value of $(X-\mu)(X-\mu)^t$, what I get is a dxd dimensional matrix. That is what is nothing but your covariance matrix or expectation value of this is becomes the covariance matrix. So, as before, this can also be written as in the integral form $\int_{-\infty}^{\infty} (X-\mu)(X-\mu)^t p(X)dX$. Take the integral of this over the limit $-\infty$ to ∞ . Then what I get is this covariance matrix Σ . Now, from here, if I try to compute what is the expected values of individual components?

(Refer Slide Time: 18:30)

$$\mu_i = E[x_i]$$

$$\sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)]$$

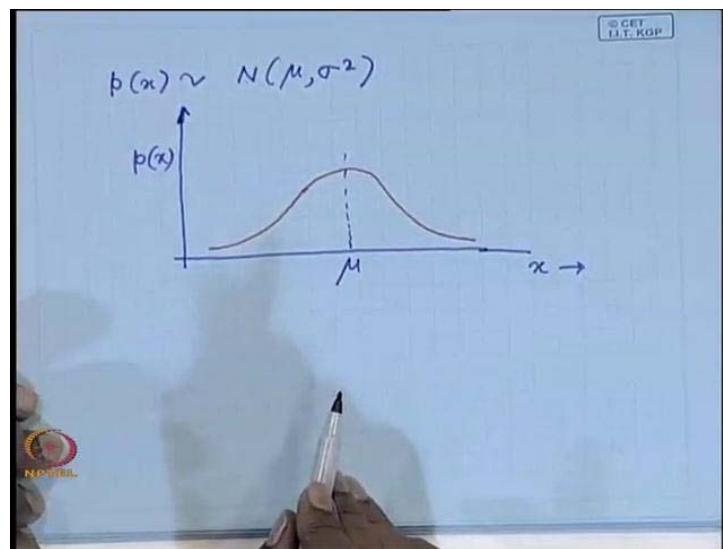
$$\sigma_{ii} = E[(x_i - \mu_i)^2] = \sigma_i^2$$

So the expected value of the i th component μ_i , which is nothing but the expected value of x_i which is the i th component of a feature vector capital X . Similarly, σ_{ii} which is the i th component in my covariance matrix Σ . σ_{ii} , this will simply become expected value of $(x_i - \mu_i)(x_i - \mu_i)$. Let us make it more general σ_{ij} ij^{th} component in my covariance matrix that will simply become expected value of $(x_i - \mu_i)(x_j - \mu_j)$.

So, you find that what the variance is when I take these two different components x_i and x_j that is what it gives you the covariance between the i^{th} component and j^{th} component. And when I compute this σ_{ij} which is $i j^{\text{th}}$ component in my covariance matrix that is given by this expression. So, obviously if I make $i = j$ that means σ_{ii} which are nothing but diagonal components, the components on the diagonal of the covariance matrix that simply becomes expected value, expectation value of $(x_i - \mu_i)^2$. And this is nothing but our σ_i^2 square which is the variance of the i^{th} component of feature vector.

So, when I talk about this covariance matrix, the diagonal elements in the covariance matrix actually give you the variance of the individual components of the feature vector. And the off elements, $i j^{\text{th}}$ elements actually give you the covariance when I consider the i^{th} component and the j^{th} component of the feature vector together. So, the covariance matrix is a more general form of the variance that we usually use in case of a single variable matrix. Now, what does this multivariate normal density or the multivariate Gaussian density actually tell you? Let us consider a case of single variable normal density.

(Refer Slide Time: 21:32)



So, I have p of x which is given by the normal density, this with mean value μ and the variances σ^2 . So, this simply means if I plot this normal density, plot of the normal density will be like this where the peak is, so I put x along the horizontal direction and $p(x)$ along the vertical direction.

So, when I have this single variable normal density, this simply tells us that if you take the samples from the same population, then how those samples are going to be placed? How those samples are actually distributed? So, this distribution shows that most of, maximum of the samples will be around this mean value μ . The other sample values will be distributed according to this. As you go away from the mean value, the population density will go on decreasing, and there is certain limit after which I can actually neglect the population density, which is given by say $\pm 2\sigma$ that is the twice of the standard deviation on this side as well as this side.

If I go beyond that, I can actually neglect the population density. So, this is what is meant by this normal density function. Now, it is interpretation in the multivariate density case. Before I go to this d dimensional case, let me just see what will happen in a 2 dimensional case, that is bivariate normal density that possibly you were asking in the last class yesterday.

(Refer Slide Time: 23:56)

$$p(x) = \frac{1}{(2\pi)^{1/2} |\Sigma|^{1/2}} \exp \left[-\frac{1}{2} \left\{ \left(\frac{x_1 - \mu_1}{\sigma_1} \right)^2 + \left(\frac{x_2 - \mu_2}{\sigma_2} \right)^2 \right\} \right]$$

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

So, suppose I talk about only two variables x and y or I talk about the normal density of a vector x . Where this vector x has only two components, x_1 and x_2 . So, this is a bivariate normal density. More generalization of this is multivariate normal density. Now, coming back the same expression, we will find that in such bivariate normal density, this expression as we have written in case of multivariate normal density that is this, this expression can be simplified for a bivariate normal density. It is not simplified.

If I simply expand this exponential term assuming that this X is on the two components x1 and x2, so by considering that, this p(X), when X is two dimensional will simply be written

$$\text{as } P(X) = \frac{1}{(2\pi)|\Sigma|^{1/2}} \exp \left[-\frac{1}{2} \left\{ \left(\frac{x_1 - \mu_1}{\sigma_1} \right)^2 + \left(\frac{x_2 - \mu_2}{\sigma_2} \right)^2 \right\} \right]. \text{ When I say } |\Sigma|, \text{ it is nothing but}$$

determinant of the covariance matrix. So, this $|\Sigma|$ means it is determinant of the covariance matrix because I cannot take square root of a matrix.

However, I can take square root of a determinant because determinant has a value, matrix does not have a value. Matrix has to be interpreted. So,

$$P(X) = \frac{1}{(2\pi)|\Sigma|^{1/2}} \exp \left[-\frac{1}{2} \left\{ \left(\frac{x_1 - \mu_1}{\sigma_1} \right)^2 + \left(\frac{x_2 - \mu_2}{\sigma_2} \right)^2 \right\} \right], \text{ this can be written in the form. Now, I}$$

have two components. One is x1 and other is x2. So, $\left(\frac{x_1 - \mu_1}{\sigma_1} \right)^2 + \left(\frac{x_2 - \mu_2}{\sigma_2} \right)^2$, this is a

simplified expression because I had assumed that the components x1 and x2 were statistically independent.

So, that is why, I could have a simplified expression only in terms of σ_1 and σ_2 only the diagonal elements in a covariance matrix. It is only because my assumption that x1 and x2 are statistically independent. So, σ_{12} or σ_{21} will be equal to 0 because x1 and x2 are statistically independent. If they are not statistically independent, then in this exponential term, I will also have terms corresponding to σ_{12} and σ_{21} .

So, for explanation, let me simplify. Let me have a simplified assumption that as if x1 and x2, they are statistically independent.

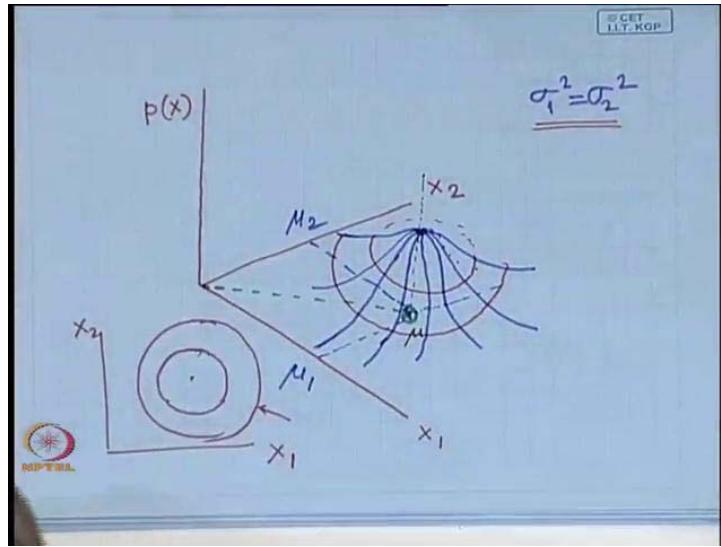
However, x1 component has a mean value of μ_1 and x2 component has a mean value of μ_2 ,

that means my mean vector μ is nothing but $\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$. And the covariance matrix Σ is

nothing but $\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$. So, this is my covariance matrix and this is my mean vector. So,

what is the physical interpretation of this particular bivariate normal density function? Let us draw this two dimensional space or three dimensional space because x1 and x2 are the axis of the sample points and the density function is p(X), which is the third dimensions.

(Refer Slide Time: 28:45)



So, let me draw this third dimensional space. So, I put x_1 along this direction, x_2 along this direction and $p(x)$ along this direction. Suppose μ_1 is or the mean vector, μ is somewhere over here. So, if μ is at this position, so that means the components, this is the location of μ_1 and this is the location of μ_2 . Is that okay? Now, find that if I break this expression, this term

$$\exp\left[-\frac{1}{2}\left\{\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right\}\right], \text{ this is actually product of two exponential terms,}$$

$$\exp\left[-\frac{1}{2}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2\right] \cdot \exp\left[-\frac{1}{2}\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right].$$

So, when I consider forgetting this part, $\exp\left[-\frac{1}{2}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2\right]$, this is actually a normal density

along x_1 direction with a mean value at μ_1 and the variance σ_1^2 . When I consider only this component $\exp\left[-\frac{1}{2}\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right]$, this becomes a normal density along x_2 direction with mean value μ_2 and standard deviation σ_2 . So, this overall exponential term is actually a product of two normal density functions two Gaussian density functions.

So, if I plot this in these two dimensions, I will have something like this. I will have a peak of the density around this mean value μ , mean position μ and the densities will be, I will have Gaussian along this. I will also have Gaussian along this. And if it is a single Gaussian, if I

assume that my σ_1 , let me take a simplified case that $\sigma_1^2 = \sigma_2^2$ that means the variance along x_1 and the variance along x_2 they are same.

So, the kind of shape that I will have over here is if I have a single normal density, so that single normal density profile, you rotate around the vertical axis.

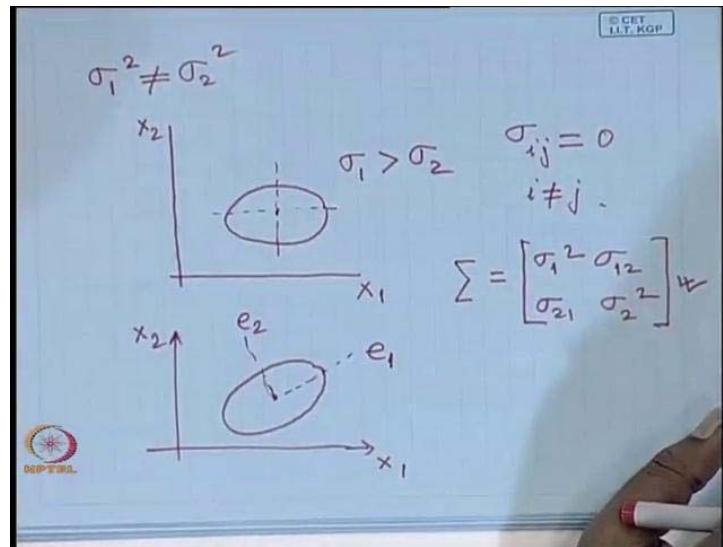
So, whatever surface it will trace; that surface will be the surface of this probability density function $p(X)$. So, if I plot over this thing, the surface will be something like this. It will go on this side. It will come like this, of this form. Now, given this surface of this bivariate probability density function, if I now try to chase the loci of points of constant density that means all those values of x for which $p(X)$ is constant. So, those loci will be nothing but circles, something like this because along this value of $p(X)$ will be constant.

So, if I take the footprint of this loci on this $x_1 x_2$ plane, so on $x_1 x_2$ plane, I will have a number of concentric circles like this. Where these circles, each of these circles represents the loci of points having constant density. As I move inside the circle, move towards the centre of the circle, these loci of the density, value of the density will increase. As I move away from the centre, the value of density function will go on decreasing. So, that indicates that along the circle, I have more the probability of occurrence of the points, which are drawn from a single population arbitrarily or drawing of one sample does not depend on the drawing of another sample.

What does it mean? When I simply take up the first object, I do not make use of any information of what was the previous object that I have taken. So, just blindly you go on taking samples from a population and then the points will be distributed in this form.

So, this is a simple case that I have taken when $\sigma_1^2 = \sigma_2^2$. That means the variance along x_1 dimension and the variance along x_2 dimensions are the same. What happens if the variances are different? So in that case, this circle will become ellipse.

(Refer Slide Time: 35:21)



So, this circle that I have shown when I have say $\sigma_1^2 \neq \sigma_2^2$, so one will be greater than the other. So, when $\sigma_1^2 = \sigma_2^2$, I had the footprints of the loci of the point of constant density as circles. Now, the loci of points of constant density will be ellipses. How these ellipses will be formed? Something like this. So, this is a case when $\sigma_1 > \sigma_2$, that means the spread of points along x_1 axis is more than the spread of points along x_2 axis.

If $\sigma_2 > \sigma_1$ that is spread of points along x_2 axis is more than spread of points along x_1 axis, then I will have the major axis aligned towards x_2 direction and the minor axis along x_1 direction.

So, this is the nature of the loci of the points of constant density when my assumption was that different components of the feature vectors are statistically independent that means σ_{ij} was equal to 0, for $i \neq j$. What happens if $\sigma_{ij} \neq 0$? That means the samples are not or the components of the feature vector are not statistically independent. So, over there, I will have my covariance matrix Σ . In this case, co variance matrix is a diagonal matrix where only diagonal elements are non-zero elements. All the off diagonal elements are 0s. If the components are not statistically independent, then even the off diagonal elements will also be non-zero.

So I will have the situation that $\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$. So, if it so happens here, you find that the major axis and the minor axis of these ellipses they are aligned along x_1 dimension and x_2

dimension. In this case, when the components are not statistically independent, then the directions of the major axis and minor axis of the ellipse will be given by Eigen vectors of this covariance matrix. The lengths of the major axis and minor axis will correspond to the Eigen values of the covariance matrix.

So, instead of having a footprint of the point, of the loci of points of constant density like this, the footprints will be something like this. Again, it will be centered around μ , but the directions of major axis and minor axis will be given by the Eigen vectors of this covariance matrix say e_1 and e_2 . So, this is my x_1 direction, this is my x_2 direction and as I said that these directions actually tell you how the points are distributed. I will have maximum density around mean. As I go along this direction, the density will go on reducing. As I move along this direction also, the density will also go on reducing, but here, the rate of decrease of the density function will be more, here the rate of decrease of the function will be less. Is that okay?

Now, what happens in case of multivariate normal density? I have taken the example of bivariate normal density, but because still up to two variables, I can visualize. For the moment, consider the case of two variables and three variables. So, I will have three axes, one corresponding to x_1 , one corresponding to x_2 and other corresponding to x_3 . I have to have a fourth axis which corresponds to $p(x)$, the probability density function. I cannot draw it on a two dimensional plane, up to two dimensional, I can easily draw, when it goes to three dimensional plane, still I can draw, but some difficulty. The moment it goes beyond three, I cannot draw it on a two dimensional plane, but I can think of how the nature can be. So, when it becomes more than two variables that is for multivariate of normal density for a multi variable multivariate Gaussian density, again if I draw the sample of points from that multi-dimensional space, the points will form clouds. Here also points are forming point clouds or clusters of points, where the maximum density will be around μ_1 or around μ .

As I move away from μ , the density of the points goes on reducing. When I go for this multivariate normal density, then also I can say that those multi-dimensional points will form point clouds in a multi-dimensional space in such a way that again around μ , the density will be maximum. As we move away from μ , the density will go on reducing and they will be clustered. Here, this clustering is in the form of an ellipse. In a multi-dimensional space, it will be an ellipsoid. So, the points will clustered in an ellipsoid. So, again in three dimensional spaces, if I simply rotate this ellipse along a particular axis, I will have volume.

So, the points will be clustered in that volume. At that center of the volume, the ellipsoidal volume, the density of the point will be maximum. As I move away from ellipsoid from the volume, the density will go on reducing again following that normal density function or Gaussian density function. So, over here as in this case, the loci of points of constant density forms ellipses, in multi-dimensional space, the loci of points of constant density they form ellipsoids.

(Refer Slide Time: 43:05)

This ellipsoid will have a quadratic form, which is given by $(X-\mu)^T \Sigma^{-1} (X-\mu)$. So, this is the quadratic form of loci of points of constant density. And the principle axis of this ellipsoid as in this case, the principle axis of the ellipses are given by the Eigen vectors of the covariance matrix. So, in the same way, the principle axis of the ellipsoids in this case will be given by Eigen vectors of the covariance matrix. And the lengths of the axis will be given by the corresponding Eigen vectors. Is that okay?

And this term, the distance function actually, this locus is nothing but the value of the distance values from the centroid.

So, μ gives you the center or the centroid and x is the points lying on the peripheral. This term actually gives you the distance of x from μ . So, distance function r^2 , I can define which is given by $r^2 = (X-\mu)^T \Sigma^{-1} (X-\mu)$. In previous case, I can define $d^2 = \sum_{i=1}^d (x_i - \mu_i)^2$,

take the summation over $i = 1$, instead of calling d , let me call it say d , $d^2 = \sum_{i=1}^d (x_i - \mu_i)^2$ because we are assuming vectors d dimensional.

So, this $l^2 = \sum_{i=1}^d (x_i - \mu)^2$. Take the summation from $i = 1$ to d . This is what is our Euclidean distance. This distance function if I define as $(x_i - \mu)^T \Sigma^{-1} (x_i - \mu)$, this is what is called Mahalanobis distance, named after great statistician P C Mahalanobis. So, this is what is called Mahalanobis distance.

Student: Sir, it is a square or its root should be...

No square, it is a quadratic term $(x_i - \mu)^T \Sigma^{-1} (x_i - \mu)$.

Student: Sir, I am talking about is r^2 or r .

The distance is r . The expression is given by this. That is quite obvious.

So now, let us see that how this can be utilized. So, so far what I have discussed is about the probability density functions, univariate case, bivariate case, multi variate case. I have not gone beyond that, but we started with that how these probability functions actually influence the structure of the decision surface. We started with that because $g_i(x) = \ln P(x/\omega_i) + \ln P(\omega_i)$, probability density and the conditional density function. So, we started with this. So, our purpose is to find out $g_i(x)$, which is the discriminant function.

(Refer Slide Time: 47:48)

© CET
IIT-KGP

$$g_i(x) = \ln p(x|\omega_i) + \ln P(\omega_i)$$

$$p(x|\omega_i) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp \left[-\frac{1}{2} \left\{ (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right\} \right]$$

$$g_i(x) = -\frac{1}{2} [(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i)] - \frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln |\Sigma_i| + \ln P(\omega_i)$$

So as given $g_i(x) = \ln P(x/\omega_i) + \ln P(\omega_i)$. Now, the expression is $P(x/\omega_i)$. What is said is,

$P(x/\omega_i)$ means we are taking the samples from class ω_i and finding out the sphere density

function. So expression will remain the same. The μ and Σ will be replaced by μ_i, Σ_i . That means the mean vector for the samples taken from class ω_i , covariance matrix computed from the samples taken from class ω_i that is all, expression will remain the same.

So the expression for the probability density function if I write $P(x/\omega_i)$, it is nothing but

remains the same expression $P(x/\omega_i) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} \{(X-\mu_i)' \Sigma_i^{-1} (X-\mu_i)\}\right]$. So, you find

that Σ is replaced by Σ_i . Now, μ has to be replaced by μ_i . This μ_i I do not put it as i^{th} component of mu, rather this is the mean vector of class ω_i . So,

$P(x/\omega_i) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} \{(X-\mu_i)' \Sigma_i^{-1} (X-\mu_i)\}\right]$, this is my probability density function or

class conditional probability density function.

So, by using this particular probability density function, multivariate probability density function, now I can define $g_i(x) = \ln P(x/\omega_i) + \ln P(\omega_i)$. So, it will simply become, this is an exponential. So, if we take the logarithm, it will be simply this. So, it will simply become

$$g_i(x) = -\frac{1}{2} \{(X-\mu_i)' \Sigma_i^{-1} (X-\mu_i)\} - \frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln(|\Sigma_i|) + \ln P(\omega_i).$$

So, this is the expression for discriminant function corresponding to class ω_i . So, from here, you will find that this is actually a quadratic expression. Is it not? I have $\{(X-\mu_i)' (X-\mu_i)\}$ with Σ_i^{-1} sandwiched between these 2 terms.

So, actually this is a quadratic equation. So, Bayes discriminator is actually a quadratic discriminator in general or when I want to compute the decision surfaces between two classes between ω_i and ω_j , the decision surface will actually be a quadratic surface. It is not a linear surface. So, this classifier can take care of linearly non separable classes. However, for specific cases, this can be converted to linear classifier. So, we will talk about those things in the next class. Let us stop here today.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 8
Normal Density and Discriminant Function (Cond.)

Good morning, so we are going to start, continue with our discussion on normal density and discriminant function which we started in our last class.

(Refer Slide Time: 00:40)

Normal Density &
Discriminant Function

$$g_i(x) = \ln p(x|\omega_i) + \ln P(\omega_i)$$

$$p(x|\omega_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp \left[-\frac{1}{2} (x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i) \right]$$

$$\boxed{g_i(x) = -\frac{1}{2} (x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma_i| + \ln P(\omega_i)}$$

So, then in the last class we said that the discriminant function is given by $g_i(X) = \ln P(X/\omega_i) + \ln P(\omega_i)$, where ω_i is the i th class. If we replace this probability density function $P(X/\omega_i)$ by a normal density having the covariance matrix as Σ_i and the mean vector as μ_i that is $P(X/\omega_i)$.

If we write in this form $P(X/\omega_i) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[-\frac{1}{2} \{(X-\mu_i)^T \Sigma_i^{-1} (X-\mu_i)\} \right]$, so this is a multivariate normal density function where the feature vector X are actually d dimensional vectors. So, taking this form of probability density function, normal probability density

function, the discriminate function

$$g_i(X) = -\frac{1}{2} \left\{ (X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i) \right\} - \frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln(|\Sigma_i|) + \ln P(\omega_i).$$

So, this becomes the discriminate function when we assume that the probability density function, the associated probability density function is a multivariate normal distribution, normal density function. We have said that depending upon this covariance matrix Σ_i , we can have different types of discriminate function, so we will see those cases that if this covariance matrix Σ_i takes different form. Then what are the forms of the discriminate function or what is the architecture of the decision boundary between two different classes that vary depending upon the covariance matrix Σ_i for the different classes. So, the first one that we will consider is the simplest form.

(Refer Slide Time: 04:08)

g_i(x) = -\frac{\|x - \mu_i\|^2}{2\sigma^2} + \ln P(\omega_i)
 - A circled term $\|x - \mu_i\|^2$ is annotated with 'Distance Square'.
 - A small logo 'DEPTHE' is visible at the bottom left of the whiteboard."/>

So, we will put that as case one when Σ_i that is the covariance matrix is of the form $\sigma^2 I$ where this I is nothing but an identity matrix that is, it is a diagonal matrix where only the diagonal elements are 1's and the rest of the elements are 0's and so on. So, obviously you find that when this covariance matrix Σ_i is of the form $\sigma^2 I$, where I is the identity matrix. That means we have only diagonal elements which keeps only the variance of the individual components. The variance of all the components are same which is equal to σ^2 and the off diagonal elements are equal to 0 which indicates that $\sigma_{ij} = 0$.

That means the components, different components are statistically independent, so when a covariance matrix becomes like this, so is nothing but the case when the feature of vector or the points belonging to the same class, they actually cluster in a hyper spherical space and the shape and size of all different clusters, they are same because the variance system. So, all the different clusters belong to different classes, the points belong to the hyper spherical space and they are of the same size.

So, that is what is meant by this sort of covariance matrix, now here we find that if the covariance matrix is like this, then the determinant of Σ_i that is for every class. It is nothing but σ^{2d} , where this σ is the standard deviation of independent components. It is simply because I have a $d \times d$ dimensional identity matrix okay, which is multiplied by σ^2 , so I have a diagonal matrix where only the diagonal elements are equal to σ^2 and the off diagonal element are all 0's.

I have d number of diagonal elements, so the value of the determinant is nothing but product of all the diagonal elements which is nothing but σ^{2d} . It is also quite obvious that inverse of the covariance matrix that is Σ_i^{-1} will be nothing but $\frac{1}{\sigma^2}$ into identity matrix I , so when I have in this case one that covariance matrix is same for all different classes. In this case having this simplified form this discriminant function $g_i(x)$, you can compute it will simply

$$\text{become } g_i(x) = -\frac{\|x - \mu_i\|^2}{2\sigma^2} + \ln P(\omega_i).$$

So, what is this term, this is nothing but the squared distance of sample x from the mean vector μ_i . So, in the situation that if all the classes are equally probable that is $\ln P(\omega_i)$ is there for all the classes, then this term becomes irrelevant in the discriminant function. So, my discriminant function is simply this equation and this being the distance square from the mean of the vectors, you find that it becomes a minimum distance classifier because this term will be maximum, when this term is minimum because it is negative and $\|x - \mu_i\|^2$ being the squared distance from the mean vector and all the classes having same variance σ^2 you find that this simply becomes the minimum distance classifier.

So, given a feature vector x , you simply compute its distance from the mean vectors from all different classes and whichever distance is the minimum that is the nearest mean of the classes. This x will be classified to that particular class, now if we, and here you find that even in this $P(\omega_i)$ is present.

That means, the a priori probability of different classes are different they are not same then for a sample point which is equidistant from all the cluster means, all the class mean. The decision will actually be biased by this term $\ln P(\omega_i)$.

(Refer Slide Time: 10:22)

$$\begin{aligned}
 g_i(x) &= -\frac{1}{2} (x - \mu_i)^t \Sigma_i^{-1} (x - \mu_i) + \ln P(\omega_i) \\
 &= -\frac{1}{2\sigma^2} [x^t x - 2\mu_i^t x + \mu_i^t \mu_i] + \ln P(\omega_i) \\
 g_i(x) &= -\frac{1}{2\sigma^2} [-2\mu_i^t x + \mu_i^t \mu_i] + \ln P(\omega_i) \\
 &= \boxed{\mu_i^t x + w_{i0}} \rightarrow \text{Linear Machine} \\
 w_i &= \frac{1}{\sigma^2} \mu_i \\
 w_{i0} &= -\frac{1}{2\sigma^2} \mu_i^t \mu_i + \ln P(\omega_i)
 \end{aligned}$$

Now, if I go for expansion of the term, discriminant function, then you find that this $g_i(X)$,

the discriminant function is simply of the form $g_i(X) = -\frac{1}{2} \{(X - \mu_i)^t \Sigma_i^{-1} (X - \mu_i)\} + \ln P(\omega_i)$

Why I am writing like this, is in this expression you find that $-\frac{d}{2} \ln(2\pi)$, this is independent of class this is a constant term. So, I can simply ignore this from the expression of the discriminant function because this term anyway is not going to influence our decision because for all the classes for the discriminant functions of all the classes this term is going to be present.

Similarly, this $|\Sigma_i|$ that is covariance matrix in this particular case which is nothing but σ^{2d} , this is also same for all the classes. So, this also becomes irrelevant in the discriminant function, because for all the discriminant function the same term will be present and the value of this term will be same.

So, we will discriminate among the classes is the first term and the last term, so we have this simplified discriminate expression when $\Sigma_i = \sigma^2 I$. So, if I expand this term you find that this expression can be written as $g_i(X) = -\frac{1}{2\sigma^2} [X^t X - 2\mu_i^t X + \mu_i^t \mu_i] + \ln P(\omega_i)$. This is nothing but expansion of this term where Σ_i^{-1} , we have said is $\frac{1}{\sigma^2} I$, so that is why here I get $\frac{1}{2\sigma^2}$.

And if I simply expand this I get this particular expansion, now again here we will find that the first term within the bracket that is this $X^t X$. This is also independent of class for all values of i in all the discriminant functions $g_i(X)$, the same term will be present and this σ^2 being same for all the classes. This term $\frac{X^t X}{2\sigma^2}$ that also becomes redundant term it is not really deciding to which class this X will belong, so I can ignore this term from the discriminant function.

So, by ignoring this discriminate function becomes a simplified form like this $g_i(X) = \frac{1}{\sigma^2} [\mu_i^t X - \mu_i^t \mu_i] + \ln P(\omega_i)$. and this you find that I can write this in the form

$W_i^t X + W_{i0}$ where you find, you will find that this W_i is simply $W_i = \frac{1}{\sigma^2} \mu_i$ and

$$W_{i0} = -\frac{1}{2\sigma^2} \mu_i^t \mu_i + \ln P(\omega_i).$$

So, this particular expression can be written in this form $W_i^t X + W_{i0}$ where this $W_i = \frac{1}{\sigma^2} \mu_i$

and $W_{i0} = -\frac{1}{2\sigma^2} \mu_i^t \mu_i + \ln P(\omega_i)$. So, if you look at this expression that finally, how we get

the discriminant function expression for $g_i(X)$, which is $g_i(X) = W_i^t X + W_{i0}$ this is nothing but a linear equation. So, in this simplified case, where all the covariance matrices covariance

matrices for all the classes are same and is of the form $\sigma^2 I$, the discriminant function simply becomes a linear equation.

So, this is, and if we have a classifier for the discriminant function which classify based on this linear equation that is called a linear machine. So, a classifier which uses the linear discriminant functions to decide about the class belongingness of an unknown vector that is called a linear machine, so this will form a linear machine, so that is about the discriminant function of an individual class or i^{th} class. Now, if I want to find out the decision boundary between two different classes, say I have two class, class ω_i and class ω_j .

(Refer Slide Time: 17:14)

$\omega_i \leftarrow \omega_j \rightarrow$ Two classes

$$g_i(x) - g_j(x) = 0 \rightarrow \text{Eqn of Decision boundary.}$$

$$g_i(x) = w_i^t x + w_{i0}$$

$$g_j(x) = w_j^t x + w_{j0}$$

$$(w_i - w_j)^t x + w_{i0} - w_{j0} = 0$$

$$\Rightarrow \frac{1}{\sigma^2} (\mu_i - \mu_j)^t x - \frac{\mu_i^t \mu_i}{2\sigma^2} + \ln P(\omega_i)$$

$$+ \frac{\mu_j^t \mu_j}{2\sigma^2} - \ln P(\omega_j) = 0$$

MPTEL

So I have two different classes. So what will be the nature of the decision boundary between two these two classes the i^{th} class and the j^{th} class. So, here we find that if I want to find out that decision boundary, on the decision boundary $g_i(X)$ for a featured vector lying on the decision boundary $g_i(X) = g_j(X)$. Because, if that is the case, I mean that is the situation which differentiates between two different classes if $g_i(X) > g_j(X)$, X will belong to class i , if $g_j(X) > g_i(X)$ then X belong to class j .

The situation that $g_j(X) = g_i(X)$ tells you the decision boundary, that means the equation of the decision boundary is simply given by $g_i(X) - g_j(X) = 0$ this is the equation of decision boundary. And we have just seen that $g_i(X)$ is nothing but $g_i(X) = W_i^t X + W_{i0}$ it is the linear

equation. Similarly, $g_j(X)$ will be $g_j(X) = W_j^t X + W_{j0}$, so in this expression if I replace $g_i(X) = W_i^t X + W_{i0}$ and $g_j(X) = W_j^t X + W_{j0}$. So, I simply get the equation as $(W_i - W_j)^t X + W_{i0} - W_{j0} = 0$.

Here, if I replace W_i by the value of W_i which is nothing but $W_i = \frac{1}{\sigma^2} \mu_i$ and W_{i0} by the value of W_{i0} which is nothing but $W_{i0} = -\frac{1}{2\sigma^2} \mu_i^t \mu_i + \ln P(\omega_i)$. Similarly, for W_j and W_{j0} , so by replacing these values of W_i W_j and W_{i0} W_{j0} . This expression will simply become

$$\frac{1}{\sigma^2} (\mu_i - \mu_j)^t X - \frac{\mu_i^t \mu_i}{2\sigma^2} + \ln P(\omega_i) + \frac{\mu_j^t \mu_j}{2\sigma^2} - \ln P(\omega_j) = 0.$$

(Refer Slide Time: 21:45)

The image shows a handwritten derivation of the decision boundary equation. It starts with the expression $(\mu_i - \mu_j)^t X - \frac{1}{2} (\mu_i^t \mu_i - \mu_j^t \mu_j) + \sigma^2 \ln \frac{P(\omega_i)}{P(\omega_j)}$. This is set equal to zero. Then, it is rearranged into a form involving the mean vector difference $(\mu_i - \mu_j)$ and a term involving the covariance matrix $\frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(\omega_i)}{P(\omega_j)}$. Finally, it is shown that this is equivalent to $W^t (x - x_0) = 0$, where $W = \mu_i - \mu_j$ and $x_0 = \frac{1}{2} (\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(\omega_i)}{P(\omega_j)} (\mu_i - \mu_j)$.

And this same expression if I go on simplifying, this will lead to a form $(\mu_i - \mu_j)^t X - \frac{1}{2} (\mu_i^t \mu_i - \mu_j^t \mu_j) + \sigma^2 \ln \frac{P(\omega_i)}{P(\omega_j)} = 0$, it is simply the same expression. Where we have grouped together $\frac{\mu_i^t \mu_i}{2\sigma^2}$ and $\frac{\mu_j^t \mu_j}{2\sigma^2}$ and grouped together, this $\ln P(\omega_i)$ and $\ln P(\omega_j)$. So,

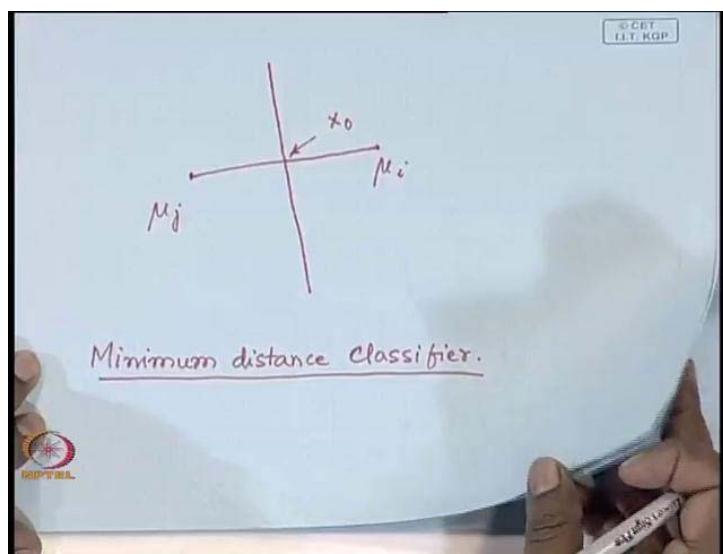
I get this particular equation and this expression can further be simplified so I can write it in

$$\text{the form } (\mu_i - \mu_j)^t \left[X - \left\{ \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(\omega_i)}{P(\omega_j)} \cdot (\mu_i - \mu_j) \right\} \right] = 0.$$

I can convert this expression into this form where you find that this $(\mu_i - \mu_j)^t$ that has been taken out of all the terms. So, when I can write in this form this is simply of the form $W^t(X - X_0) = 0$ where this term W is nothing but $(\mu_i - \mu_j)$. Where, μ_i is the mean vector of the i^{th} class and μ_j is the mean vector of the j^{th} class and X_0 is simply $\frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(\omega_i)}{P(\omega_j)} \cdot (\mu_i - \mu_j)$.

So, what information we get from this here, you find that this is $W^t(X - X_0) = 0$, that is the decision boundary between the i^{th} class and j^{th} class, where this W is given by $(\mu_i - \mu_j)$. That means it is a vector drawn from the j^{th} mean vector μ_j to the i^{th} mean vector μ_i or the line joining μ_i and μ_j . So, that is what is W and as the decision surface is $W^t(X - X_0) = 0$, so the decision surface is orthogonal to the vector joining μ_i and μ_j .

(Refer Slide Time: 26:14)

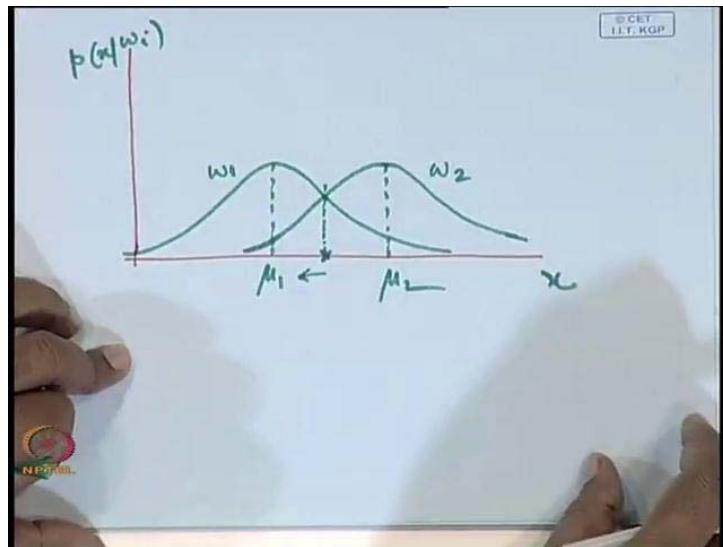


So, the situation is simply like this that if two vector positions are say I have μ_i over here and μ_j over here and this is the line joining μ_i and μ_j . The decision surface will be orthogonal to μ_i and μ_j and because it is the linear equation, so the decision surface is a hyperplane. A hyperplane which is orthogonal to the line joining μ_i and μ_j . And if this term is equal to 0 that is when the a priori probabilities is $P(\omega_i) = P(\omega_j)$, then this term is equal to 0.

This orthogonal plane it passes to point X_0 which is on the line μ_i and μ_j , so this is point X_0 if this term is 0 in case that probability is $P(\omega_i) = P(\omega_j)$ then $X_0 = \frac{1}{2}(\mu_i - \mu_j)$. So, this X_0 in that case is midway between μ_i and μ_j and I have this hyperplane which in this case is nothing but orthogonal bisector of the line joining μ_i and μ_j and because it's the orthogonal bisector.

Again, I came back to the same situation that are kind of classifier that I have is the minimum distance classifier because if I have a have an orthogonal bisector of the line joining μ_i and μ_j . Then for all the points which are falling on the side of μ_i , its distance from mu i will be less than its distance from μ_j . Similarly, for all the points which are falling on the side of μ_j , its distance from μ_j will be less than its distance from μ_i . And we put the sample X to that particular class from which its distance is minimum. So, when it is an orthogonal bisector or the a priori probabilities $P(\omega_i)$ and $P(\omega_j)$, they are same then effectively what I get is a minimum distance classifier.

(Refer Slide Time: 29:12)



So, effectively it says that if the probability density functions, the normal probability functions like this, again plotting in univariate case, so in this side I put X and this is $P(x/\omega_1)$ and suppose this is for ω_1 and this is for ω_2 . So, if the probability is $P(\omega_1)$ and $P(\omega_2)$ are same and this is what is μ_1 and this is what is μ_2 . So, if the probabilities are same my decision will be decision surface will be at the middle if the probabilities are more if $P(\omega_1)$ is more than $P(\omega_2)$ a priori probability.

Then, the decision surface will be pushed towards $P(\omega_2)$ that is away from μ_1 if $P(\omega_1) > P(\omega_2)$, then this decision surface will be pushed away from μ_1 , if $P(\omega_1) < P(\omega_2)$. Then the decision surface will be pulled towards μ_1 , so in the simplified case when all the covariance matrices are same and they are of the form $\sigma^2 I$, I get such simplified linear classifiers.

And the decision boundary between the two classes will be orthogonal line joining the mean corresponding vectors and if the two classes are equally probable that is if the a priori probabilities are same. Then the decision surface is nothing but a orthogonal bisector of the line joining μ_1 and μ_2 , in which case the classifier effectively becomes a minimum distance classifier.

(Refer Slide Time: 31:36)

Case II

$$\Sigma_i = \sum$$

© GET
I.I.T. KGP

$$g_i(x) = -\frac{1}{2} \underbrace{(x - \mu_i)^t}_{\text{Mahalanobis distance}} \underbrace{\Sigma_i^{-1} (x - \mu_i)}_{\text{Squared}} - \frac{d}{2} \ln 2\pi - \frac{1}{2} \cancel{\ln |\Sigma_i|} + \ln P(\omega_i)$$

The whiteboard also features the logo of IIT Kharagpur (IITKGP) and a small circular emblem with the text "NPTEL".

Let us take another case that is case 2, here I assume again that all the classes have the same covariance matrix that $\Sigma_i = \Sigma$. In a first case we have said that this Σ is of the form $\sigma^2 I$, but now we are not putting that restriction the Σ or the covariance matrix is arbitrary, but all the classes samples belonging to all the classes have the same covariance matrix though it is arbitrary. So, in the previous case we said that the samples are clustered into hyper spherical spaces of equal size. In this case, the sample space is clustered into hyper ellipsoidal spaces of same shape and same size.

It will be same shape and same size because the covariance matrix system and as the covariance matrix is arbitrary, so in general it will be a hyper ellipsoidal space. So, the samples belonging to different classes, they cluster into ellipsoidal space of same shape and same size, is that okay? So that is the second case. So, here again we find that out of this expression of the discriminant function the general expression, in the general expression we

$$\text{had } g_i(X) = -\frac{1}{2} \left\{ (X - \mu_i)' \Sigma_i^{-1} (X - \mu_i) \right\} - \frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln(|\Sigma_i|) + \ln P(\omega_i).$$

So, this is the general expression of the discriminant function, now we are saying that Σ_i or the covariance matrix is same for all the classes so if it is same for all the classes. Then obviously, this term obviously this is a constant term, so this does not give you any discriminating power and again this term this is same for all the classes because this Σ_i is same for all the classes. So, this also becomes irrelevant, so we are left with

$$g_i(X) = -\frac{1}{2} \left\{ (X - \mu_i)' \Sigma_i^{-1} (X - \mu_i) \right\} + \ln P(\omega_i).$$

And here, again we find that if this $\ln P(\omega_i)$, this also becomes irrelevant that is if all the classes are equally probable, then my discriminant function simply is $g_i(X) = -\frac{1}{2} \left\{ (X - \mu_i)' \Sigma_i^{-1} (X - \mu_i) \right\}$. And we said that this is nothing but a distance function which is squared Mahalanobi's distance, squared. So, effectively the classifier again becomes the minimum distance classifier because this being a distance function if $g_i(X)$ has to be maximum.

Then, this has to be minimum and X will be put into that particular class for which this term is minimum, so again it becomes a minimum distance classifier, but the distance that we have

to compute in this case is not the Euclidean distance, but the Mahalanobi's distance. So, considering the fact that we will compute Mahalanobi's distance again, we have a minimum distance classifier, so as we have done before if we simply expand this term.

(Refer Slide Time: 36:41)

$$g_i(x) = w_i^t x + w_{i0} \rightarrow \text{Linear Equation.}$$

$$w_i = \Sigma^{-1} \mu_i$$

$$w_{i0} = -\frac{1}{2} \mu_i^t \Sigma^{-1} \mu_i + \ln P(w_i)$$

Linear Machine.

Then, again we will find that the discriminant function $g_i(X)$ will again be of the form

$g_i(X) = W_i^t X + W_{i0}$. In the earlier case, this W_i had a very simple form that is $W_i = \frac{1}{\sigma^2} \mu_i$,

now this W_i will be of the form

$W_i = \Sigma^{-1} \mu_i$. Earlier this Σ^{-1} was replaced by $\frac{1}{\sigma^2}$ in the first case, so now it becomes

$W_i = \Sigma^{-1} \mu_i$ and W_{i0} is simply $W_{i0} = -\frac{1}{2} \mu_i^t \Sigma^{-1} \mu_i + \ln P(\omega_i)$.

So, again we find that this term the nature of the discriminant function this is again a linear equation because this is $g_i(X) = W_i^t X + W_{i0}$, so this is again a linear equation. So, our classifier making use of such linear discriminant functions that again becomes a linear machine now using these linear discriminant functions if I want to find out as I have done before.

(Refer Slide Time: 38:43)

Handwritten notes on a whiteboard:

$$\omega_i \text{ & } \omega_j$$

$$W^t(X - X_o) = 0$$

$$W = \Sigma^{-1}(\mu_i - \mu_j)$$

$$X_o = \frac{1}{2}(\mu_i + \mu_j) - \frac{\ln [P(\omega_i)/P(\omega_j)]}{(\mu_i - \mu_j)^t \Sigma^{-1}(\mu_i - \mu_j)} \cdot (\mu_i - \mu_j)$$

Decision Surface.

Diagram illustrating the decision surface:

A diagram shows two points μ_i and μ_j connected by a vector. A point X_o is marked on the line segment between μ_i and μ_j . A line passes through μ_i and X_o , labeled "Decision Surface".

That we want to find out the nature of or the structure of the decision surface between two classes ω_i and ω_j , so again we will follow the same procedure that $g_i(X) - g_j(X) = 0$.

That is the equation of the decision surface separating the two classes ω_i and ω_j and if I follow the same steps again. Here, we will find that the equation will come in the form $W^t(X - X_o) = 0$, where this W is given by $W = \Sigma^{-1}(\mu_i - \mu_j)$ and X_o is given by

$$X_o = \frac{1}{2}(\mu_i + \mu_j) - \frac{\ln \left[P(\omega_i) / P(\omega_j) \right]}{(\mu_i - \mu_j)^t \Sigma^{-1}(\mu_i - \mu_j)} (\mu_i - \mu_j)$$

So, the expression becomes more or less same, I get similar expression $W^t(X - X_o) = 0$, where this $W = \Sigma^{-1}(\mu_i - \mu_j)$. If you remember earlier case our W was simply $W = (\mu_i - \mu_j)$ and $\mu_i - \mu_j$ is the line or the vector from μ_j to μ_i . So, W had the same direction as the vector $\mu_i - \mu_j$ but $\Sigma^{-1}(\mu_i - \mu_j)$ is not in general in the direction of $\mu_i - \mu_j$. So, as a result this vector W which is $W = \Sigma^{-1}(\mu_i - \mu_j)$ this is not in general in the same direction of the vector μ_j to μ_i .

So, we find that from this expression though my decision surface is orthogonal to vector W in the previous case W was in the direction of the vector $\mu_i - \mu_j$. So, the decision surface was

orthogonal to the line joining μ_j and μ_i , and in this case because W is not in the direction of line joining μ_j and μ_i . So, the decision surface is not in general orthogonal to the line joining μ_j and μ_i but the decision surface passes through the point X_o where the equation expression

$$\text{for } X_o \text{ is given by } X_o = \frac{1}{2}(\mu_i + \mu_j) - \frac{\ln \left[\frac{P(\omega_i)}{P(\omega_j)} \right]}{(\mathbf{X} - \mu_i)^t \Sigma_i^{-1} (\mathbf{X} - \mu_i)} (\mu_i - \mu_j) .$$

So, here again you find that if $P(\omega_i)$ and $P(\omega_j)$ they are same then this term becomes 0,

when this term becomes 0, then $X_o = \frac{1}{2}(\mu_i + \mu_j)$ that means it is halfway between μ_j and μ_i

. So, when the classes are equally probable the decision surface or the hyper plane passes through the midpoint of the line joining μ_j and μ_i , but it is in general not a orthogonal bisector, it is a bisector, but not orthogonal.

So, effectively the kind of situation that we will have is, if I have these two means μ_j and μ_i this is the line joining μ_j and μ_i and the decision surface will be something like this. This is point X_o and this is your decision surface, so we still have the linear classifiers, the decision surface two classes are still hyper planes, but this is not a orthogonal bisector or this is not orthogonal to the line joining μ_j and μ_i .

(Refer Slide Time: 43:43)

© CET
IIT RGPV

Case III

$\Sigma_i \rightarrow \text{arbitrary}$

$g_i(x) = x^t A_i x + B_i^t x + C_{i0} \rightarrow \text{Quadratic}$

$A_i = -\frac{1}{2} \Sigma_i^{-1}$

$B_i = \Sigma_i^{-1} \mu_i$

$C_{i0} = -\frac{1}{2} \mu_i^t \Sigma_i^{-1} \mu_i - \frac{1}{2} \ln |\Sigma_i| + \ln P(\omega_i)$



Now, let us take the third case which is the most general one that is the covariance matrix Σ_i is totally arbitrary that means different classes have different covariance matrices. In case two we considered that different classes have arbitrary covariance matrices, but covariance matrices of all the classes are same though arbitrary. The first case was the simplest case where we assume that the components, different components are statistically independent and individual components are the same variance. So, this is the most general case where the different classes will have the different covariance matrices.

So, here I do not have any other option I have to keep all the terms, so $g_i(X)$ if I keep all the terms. You will find that it will be of the form $g_i(X) = X^t A_i X + B_i^t X + C_{io}$, where this $A_i = -\frac{1}{2} \Sigma_i^{-1}$, $B_i = \Sigma_i^{-1} \mu_i$ and $C_{io} = -\frac{1}{2} \mu_i^t \Sigma_i^{-1} \mu_i - \frac{1}{2} \ln(|\Sigma_i|) + \ln P(\omega_i)$. So, you find that in the earlier cases where we could remove the quadratic terms that is term involved in $X^t X$ or $X^t A_i X$ we could have removed term because $X^t X$ does not give you any discriminating power, but over here this is $X^t A_i X$, where A_i depends upon the value of Σ_i that is the covariance matrix and different classes have different covariance matrices.

So, this term $X^t A_i X$, now becomes class dependent because it is class dependent it really contribute to decide whether X will belong to class ω_i or X will belong to class ω_j .

So, I cannot remove any of the terms and this expression that I have it is nothing but a quadratic expression, so giving you a quadratic classifier in the earlier two cases we had linear classifiers in this case we get a quadratic classifier. And using this quadratic classifier if I want to find out the decision surface between two classes ω_i and ω_j , the decision surface will be a hyperquadrics. So, it is a quadratic surface in a multi-dimensional space, so these are the different cases that we can have assuming multi variate normal density function for the samples belonging to different classes.

(Refer Slide Time: 48:00)

© CET
I.I.T. KGP

$$\omega_1 \rightarrow \begin{pmatrix} 2 \\ 6 \end{pmatrix}, \begin{pmatrix} 3 \\ 4 \end{pmatrix}, \begin{pmatrix} 3 \\ 8 \end{pmatrix}, \begin{pmatrix} 4 \\ 6 \end{pmatrix}$$

$$\omega_2 \rightarrow \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \begin{pmatrix} 3 \\ -4 \end{pmatrix}, \begin{pmatrix} 5 \\ -2 \end{pmatrix}$$

$$P(\omega_1) = P(\omega_2) = 0.5$$

→ Decision boundary between ω_1 & ω_2

$$\mu_1 = \begin{pmatrix} 3 \\ 6 \end{pmatrix} \quad \mu_2 = \begin{pmatrix} 3 \\ -2 \end{pmatrix}$$

$$\Sigma_1, \Sigma_2 \quad \sum (x - \mu)(x - \mu)^T$$

Let us see an example, suppose we have points belonging to two classes one is ω_1 , class ω_1 and class ω_2 , the training samples which are provided for these two classes are like this. For class ω_1 it is $\begin{pmatrix} 2 \\ 6 \end{pmatrix}, \begin{pmatrix} 3 \\ 4 \end{pmatrix}, \begin{pmatrix} 3 \\ 8 \end{pmatrix}, \begin{pmatrix} 4 \\ 6 \end{pmatrix}$, so these are the training samples which are given for class ω_1 and the training samples for ω_2 are given as $\begin{pmatrix} 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \begin{pmatrix} 3 \\ -4 \end{pmatrix}, \begin{pmatrix} 5 \\ -2 \end{pmatrix}$. So, naturally we are considering a two dimensional space and as I said that the examples typically I will take in two dimension because I can draw it on a plane paper.

So this is $\begin{pmatrix} 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \begin{pmatrix} 3 \\ -4 \end{pmatrix}, \begin{pmatrix} 5 \\ -2 \end{pmatrix}$, so these are the training samples for the belonging to two classes ω_1 and ω_2 which is given. It is also said that the a priori probabilities, that is $P(\omega_1)$ which is equal to $P(\omega_2)$ and naturally this is equal to 0.5. So, what we have to do is we have to find out that the decision boundary between class ω_1 and class ω_2 , so to find out the decision boundary I have to find out the discriminant function between two classes ω_1 and ω_2 .

Then, subtract that equate that to 0 or in general because I have to compute Σ_i that is the covariance matrix for independent classes and I do not know what is the nature of Σ_i . So, in

general I can make use of this expression and go ahead with it assuming that my factors will be quadratic in case the surface is not quadratic all the quadratic terms will get cancelled.

So, first what I have to do is I have to find out the mean vector which is μ_1 and μ_1 is nothing but I will take all these vectors and find out their means. So, this μ_1 in this case will come out to be $\begin{pmatrix} 3 \\ 6 \end{pmatrix}$, you can verify this, similarly for class two μ_2 will come out to be $\begin{pmatrix} 3 \\ -2 \end{pmatrix}$, so that is the mean vector for class 2. We also have to find out the covariance Σ_1 , the covariance matrix Σ_2 . You know how to compute the covariance matrix once I get the mean vector. Then what I have to do is I have to find out $\Sigma(X-\mu)(X-\mu)^t$ take the summation over all the vectors, and then normalize by the number of vectors not taken.

(Refer Slide Time: 51:56)

$$\Sigma_1 = \begin{pmatrix} 1/2 & 0 \\ 0 & 2 \end{pmatrix} \quad \Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

$$\Sigma_1^{-1} = \begin{pmatrix} 2 & 0 \\ 0 & 1/2 \end{pmatrix} \quad \Sigma_2^{-1} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

$$g_1(x) = x^t A_1 x + B_1^t x + C_1$$

$$x_2 = 3.514 - 1.12 x_1 + 0.1875 x_1^2$$

So, that gives you the covariance matrices Σ_1 , Σ_2 , so if you do it that way you will find that

Σ_1 comes out to be $\Sigma_1 = \begin{pmatrix} 1/2 & 0 \\ 0 & 2 \end{pmatrix}$. So, this is the covariance matrix of class 1 and Σ_2 , that is the covariance matrix of class two comes out to be $\Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$.

So, we find that the covariance matrices of the two classes are different and as the covariance matrices of the two classes are different the classifier is not a linear classifier. It will be

quadratic classifier; from here I can compute Σ_1^{-1} which is nothing but $\Sigma_1^{-1} = \begin{pmatrix} 2 & 0 \\ 0 & 1/2 \end{pmatrix}$.

From here, you can compute Σ_2^{-1} which is nothing but $\Sigma_2^{-1} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$, so these are the

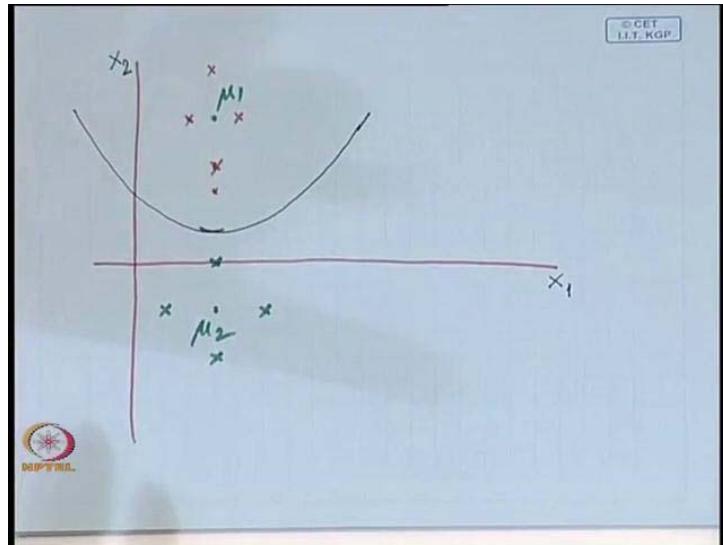
inverses of the covariance matrices. Now, once I get this I find out $g_1(X)$ which is of the form $g_1(X) = X^t A_1 X + B_1^t X + C_{10} X$ and the expressions. We have given over here in a quadratic classifier, where the value of A_1 will be nothing but $A_1 = -\frac{1}{2} \Sigma_1^{-1}$, value of B_1 will be $B_1 = \Sigma_1^{-1} \mu_1$ and value of C_{10} will be

$$C_{10} = -\frac{1}{2} \mu_1^t \Sigma_1^{-1} \mu_1 - \frac{1}{2} \ln(\|\Sigma_1\|) + \ln P(\omega_1).$$

Now, in this case $P(\omega_1)$ is same as $P(\omega_2)$, so this term can be neglected even if I keep it on both sides they will get cancelled. So, I do not have to really compute this, so if you compute all these terms and put in this expression. And similarly, you find out $g_2(X)$ following similar expressions and then equate $g_1(X) = g_2(X)$, or $g_1(X) - g_2(X) = 0$.

Then, we will find that finally the decision surface I am not going to the detailed calculation, let us look a simple one the final decision surface will come out to be $x_2 = 3.514 - 1.2x_1 + 0.1875x_1^2$. So, you find that the decision surface is linear quadratic, in this case it is the curve because I am we are in two dimensional space. So, the decision boundary is really a quadratic curve, now if I want to find out how this quadratic curve looks like I will simply plot these points on paper and find out how the surfaces look like.

(Refer Slide Time: 55:50)



So, I have points $\begin{pmatrix} 2 \\ 6 \end{pmatrix}$, then I have points $\begin{pmatrix} 3 \\ 4 \end{pmatrix}$, I have $\begin{pmatrix} 3 \\ 8 \end{pmatrix}$ is here, this is one point, this is one point and I have $\begin{pmatrix} 4 \\ 6 \end{pmatrix}$ that is this point. So, these are the points which belong to class ω_1 , similarly for ω_2 I have $\begin{pmatrix} 3 \\ 0 \end{pmatrix}$, this is a point belonging to class ω_2 , I have $\begin{pmatrix} 1 \\ -2 \end{pmatrix}$, $\begin{pmatrix} 3 \\ -4 \end{pmatrix}$ is over here. I have $\begin{pmatrix} 5 \\ -2 \end{pmatrix}$ that is this one, so these are the point which belongs to class ω_2 , so all the reds belong to class ω_1 and the greens belong to class ω_2 .

μ_1 is somewhere over here, this is μ_1 and μ_2 is over here and if you plot this quadratic curve into this space, so this quadratic curve be something like this where I am assuming that I have two components one is x_1 other one is x_2 . So, we find that we really get a quadratic curve separating the two classes ω_1 and ω_2 , so I will stop here today and we will continue with other classifiers next day.

Thank you.

Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 9
Bayes Decision Theory – Binary Features

(Refer Slide Time: 00:41)

Good morning, so we are going to discuss today the Bayes decision theory for binary features. So far what we have discussed is assuming that the feature vectors are distributed following some normal distribution of the form

$$P(x/\omega_i) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}\{(X-\mu_i)' \Sigma^{-1} (X-\mu_i)\}\right].$$

So, this is the expression for a

multivariate normal distribution. And we have seen that, in this expression where Σ_i represents the covariance matrix, that for different conditions of covariance matrices, we can have different types of classifier.

Say the first case we have discussed is, if the covariance matrix for every class i is of the form, $\sigma^2 I$, where both this covariance matrix is of dimension $d \times d$. Where our feature vectors are of dimension d and this identity matrix is also of dimension $d \times d$. So, because this is identity matrix so this simply says that covariance matrix is a diagonal matrix, where every diagonal element is of value σ^2 . That means every component have the same variance whereas off diagonal elements are equal to 0 so because off diagonal elements are 0's so the components, different components of the feature vectors are statistically independent.

So, in such cases we have seen that, the classifier is nothing but a linear classifier or when we talked about the discriminant function, the discriminant function for, function for individual classes, they are also linear functions. And because they are linear functions so the classifier which employs this linear functions to classify an unknown feature vector, that is a linear machine. And in particular, if we want to find out the decision boundary between two different classes say i^{th} class, ω_i and the j^{th} class, ω_j the decision boundary between these two different classes is a hyper plane, which is orthogonal to the line joining μ_i and μ_j , where μ_i and μ_j are the centers of the classes ω_i and ω_j .

So, effectively I have a situation something like this, that, if I have μ_i somewhere over here, which is the mean of the class ω_i and μ_j is somewhere over here. Then the decision surface is orthogonal, is a hyper plane, which is orthogonal to the line joining μ_i and μ_j . And if the apriori probabilities $P(\omega_i) = P(\omega_j)$ then this decision boundary or the decision surface becomes an orthogonal bisector of the line joining μ_i and μ_j . So, this was our simplest case.

(Refer Slide Time: 04:20)

$$\sum_i = \sum$$

$$\mu_i \quad \mu_j$$

$$P(\omega_i) = P(\omega_j)$$

$$\sum_i \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

In the second case, we have seen that if $\sum_i = \sum$, that is every class or the samples belonging to every class have the same covariance matrix, but otherwise the covariance matrix is arbitrary, unlike in the first case where, covariance matrix has its specific form like this. In the second case the covariance matrices are arbitrary, but every class have the same covariance matrix, which ideally means that, the points belonging to the same class or the points belonging to different classes, they are clustered in hyper ellipsoidal

spaces of same shape and same size. And in such case we have seen that, the discriminating function that we get for different classes they are also linear.

So, in both the cases in the first case, as well in the second case the discriminating functions becomes linear so the classifier is nothing but a linear machine. However, there is some difference between the decision surfaces, that we get between two different classes ω_i and ω_j . In the first case, the decision surface was orthogonal to the line joining μ_i and μ_j , in the second case the decision surface is not in general orthogonal to the line joining μ_i and μ_j .

So, here again if μ_i and μ_j say this is μ_i and this is μ_j , which are the centers of the two classes ω_i and ω_j then the decision surface between these two classes will be something like this. In the previous case, it was orthogonal to the line joining μ_i and μ_j , in this case in general it is not orthogonal. However, the decision surface is a hyper plane or it represents the linear equation and here again, if the a priori probability is $P(\omega_i) = P(\omega_j)$. Then this decision surface though it is not orthogonal to the line joining μ_i and μ_j , but it will pass through the point which is midway between the points μ_i and μ_j .

And the third case we have said that, the covariance matrices of different classes are totally arbitrary. So for i^{th} class I will have one covariance matrix, for j^{th} class I will have another covariance matrix. So, that effectively means that the clusters or the points, vectors belonging to different classes they are clustered into hyper ellipsoidal spaces. In this, in the first, second case the hyper ellipsoidal spaces were of same shape and same size. In this case, the points belonging to different classes will also form hyper ellipsoidal spaces, but these hyper ellipsoidal spaces may not have same shape or may not have same size. So, they will have different shapes as well as different sizes, but the points belonging to the same class they, form an hyper ellipsoidal spaces.

However, in all these three different cases that we have discussed, we have assumed that the feature vector x is continuous or the individual components of the feature vector x . Because our feature vector X is, nothing but a d dimensional vector having the components x_1, x_2 up to x_d . So, it is a d dimensional feature vector so in all this three different cases our basic assumptions was that, the feature vectors are continuous or in otherwise, individual components are also continuous. That effectively means that if I consider a d dimensional feature space then the feature vector can be represented by any

point, is represented by any point within that d dimensional feature space. I do not have any specific set of points, from which the feature vectors are drawn.

However, in most of the practical applications and particularly in these days as we are walking with digital computers, all the data that we get are digital data. And the moment we get digital data, the vectors that we generate are no more continuous rather, they are discrete vectors or every component of the feature vector every x_i will have a discrete values. Discrete values mean, it will assume one of a set of specific values so instead of the continuous variable it becomes a discrete variable.

So, when it is a discrete variable, in that case in all our previous lectures wherever we have talked about integration, the integration is to be replaced by summation. And the summation has to be carried out, over the discrete space. So, we will take a specific case of this discrete feature vectors. So, let us consider a case where, so we will have a feature vector X which will be discrete.

(Refer Slide Time: 10:11)

The notes are handwritten in blue ink on a whiteboard:

- $X \rightarrow \text{discrete}$
- $\text{Two class problem} \rightarrow w_1 \text{ & } w_2$
- $\text{Binary Feature Vectors}$
- $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \quad x_i = 0/1$
- $p_i = P_r[x_i=1 | w_1]$
- $q_i = P_r[x_i=1 | w_2]$
- $p_i > q_i \Rightarrow x_i \text{ is more likely to have value 1 if } X \in w_1$

In the bottom left corner, there is a logo for IIT Kharagpur (IITKGP) and the text "NPTEL". In the top right corner, there is a small box containing "© CET I.I.T. KGP".

And we consider a specific case of a two class problem and the feature vectors, we will assume to be binary feature vectors. Binary feature vector means, every component of the feature vector will assume binary value either 0 or 1. So, when I have this feature vector X, which is given by x_1, x_2 up to x_d . Again we assume that, we have d dimensional feature vectors. Every component of the feature vector x_1, x_2 or x_3 can assume either a value equal to 0 or a value equal to 1.

So, that means that, if a feature vector is taken from a particular class it will say whether a particular feature is present or it is absent. So, when I have this sort of binary feature vectors so we will also assume the different components of this feature vectors are conditionally independent. To have something similar to statistical independence of different components in continuous feature vectors. So, here also we will assume that different components of the feature vectors are conditionally independent.

So, every feature value in this feature vector x_i , every feature value can have a value either 0 or 1. So, it will say whether the feature is present or the feature is absent and correspondingly, the probabilities will be something like this. So, every feature component will be represented by a probability value where, the probability is something like this, that I will represent p_i to represent the probability of the i^{th} component. So, the p_i will be equal to probability that, component x_i is equal to 1 given that, the true state of nature or the true class is class ω_1 . We are considering a two class problem.

So, we have classes ω_1 and ω_2 . So, p_i represents probability that $x_i = 1$ given that, the true state of nature is ω_1 similarly, q_i is probability that, x_i the same component is equal to 1, given the true state of nature is ω_2 . So, given this type of probability measures, it simply means that if I have a situation that p_i is greater than q_i . In such cases, it simply means that i^{th} component x_i is more likely to have value 1 if, x belongs to class ω_1 .

Because, it is the probability of assuming a value equal to 1, when the true state of the nature of the two classes ω_1 or when the true state of the nature of two classes ω_2 . So, if p_i is greater than q_i that simply means that, if the sample is taken from class ω_1 . Then it is more likely that the i^{th} component x_1 will have value equal to 1 or x_i will have a value equal to 1, more frequently if the vector is taken from class ω_1 .

And if it is taken from class ω_2 then it is less frequent that the i^{th} component x_i will have a value equal to 1.

So, when I have this kind of situation now, let us see that what are the cases in which these kind of feature vectors are more useful. Say for example, if we want to find out the health of a plant say, power plant, if I want to determine the health of a power plant. Then what, we normally do is there are number of sensors which are used to monitor different parameters of the plant. And after monitoring those different parameters, you decide whether the plant is okay or there is some danger in the plant.

And when you sensor, when you monitor the sensor outputs you just see that, whether the sensor output is above a threshold level or below a threshold level. So, if it is above a threshold level, we set a value equal to 1, if it is below a threshold level we set a value equal to 0. Let me take a more obvious example, when I go to the market to purchase oranges, you must have noticed that even in our tech market you usually get two types of oranges. One type of oranges, which are produced at Nagpur and one type of orange which are coming from Darjeeling. Have you noticed any difference in appearance between these types of oranges.

Student: Color is different

Color is different, if it is from Darjeeling color is more attractive. It is really orangey color, it is more yellowish whereas, oranges from Nagpur they are more greenish and if it becomes quite old, it becomes more reddish. If you look at the surface texture, the oranges which are coming from Darjeeling, they are smooth whereas, the oranges which are taken from Nagpur, they are rough. So, if simply based on these two features I want to determine, I want to have an automated machine which will simply classify, tell me that whether these are Darjeeling orange or a Nagpur orange. So, it will try to take the decision based on the color in the simplest case or it will try to take the decision based on the feature, the texture feature.

So, if I keep some of the feature vectors like, whether the color is yellowish, answer will be either yes or no, whether it is greenish either it will be, answer will be either yes or no whether it is smooth, answer will be either yes or no. However, when I get an orange from Darjeeling and I take an orange from Nagpur, there is no guarantee that all the oranges from Darjeeling, will always have smooth texture or will always have orangey or yellowish color. Or if take oranges from Nagpur there is no guarantee that, I will always have greenish color, Nagpur even may produce some oranges which will have yellowish color or which will have smooth textures.

So, there is always a finite probability that an orange produced at Nagpur will have yellowish color. So, it is not necessary if that color to be yellowish, I put that as a binary feature, for all the oranges coming from Nagpur that binary value will always be equal to 0, it is not guaranteed, for some of them I may get values which are equal to 1, but that is less frequent than, the value equal to 1 when the oranges are taken from Darjeeling.

So, simply over here if that feature I put as the i th feature x_i then p_i will be more frequently equal to 1, x_i will be more frequently equal to 1, if this ω_i is Darjeeling. And this will be less

frequently equal to 1, if ω_2 is from Nagpur so coming over here if p_i is greater than q_i . So, it simply explains this particular situation, that is for oranges coming from Darjeeling, I will have more number of oranges having yellowish color than, the number of oranges I get with yellowish color from the Nagpur oranges. So, this is a typical situation like this.

And what does conditional independence mean, the texture and the color they are independent. I mean if the texture is rough, that does not necessarily mean that the color will be greenish or if the texture is smooth, that does not necessarily mean that the color will be yellowish. So, coming to the plant if I take two features say, I want to monitor the health of a boiler. If I take two features, one is pressure inside the boiler and temperature inside the boiler.

They are not independent because if temperature increases the pressure will increase. I mean just from the basic laws of gas so they are not independent they are dependent. So, when I try to select the features I should select in such a way that the features are independent because that solves many of the mathematical problems, I do not have to look for complicated mathematics. So, if I assume that the features are conditionally independent.

(Refer Slide Time: 21:17)

$$p(x|\omega_1) = \prod_{i=1}^d p_i^{x_i} (1-p_i)^{1-x_i}$$

$$p(x|\omega_2) = \prod_{i=1}^d q_i^{x_i} (1-q_i)^{1-x_i}$$

Likelihood Ratio.

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} = \prod_{i=1}^d \left(\frac{p_i}{q_i}\right)^{x_i} \left(\frac{1-p_i}{1-q_i}\right)^{1-x_i}$$

Then, the same probability function I can write as $P(x/\omega_1)$, where x is the feature vector, which has d number of binary valued feature components. So, this $P(x/\omega_1)$, I can simply

write as $P(x/\omega_1) = \prod_{i=1}^d p_i^{x_i} (1-p_i)^{1-x_i}$, p_i is the probability that $x_i = 1$, given the true state of nature is ω_1 .

But it also has a finite probability that, it may have a value equal to 0 so I have to consider that as well.

So, it is $p_i^{x_i} (1-p_i)^{1-x_i}$ obviously, if $x_i = 1$, $(1-x_i) = 0$, if $x_i = 0$, $(1-x_i) = 1$, and because the components are conditionally independent so the overall probability will be product of independent probability values. So, this I have to take for $i = 1$ to d , as I have d number of components so this is the class conditional probability of a feature vector x , if the state of nature is ω_1 .

So, in the same manner I can write, $P(x/\omega_2)$, that is class conditional probability if the feature vector belongs to class ω_2 is nothing but $P(x/\omega_2) = \prod_{i=1}^d q_i^{x_i} (1-q_i)^{1-x_i}$. Now, the probability of $x_i = 1$, when the true state of nature is ω_2 is q_i . So, I will have $q_i^{x_i} (1-q_i)^{1-x_i}$, take the product from $i = 1$ to d .

Student: Sir where d is dimension of the...

D is the dimension of the feature vector, so I have d number of components in the feature vector. So, these are the two class conditional probability values now, from here I can find

out, what is called likelihood ratio. So, the likelihood ratio is given by $\frac{P(x/\omega_1)}{P(x/\omega_2)}$ which is

nothing but if I simply multiply these two, it becomes $\frac{P(x/\omega_1)}{P(x/\omega_2)} = \prod_{i=1}^d \left(\frac{p_i}{q_i} \right)^{x_i} \left(\frac{1-p_i}{1-q_i} \right)^{1-x_i}$. Take

the product where, i varying from 1 to d so this is what is the likelihood ratio, right.

Student: Sir x_i could be a vector or x_i is a single component?

x_i is the single component, it is a scalar, x_i is the single component, the capital X is the vector, having d number of components so but the value of i will vary from 1 to d .

(Refer Slide Time: 25:56)

$$g(x) = \ln \frac{P(x|\omega_1)}{P(x|\omega_2)} + \ln \frac{P(\omega_1)}{P(\omega_2)}$$

$$\Rightarrow g(x) = \sum_{i=1}^d \left[x_i \ln \frac{p_i}{q_i} + (1-x_i) \ln \frac{1-p_i}{1-q_i} \right] + \ln \frac{P(\omega_1)}{P(\omega_2)}$$

Now, you know that the decision surface or our decision function between the two classes

given, two classes is given by $g(X) = \ln \frac{P(x/\omega_1)}{P(x/\omega_2)} + \ln \frac{P(\omega_1)}{P(\omega_2)}$. This we derived earlier now, if

we put this $\frac{P(x/\omega_1)}{P(x/\omega_2)} = \prod_{i=1}^d \left(\frac{p_i}{q_i} \right)^{x_i} \left(\frac{1-p_i}{1-q_i} \right)^{(1-x_i)}$, which is equal to this expression,

$$g(X) = \sum_{i=1}^d \left[x_i \ln \frac{p_i}{q_i} + (1-x_i) \ln \frac{1-p_i}{1-q_i} \right], \text{ if I put this expression into this function.}$$

So, it will give us, because we are taking logarithm so the product term over here will be converted to sum of logarithmic functions. So, what I simply get is

$$g(X) = \sum_{i=1}^d \left[x_i \ln \frac{p_i}{q_i} + (1-x_i) \ln \frac{1-p_i}{1-q_i} \right] + \ln \frac{P(\omega_1)}{P(\omega_2)}. \text{ So, this product will become sum in the}$$

logarithmic term. So, this is the decision function and for a two category case we have already seen that, if $g(X) > 0$ then our decision was that X belongs to class ω_1 . If $g(X) < 0$ then our decision was x belongs to class ω_2 . If $g(X) = 0$, that actually tells us that what is the decision boundary between the classes ω_1 and ω_2 .

Now, if you notice that this equation is a linear equation, isn't it? Because this simply says, the linear combinations of different components x_i of the feature vector x , I do not have any x_i square term or x_i cube term. So, the equation is a linear equation and this linear equation

can simply be written in the form, if I just rearrange this particular linear equation I can write it in the form.

(Refer Slide Time: 29:31)

$$g(x) = \sum_{i=1}^d w_i x_i + w_0$$

$$w_i = \ln \frac{p_i(1-q_i)}{q_i(1-p_i)} ; i=1 \dots d$$

$$w_0 = \sum_{i=1}^d \ln \frac{1-p_i}{1-q_i} + \ln \frac{P(\omega_1)}{P(\omega_2)}$$

$$g(x) > 0 \Rightarrow x \in \omega_1$$

$$g(x) < 0 \Rightarrow x \in \omega_2$$

$g(X) = \sum_{i=1}^d w_i x_i + w_0$, do not confuse w with ω . So, it becomes sum of $w_i x_i + w_0$ where i

varies from 1 to d , because I have d number of components in the feature vector. So, you find that this is, nothing but a linear combination of the different components, of the feature vector x_i plus a threshold term, which is w_0 . And this w_i 's for different values of i , this represents a weight vector and this is, nothing but a dot product or inner product of the weight vector with the feature vector.

So, when I write it like this, over here w_i = that is quite obvious. Because, you find that this

becomes $x_i \ln \frac{p_i}{q_i} - x_i \ln \frac{1-p_i}{1-q_i} + \ln \frac{1-p_i}{1-q_i}$. So, this $1-q_i$ term that goes to the numerator and $1-p_i$

term that comes to the denominator.

So, it simply becomes $\ln \frac{p_i(1-q_i)}{q_i(1-p_i)}$, where, i varies from 1 to d . Because, I have d number of

components in the weight vector as well and w_0 , that is the threshold is given by

$$w_0 = \sum_{i=1}^d \left[\ln \frac{1-p_i}{1-q_i} \right] + \ln \frac{P(\omega_1)}{P(\omega_2)} . \text{ Is that okay?}$$

Now if you analyze this, as I said that our decision will be that if, $g(X) > 0$ then we decide that x belongs to class ω_1 . If $g(X) < 0$ then we decide that x belongs to class ω_2 , if this is equal to 0 that is a boundary case. So, if you analyze this expression or what do we get, what does these different components of w , that is w_i , that effectively tell us. You find that x_i that is the i th component of the feature vector x is the binary value feature vector. It can have a value equal to 0, it can have a value equal to 1.

Now, if it has a value equal to 1 then the contribution of the term $w_i x_i$ for that particular component x_i , to this function $g(X)$ is, nothing but equal to the magnitude of w_i . Because, x_i is equal to 1 so it is nothing but equal to the value, the magnitude of that particular component of w_i . So, effectively this w_i , magnitude of it simply tells you that what is the importance or what is the relevance of the component x_i in decision making that, whether the sample will belong to class ω_1 or the sample will belong to class ω_2 . If value of w_i is very large then x_i has more weightage to decide about the class, if value of w_i is small then x_i has less weightage to decide about the class.

And in other case, if p_i is equal to q_i that means value of x_i to be equal to 1 is more likely, is same equally likely, even if the x belongs to class ω_1 or the feature vector x belong to class ω_2 . So, p_i is equal to q_i , p_i as we said it is the probability that x_i will be equal to 1, if the true state of nature is ω_1 . And q_i is the probability that, x_i will be equal to 1 if the true state of nature is ω_2 . So, if p_i is equal to q_i that simply indicates that, whether x belongs to class ω_1 or x belongs to ω_2 , x_i is equally likely to have value equal to 1. So, that simply means that x_i has no relevance in deciding the class.

So if x_i has no relevance in deciding the class then why should the corresponding vector w_i be there. I can make $w_i = 0$, without hampering my decision. So, if you come to this w_i , the expression for this w_i you have find that if p_i is equal to q_i then this expression p_i into 1 minus q_i upon q_i into 1 minus p_i . This expression will be equal to 1 log of this is equal to 0 so the corresponding w_i is equal to 0. And that is quite obvious because if p_i is equal to q_i then x_i , the particular feature vector x_i has no relevance in deciding the class of the feature vector x .

On the other hand, if $p_i > q_i$, if $p_i > q_i$ then having value of $x_i = 1$ should tell me that, the sample is more likely to belong to class ω_1 than, to belong to class ω_2 . Whereas, if $p_i < q_i$

then the sample $x_i = 1$ tells me that it is more likely to belong to class ω_2 than, its likelihood to belong to class ω_1 .

So, again coming to this particular case, if $p_i > q_i$, if $p_i > q_i$ then obviously $(1 - q_i) > (1 - p_i)$.

So, in this expression the numerator becomes larger than the denominator so this value is greater than 1. And when this value is greater than 1, value of w_i is positive, if value of w_i is positive what happens to my $g(X)$?

Student: Positive.

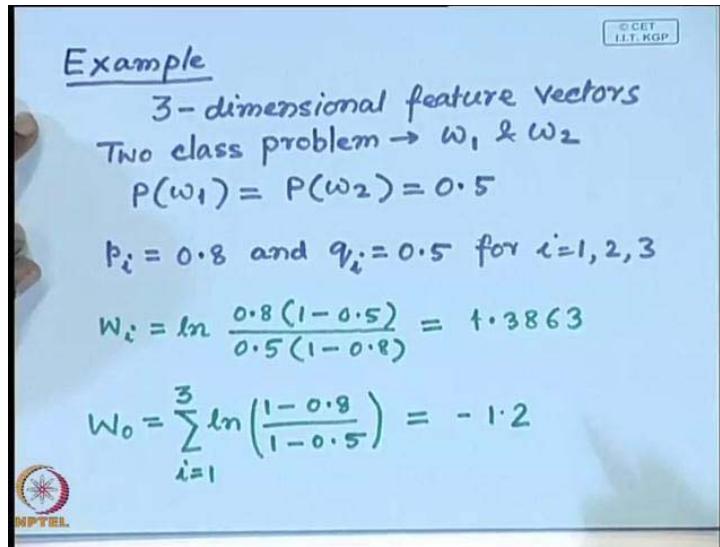
Not necessary, it depends on other values of i as well. So, effectively I can say that if $x_i = 1$, that particular component then this component x_i gives a vote of value w_i to $g(X)$ to decide that this feature vector x belongs to class ω_1 . So, it a, it gives the vote equal to the corresponding weight in favor of class ω_1 . Is that okay?

On the other hand, if $p_i < q_i$, $p_i < q_i$ so $(1 - q_i) < (1 - p_i)$ so numerator becomes less than the denominator. So, this term is a fraction which is less than 1, log of this will be negative that means, w_i in that case is negative. If w_i is negative that means, the corresponding $w_i x_i$ is trying to make $g(X) < 0$, trying to make, whether it will be less than 0 or not, that depends upon other $w_i x_i$. But what x_i is trying to do in this case, it is trying to give a vote equal to the

modulus of w_i , in favor of class ω_2 . Because, it is subtracting so it is giving a vote which is equal to modulus of w_i in favor of class ω_2 .

So if $p_i > q_i$, the component x_i gives a vote equal to w_i in favor of class ω_1 , if $p_i < q_i$ then component x_i , gives a vote equal to modulus of w_i in favor of class ω_2 . That means this component is trying to push the decision surface of the decision boundary, either towards ω_1 or towards ω_2 . Is that okay?

(Refer Slide Time: 40:15)



Let us take an example, so let us consider a three dimensional space or three dimensional feature vectors and let us consider a two class problem that is, I have classes ω_1 and ω_2 . So, these are the two classes I have and every feature vector is a three dimensional feature vector where, every individual component can be either 0 or 1.

And let us also assume that, the a priori probabilities $P(\omega_1) = P(\omega_2)$, which is equal to 0.5.

And let us assume that value of $p_i = 0.8$ and $q_i = 0.5$ for all values of i , that is for i varying from 1, 2 and 3. So, it says that every component of the feature vector has a probability of being equal to 1 is 0.8 if the feature vector is taken from class ω_1 . And every component has a probability of 0.5 of being equal to 1, if the feature vector is taken from class ω_2 .

So given this p_i and q_i values a probability values, I can compute the corresponding weight

vectors, so simply from this expression that, $w_i = \ln \frac{p_i(1-q_i)}{q_i(1-p_i)}$ for different values of i , I get

different components of the weigh vector w_i . So, I get $w_i = \ln \frac{0.8(1-0.5)}{0.5(1-0.8)}$. And if I compute

this, this becomes a value 0.3863 and the threshold w naught, which is given by this

$$\text{expression, } w_0 = \sum_{i=1}^d \left[\ln \frac{1-p_i}{1-q_i} \right] + \ln \frac{P(\omega_1)}{P(\omega_2)}.$$

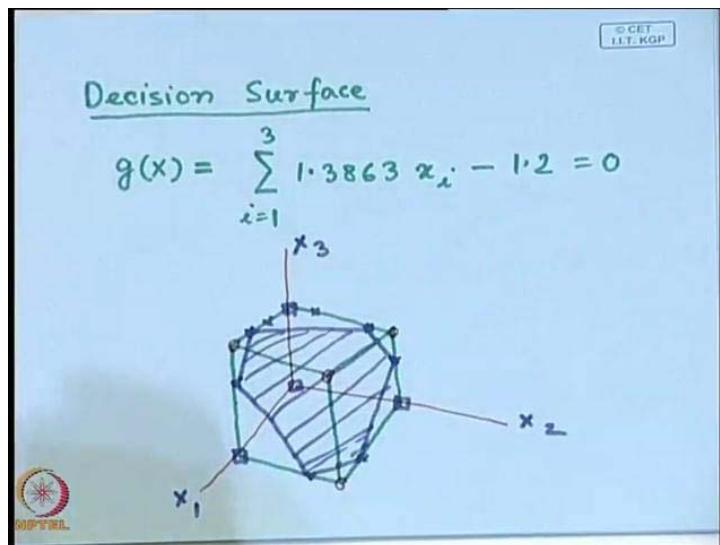
Now over here, in this case $P(\omega_1) = P(\omega_2)$ they are equal to, they are equal and both equal to

0.5. So, this last term $\ln \frac{P(\omega_1)}{P(\omega_2)}$ that will be equal to 0. So, what I have to compute is simply

this term, $w_o = \sum_{i=1}^d \left[\ln \frac{1-p_i}{1-q_i} \right]$. So, this w_o will be simply $w_o = \sum_{i=1}^3 \ln \frac{1-0.8}{1-0.5}$. This component

takes the summation for $i = 1$ to 3 and we will find that this value will be something like, it will have a value something like this.

(Refer Slide Time: 45:30)



So given this, the decision surface between the classes ω_1 and ω_2 . You verify these values, whether you are truly getting these values or not. So, assuming this our decision surface $g(X)$

will be given by $g(X) = \sum_{i=1}^3 1.3863 x_i - 1.2 = 0$, where this i varies from 1 to 3

because I have three numbers of components. So, this is nothing but

$$g(X) = \sum_{i=1}^3 1.3863 x_i - 1.2 = 0.$$

So, if I try to plot this decision surface the, in 3 d one point you might have noticed that, because our features are binary features. Every feature component can assume a value either 0 or equal to 1, so every feature vector will be represented by a vertex of a hypercube in the d

dimensional space. It can be either (0 0 0) or (0 0 1) or (0 1 0) or (1 0 0) and so on. So, every feature vector will be represented by a vertex, in the d dimensional space of a hypercube.

So, if I try to plot this decision surface, the decision surface will be something like this. So, let us take a cube in this 3 dimensional space and if you plot the surface, the surface will come out to be something like this. So, this is our decision surface so you find that, this decision surface says that, these are the points which lie on one side of the hyper plane. And these are the points which lie on the other side of the hyper plane, so it simply says that, if at least two components of the feature vector are equal to 1. Then the point is classified to class ω_1 .

Student: Sir it should be at least one because equation says it will be at least one, which one. The equation from this decision surface that you this one.

Student: Yes sir.

Student: If at least one is

I mean it is the other way, these are the points which are put to class ω_2 and the other side is put to class ω_1 . So, it says if at least one of them is equal to 1 then the decision will be in favor of class ω_1 . Otherwise, the decision will be in favor of ω_2 . Is that okay?

So, we find that I again get a simple hyper plane in three dimensions, it is just a plane, which is boundary between the two classes ω_1 and ω_2 . So, if the probability values are different, if different position have different other values then the position as well as orientation of this plane may be different. But effectively what it does is, the entire d dimensional space is broke into two halves, one half will be given to class ω_1 , the other half will be given to class ω_2 . The nature will be a bit more complicated when, the numbers of classes are more than 2, because then we have to think of more than one decision surfaces and how they combine.

Student: Sir, will axis represents x_1, x_2 ?

Axis represents x_1, x_2, x_3 . So, let us stop here today.

Thank you.

**THIS BOOK IS NOT FOR SALE
NOR COMMERCIAL USE**

PH: (044) 2257 5905(08)

nptel.ac.in

swayam.gov.in