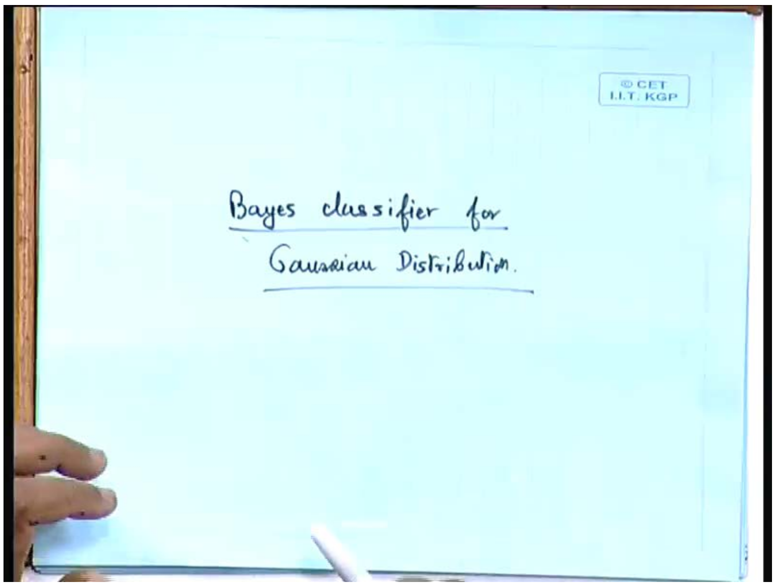
Neural Network and Applications

Prof. S. Sengupta
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 18
Bayes Classifier for Gaussian Distribution

Bayes Classifier for Gaussian Distribution.

(Refer Slide Time: 00:53)



© CET
I.I.T. KGP

Bayes classifier for
Gaussian Distribution.

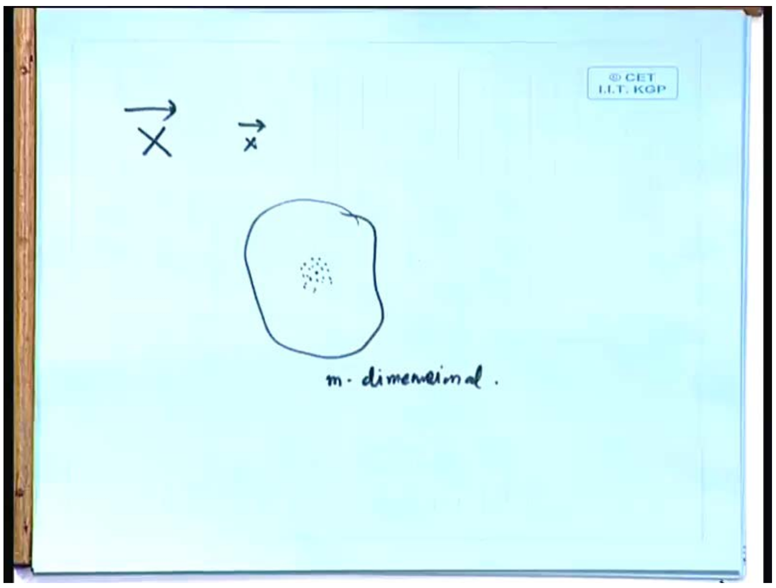
This was a topic, which I intended to cover up last time itself, in the last lecture, but due to insufficient time, we could not. So, today, we are going to first discuss upon this, in fact, I may although this is the title, given to the lecture. I think that, I am going to spend roughly half the lecture on this. And then, in the remaining part of the time, we will go over to the discussion about the multi-layer perceptron. To the extent time permit is, we will be discussing about the back propagation algorithm.

Now, in the last class, we were discussing about the Bayes classifiers and in fact to one point, which you must have noted is that, the Bayes classifier, computational part. Only involves the computation of the log likelihood ratio or rather to say the likelihood ratio is the thing; that we are first computing in terms of. If f x, x given C 1 and f of x, x given C 2, the ratio of these two, gives us a likelihood ratio.

And then, we took the log likelihood ratio and this log likelihood ratio was compare with another factor called xi, which was defined as the threshold. Say, if the log likelihood ratio is greater than the threshold, we were classifying it into one of the classes, let us say C 1. And if it is otherwise, then we were classifying it into the other class; that is C 2. That was the simplest kind of approach that we had taken.

And in fact, our intention was to show, that for a Gaussian distributed patterns, we are going to show that the Bayes classifier is going to act very similarly to what a linear Perceptron does. In fact, this is a very interesting result that we are going to talk about. So, again we consider a two class problem that, it is going to classified into either the class C 1 or C 2. And another point that, we should keep in mind is that, all the vectors, that we are deciding, that we are discussing about.

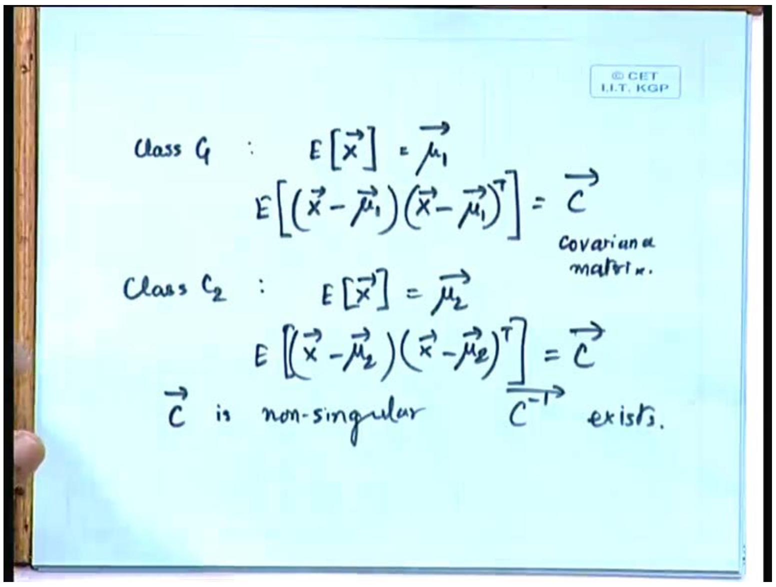
(Refer Slide Time: 03:39)



In fact, we are taking the X vector, the capital X vector to be a random variable, out of which we are taking instances as the small x vector. And we have consider, that this x vector is an m dimensional type. So, that, whenever we are considering an m dimensional space, like say for example here, if we draw any imaginary m dimensional space. And supposing, this is the space in which the x vector, that is the random variable can reside, then the instances of the small x vector that, we are going to have will be and let us say that, this is the mean vector.

Now, most of the vectors, belonging to this random variable, would be distributed close to this mean vector and as we move away from the mean vector, their population will decrease. So, that is why, the use of the probability density function, the PDF is coming into play. And we are assuming, the PDF distribution to be a Gaussian distribution in this case.

(Refer Slide Time: 05:02)



© CET I.I.T. KGP

Class C_1 : $E[\vec{x}] = \vec{\mu}_1$

$E[(\vec{x}-\vec{\mu}_1)(\vec{x}-\vec{\mu}_1)^T] = \vec{C}$ Covariance matrix.

Class C_2 : $E[\vec{x}] = \vec{\mu}_2$

$E[(\vec{x}-\vec{\mu}_2)(\vec{x}-\vec{\mu}_2)^T] = \vec{C}$

$\vec{C}$ is non-singular $\vec{C^{-1}}$ exists.

So, let us consider a two class classifier problem, where for class C 1, we are going to have the expectation of this random variable X, capital X that will be equal to what, that will be equal to the mean. That will be equal to mean and in this case, mean a scalar or a vector, definitely a vector. Because, all the instances of this random variable X, they are vectors.

So, definitely when we are talking of a mean, the mean is also a vector, mean is also an m dimensional vector, so we call this mean vector as the mu 1 vector. And then, another statistical parameter that we always consider for the case of scalar things, what we consider is the variance. Because, that shows the dispersion of the data with respect to it is mean.

That is how, we calculate, that if the data is much dispersed, from it is mean, then it shows a large variance, where as a data, which is closer to the mean, data distribution which is very close to the mean, that will show a smaller variance. Now, variance in the case of a scalar variable, the variance was also a scalar quantity. But, in this case, the variance that we are talking of is also not a scalar.

Variance has to be in this case, the vector definitely, it is in fact of a matrix style, because the matrix should indicate the dispersion, not only with reference to itself, which will be indicated by it is diagonal elements of the matrix. But, also the off diagonal elements, which will basically show the correlation between the data with it is other elements.

So, basically, we are going to form a matrix, which is called as the co-variance matrix, which is going to be the equivalent of the variance that we are having for the case of scalar data. So, the co-variance matrix that we are talking of should be defined in this manner. So, it should be the expectation of x minus mu 1, all in vector notations, times x minus mu 1 in this case transpose.

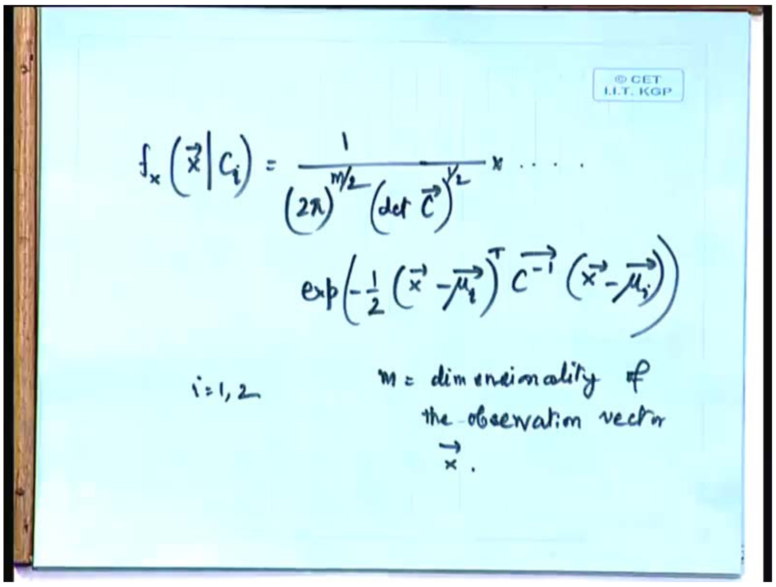
So, if we make it a transpose, we are taking an outer product of it, so by taking the outer product, we are going to get a matrix and that matrix we are going to call as the matrix C which is nothing but, the co-variance matrix, so this is the co-variance matrix. And very similarly, for the case of class C 2, we are going to have the expectation of x vector to be equal to mu 2 vectors and the expectation of x minus mu 2 vectors.

Outer product with x minus mu 2 transpose, this is also taken to this C, assuming that the co-variance characteristic is uniform throughout the entire space H that we are talking of. Because, out of C 1, we are taking the sample space to be, we are taking only a subset of that H 1 from we derive the training samples for C 1, again likewise H 2. We take a subset of C 2 and then, we were defining in the last class, the whole set H from which, we were considering.

So, assuming that the variance statistics, remaining the same, for that, we can denote it by the same C matrix, which is the co-variance matrix. Now, another point to be noted is that, this co-variance matrix C is non-singular and so it is, C inverse exists. So, now, we are going to define a Gaussian distribution for both the classes, f x, x given C 1, also f x, x given C 2, both these distributions should be Gaussian.

And the Gaussian formula is very well known to us, we have been dealing with such formulas, almost in all the subjects. We are having Gaussian distribution as a typical normal distribution case. Only thing is that, in this case our data will be little more complicated in the sense that, we are going to have m dimensional type. The Gaussian distribution is going to be m dimensional. So, let us write down the Gaussian distribution expression, considering an m dimensional case.

(Refer Slide Time: 10:25)



© CET I.I.T. KGP

$$f_x(\vec{x}|C_i) = \frac{1}{(2\pi)^{m/2}(\det \vec{C})^{1/2}} \times \dots$$

$$\exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu_i})^T \vec{C^{-1}} (\vec{x}-\vec{\mu_i})\right)$$

$i = 1, 2$

m = dimensionality of the observation vector $\vec{x}$.

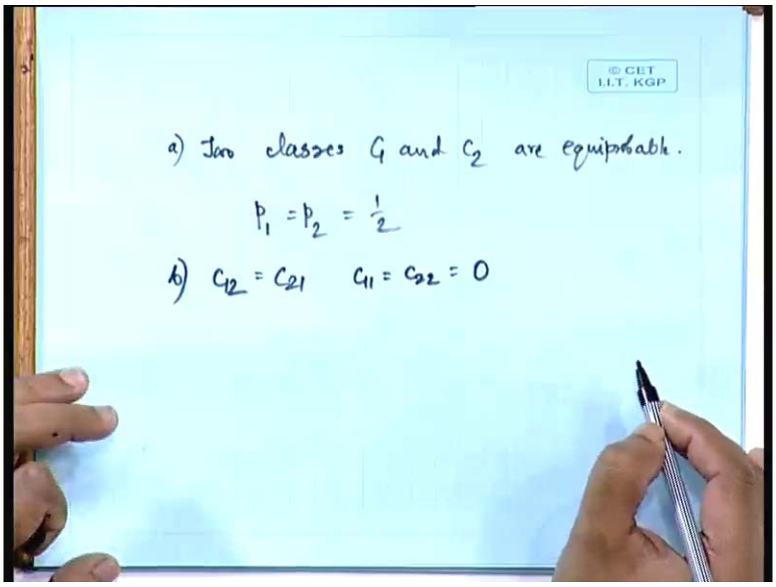
We are going to write as f x, x given C i, that is equal to 1 upon 2 pi, in this case, it will be m by 2, 2 pi to the power m by 2, m is coming because of the dimensionality, that is involved in this case, it is m dimensional. And then, instead of the 2 pi sigma square, you remember that we have root over 2 pi into sigma square. That sigma square term will be, in this case, replaced by the determinant of C and this whole thing to the power half.

And then, the exponential term that will be existing is I am not writing the exponential term here, because I need some more space for it. So, it should be exponential to the power minus half x minus mu I, so here I am putting as C i. So, that this will be valid for i is equal to 1 and 2, two class problem, because it is. So, this x minus mu i transpose C inverse x minus mu i. So, I think we have understood the form; this is a quadratic form of the expression.

So, this is in other words, the equivalent of the expression that we are going to have for the one dimensional case or the scalar case. So, this is with the knowledge, that m is equal to the dimensionality, dimensionality of the observation vector x. Now, we can make some further assumptions and that is nothing wrong in it.

Firstly, that we are going to have two classes C 1 and C 2, although in the last analysis, we have assumed the two probabilities to be independent P 1 and P 2 we took. So, were P 1 and P 2 could take any values, but we can always assume that, the two classes are equi probable, in most of the cases, it may be like that only.

(Refer Slide Time: 12:47)



© CET
I.I.T. KGP

a) Two classes C_1 and C_2 are equiprobable.

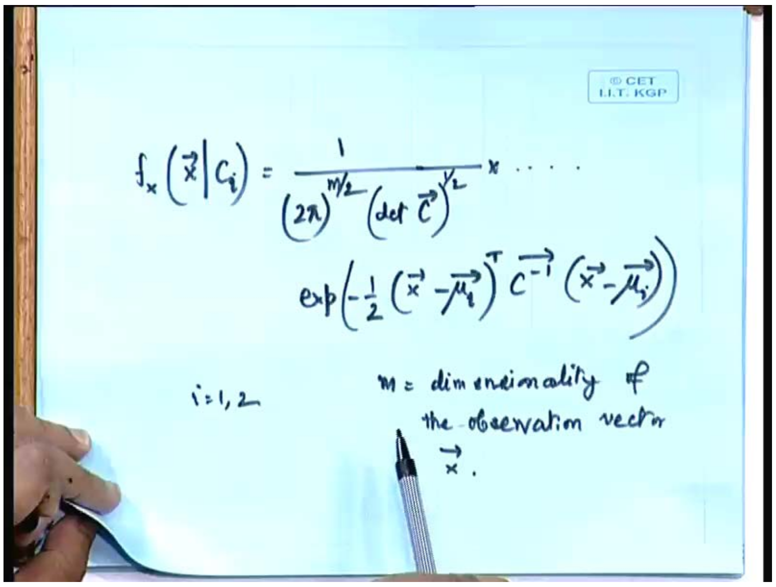
$$P_1 = P_2 = \frac{1}{2}$$

b) $C_{12} = C_{21} \quad C_{11} = C_{22} = 0$

So, if two classes C 1 and C 2 are equi probable, in that case, we are going to have in which case, we can obviously, write that P 1 is equal to P 2 is equal to half. And also, we can write down some problem with the sketch pens, any way, this is better. So, C 12 is equal to C 21, the cost function that we had taken last time. C 12 and C 21’s are what, they are the cost functions, which are associated with correct, incorrect, incorrect classifications, so C 12 and C 21 and they are the incorrect classifications cost function.

And C 11 and C 22, they are the correct classification cost functions, which there is no reason, why we should not take it to be 0. Because, we said that, those cost functions should be as minimum as possible as low as possible. So, there is nothing wrong, if I assume that C 11 is equal to C 22 is equal to 0.

(Refer Slide Time: 14:08)



$$f_x(\vec{x}|C_i) = \frac{1}{(2\pi)^{m/2}(\det \vec{C})^{1/2}} \times \ldots \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T \vec{C^{-1}} (\vec{x}-\vec{\mu}_i)\right)$$

$i = 1, 2$

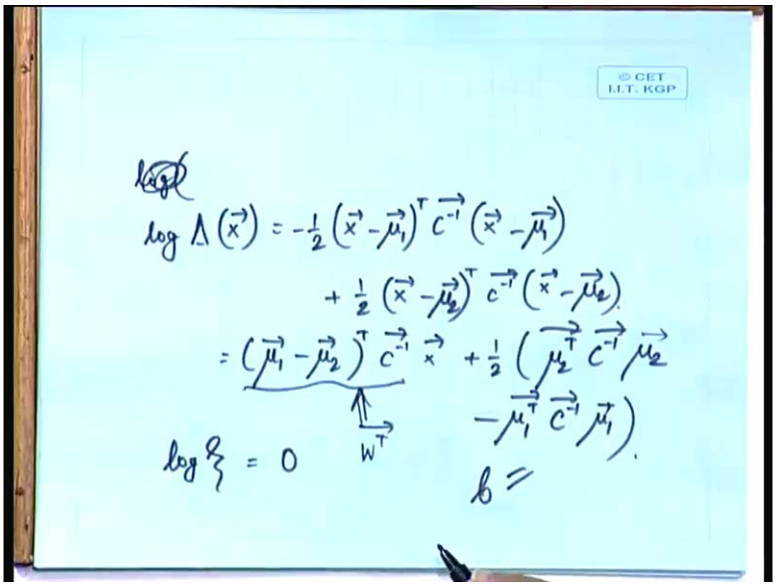
m = dimensionality of the observation vector $\vec{x}$.

So, if we have that and if we try to take the likelihood ratio, because in order to take the likelihood ratio, what we have to do is, we have to first write the expression for C 1, x given C 1. Then, we have to write it for x given C 2 and simply we have to take the ratio of these 2. And you can very clearly see, that if you are taking the likelihood ratio, then this coefficient term will get canceled out, they become one and only the quantity written within the exponential, will remain.

Exponential to the power something by exponential to the power something, which will mean that, exponential to the power, the expression that we are going to have for mu 1. And the expression that we are going to have for mu 2, they will be just subtracted from each other, if we are assuming that f x, x given C 1 by f x, x given C 2. Now, because the quantity that we are getting as a likelihood function is everything written within exponential term only, the most logical thing will be to take a natural logarithm of both the sides.

Natural logarithm of the likelihood functions, which we are going to call, which we have been calling so long as the log likelihood ratio. So, the log likelihood ratio, whenever to be take the natural logarithm of both the sides, it will be log likelihood ratio on the left hand side and the right hand side term involving the exponential will then, become a straight forward term, it becomes a linear term in fact.

(Refer Slide Time: 15:54)



$$\log \Lambda(\vec{x}) = -\frac{1}{2}(\vec{x}-\vec{\mu}_1)^T \vec{C}^{-1}(\vec{x}-\vec{\mu}_1) + \frac{1}{2}(\vec{x}-\vec{\mu}_2)^T \vec{C}^{-1}(\vec{x}-\vec{\mu}_2)$$

$$= \underbrace{(\vec{\mu}_1-\vec{\mu}_2)^T \vec{C}^{-1}}_{\vec{W}^T} \vec{x} + \frac{1}{2}\left(\vec{\mu}_2^T \vec{C}^{-1} \vec{\mu}_2 - \vec{\mu}_1^T \vec{C}^{-1} \vec{\mu}_1\right).$$

$\log \xi = 0$

$b =$

So, if we take the natural logarithm of the likelihood ratio, we can write down this expression directly, that the log of, sorry, the log lambda x vector, that is the log likelihood ratio. It is going to be equal to minus of half x minus mu 1, transpose C inverse x minus mu 1, plus half of x minus mu 2, transpose C inverse x minus mu 2. So, the mu 2 term and from that, we subtract the mu 1 term.

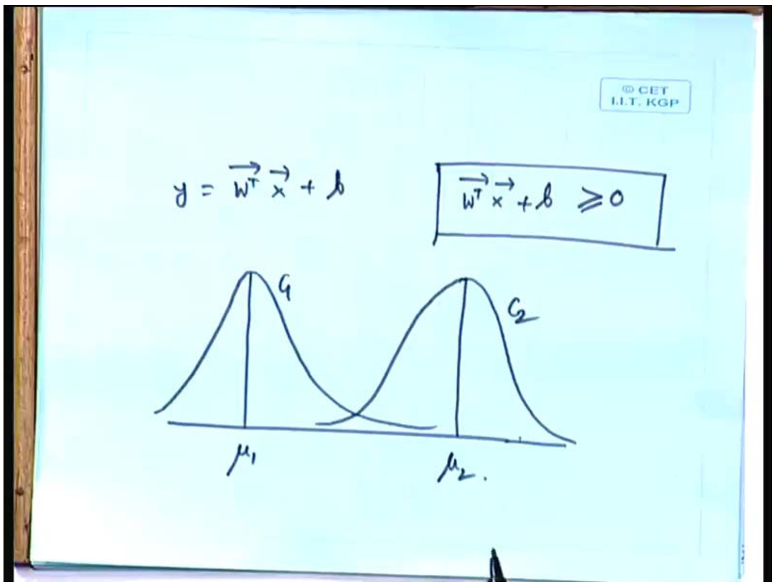
So, simply the ratio of f x, x given C 1 and f x, x given C 2, taking the logarithm of both the sides, so this could be simplified further and if we simplified it, then we will see that, we will get two terms here. One is mu 1 minus mu 2, this transpose C inverse x vector plus half of mu 2 transpose C inverse, mu 2 minus mu 1 transpose C inverse, mu 1. And what is going to be log xi in this case, because we are assuming P 1 and P 2 to be equal to half and C 12 and C 21 to be equal.

Just have a look at the earlier class note, there you will find that xi term will be equal to 1, so that in this case, log xi is simply equal to 0. So, in that case, what is going to be the boundary, what is going to be the decision boundary? The decision boundary should be when log of lambda x, the log likelihood ratio is greater than or equal to log xi in this case 0.

So, when this expression is greater than or equal to 0, have you got it, what we are wanting, is it linear. It is very much linear, you see that this involves, if you look at the addition terms of these two and there are two terms that is invert over here. One is a term involving x and the term, which does not involve x, which is independent of x. So, the term, that is independent of x; that means to say this mu 2, C 1, this C inverse mu 2 minus mu 1 transpose C inverse mu 1, this term that we have got is independent of x.

And you could imagine that, this is like the bias, this can be substituted for the bias b and if we now substitute this for the W transpose. In that case, what is the expression that we are getting, we are getting and if we call this to be y.

(Refer Slide Time: 19:23)


© CET I.I.T. KGP
$y = \vec{W}^T\vec{x} + b$
$\boxed{\vec{W}^T\vec{x} + b \geq 0}$
C_1
C_2
μ_1
μ_2.

In that case, the equation that we are getting out of this log likelihood ratio for the Gaussian distribution Bayes classifier is simply y is equal to W T x vector plus b, which is the equation of. So, this is equal to this, so that means to say that, that the decision boundaries equation is W transpose x plus b greater than or equal to 0. This is the decision boundary, which is the hyper surface equation and which is our simple Perceptron equation.

So, we started with Bayes classifier under Gaussian distribution assumption and we landed up in a form, the classification decision boundary solution that, we are getting out of it, is exactly the way, the decision boundary problem is for the Perceptron. So, the Analogy that we were talking about of since the last class, about the Bayes classifier and the single layer Perceptron, very much holds good.

So, I think you should now feel convinced the there is a very strong Analogy, a very strong similarity is indeed there. But, come to think of it, in a more deeper way, you can realize, that there are some striking differences also in addition to whatever resemblance, we have got in terms of this decision equation.

Number 1; that when we talk of the Perceptron, the first basic fundamental assumption that we are making is that the two classes C 1 and C 2, they are linearly separable, not only separable, they are linearly separable. But, the Bayes classifier problem that we have defined here is based on the assumption, that the distribution is Gaussian the distribution of the two classes, they are Gaussian in nature.

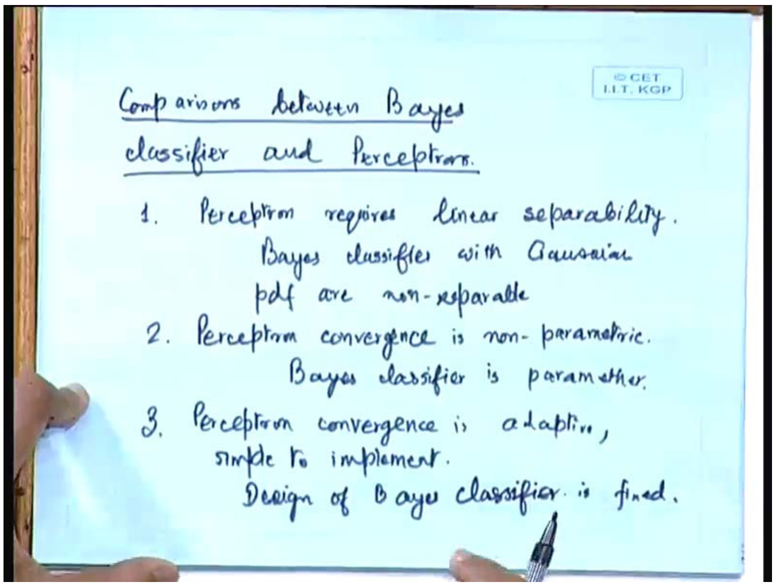
And even, if you have mu 1 and mu 2 to be much separated from each other, let us say for example, since I cannot sketch a Gaussian distribution for multidimensional, m dimensional case, I can at least sketch for a single one dimensional case of Gaussian. So, here, let us say that this is mu 1 and let us say this is mu 2, widely separated. And also, I assume that as compare to their distance of separation, mu 2 minus mu 1, there variance is also relatively of smaller magnitude.

Even, then also the distribution, which could be plotted something like this, if this is the kind of the distributions that we have got, this being the distribution of plus C 1 and this being the distribution for class C 2. They are in strict sense, not at all overlapping, they are not at all non overlapping, as the linear classifier would require, as the perceptions would require.

That is the fundamental condition of Perceptron, that the pattern classes must be linearly separable, but in this case, the pattern classes are theoretically overlapping. No matter, how for you drive them away, mu 1 and mu 2, you drive them away and however, small C 1 and C 2, you can use, still they are overlapping to some extent and the thing is that. So, the naturally, they are lies a very fundamental difference, that here, this is not linearly separable problem, so Bayes classifier can solve it, Perceptron cannot.

Again, Bayes classifier for the case of Gaussian distribution only, is getting converted into this type of a linear form. If you are taking distributions, other than the Gaussian, then this relationship will not hold good. Whereas, in the case of Perceptron, this relation always holds good, because Perceptron does not make any underlying assumption about the probability density functions of the classes, so that is why, they Perceptron we can say is non parametric.

(Refer Slide Time: 24:18)



Comparisons between Bayes classifier and Perceptrons.

1. Perceptron requires linear separability.
   Bayes classifier with Gaussian pdf are non-separable
2. Perceptron convergence is non-parametric.
   Bayes classifier is parametric.
3. Perceptron convergence is adaptive, simple to implement.
   Design of Bayes classifier is fixed.

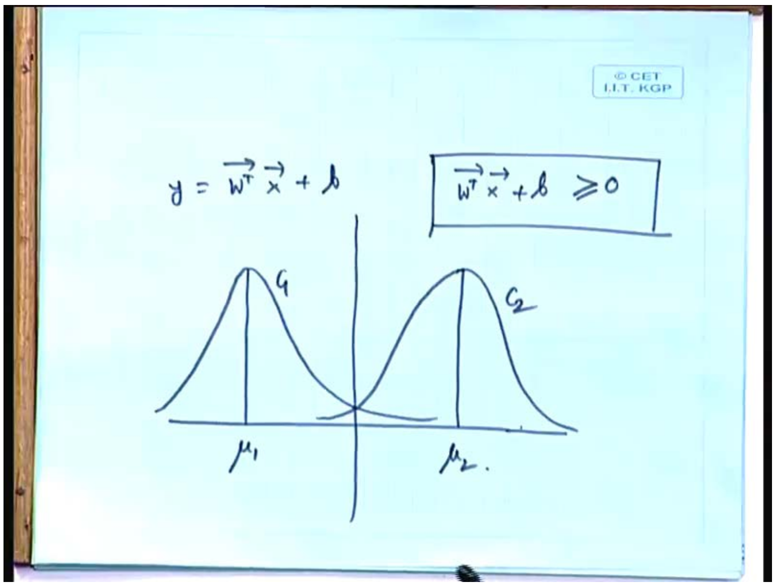
So, we can in fact just jot down a few points about it is comparison, most of it we already discussed, comparisons between Bayes classifier and Perceptrons. Number 1; comparison is that Perceptron requires linear separability, whereas, Bayes classifier with Gaussian pdf are non-separable, they are absolutely non-separable class. No question of linear separability, they are absolutely non-separable.

The second is that Perceptron convergence is non-parametric, non parametric means, it is not dependent on the underlying distribution, underlying probability density function. Whereas, Bayes classifier is indeed parametric, because it is under the assumption of Gaussian, that it is linear. And then, the third point is that, the Perceptron convergence algorithm is adaptive and simple to implement.

Whereas, the design of Bayes classifier is fixed, because in this case design of Bayes classifier is very much dependent upon the log likelihood ratio, that we are computing, so it is fixed. You can make it adaptive, when it is statistics changes, but then you have a very large memory to track down, it is statistical behavior. That means to say, what you have to do is to track down, it is mu 2, mu 1, mu 2, C, the co-variance matrix variations.

Unless, you do that, you cannot make the Bayes classifier to be adaptive, so these are some of the dissimilarities, that exists and only under the linear case. That only, under the Gaussian distribution case, that the Bayes classifier with Gaussian pdf is very much similar to the way Perceptron behaves. But, if the underlying distribution is indeed Gaussian, if the two classes of patterns are Gaussian distributed really.

(Refer Slide Time: 27:51)


© CET I.I.T. KGP
$y = \vec{W}^T \vec{x} + b$
$\boxed{\vec{W}^T \vec{x} + b \geq 0}$
C_1
C_2
μ_1
μ_2.

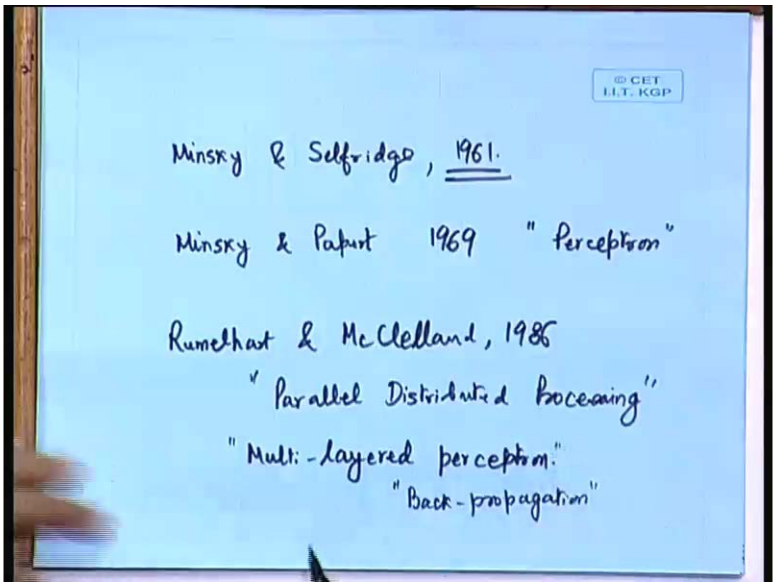
And if we want a Perceptron to solve the problem for that, then there lies a problem, like say for example, if this is a kind of problem that, we give to a Perceptron, for a solution. Then, although the ideal decision boundary could be somewhere over here, but this decision boundary is certainly going to shift. Because, whenever you are trying to achieve a Perceptron convergence, because it is not linearly separable, the two classes are not linearly separable.

So, that is why, it will not be able to find a any fixed decision boundary, if you are alternatively trying to converge a Perceptron algorithm, the decision boundary will shift in and around this region. It will oscillate in and around this region and never obtain any direct solution of the decision boundary. So, this much is the discussion that, I wanted to have on the broad chapter of single layer Perceptron.

And in fact, this chapter we should end up with some amount of historical note in it. In fact, there is some historical note background, which I had given to you earlier. But, just to re mention it and also to rethink about it, in terms of the multilayer Perceptron, which we are going to discuss shortly. The thing as I have already told you is that this theory is related to Perceptron.

The theory is related to the least means square algorithm, they were all developed in the 1950's and the Perceptron theory was actually proposed by Rosenblatt in the 1950's.

(Refer Slide Time: 29:40)



© CET I.I.T. KGP

Minsky & Selfridge, 1961.

Minsky & Papert 1969 " Perceptron"

Rumelhart & McClelland, 1986

" Parallel Distributed Processing"

" Multi-layered perceptron."

" Back-propagation"

Now, in 1960's, actually it was, after Rosenblatt theory was presented, it was Minsky and Selfridge, these two people these two scientists. In the year 1961, they had presented a paper, where they had shown that, the inadequacies of Perceptron, in the sense that, Perceptron cannot solve problems, which are not linearly separable. In fact, they were the first demonstrate the incapability of Perceptrons to classify the exclusive or kind of a problem, simple binary exclusive or problem could not be solved, this was shown.

But, subsequently, it was argued that, so in this sense means Minsky and Selfridge was very much right, made a very timely contribution, saying the inadequacies of Perceptron. And then, when it was felt that perhaps, Perceptrons limitations might be overcome, if one uses say, multi layer Perceptron. A single layer Perceptrons limitation could be overcome by multi layer Perceptron and when theories like that, were proposed.

That time Minsky and Papert, these two scientists, they in the year 1969, they published a book, a very well known book, a classical book named as Perceptron. In which book, they pointed out that the inadequacies of single layer Perceptron definitely exists and one has to accept that. In fact, people who first proposed the theory themselves accepted, there was no reason, why they should not have.

But, they were highly skeptic about the performance of multilayer Perceptron, because Minsky and Papert directly ask them. That if the problem like this is not solvable by single layer Perceptron, how can you expect, to solve it, get it solved by multilayer Perceptron. And very brilliant scientists as they were, there was a thinking in the thinking and rethinking in the neural network research community, about the whole success of neural network.

Because, any way neural network has to sometimes solve problems, which are separable, but with are non-linearly separable. So, if neural network cannot solve non-linearly separable problems, then there is no use of neural networks. In fact, following the book by Perceptron by Minsky and Papert in 1969, there was a very much subdued research activity on the neural networks.

In fact, for the next 10, 15 years not many people talked much about neural networks, especially the Perceptron and the multilayer Perceptron model, people thought that may be that this is something, which was not very successful. It was proposed, but not very successful, but in the year 1986, it was two scientists, again named Rumelhart and McClellant. In the year 1986, they published a book, which is named as Parallel Distributed Processing; just observe the name of this book.

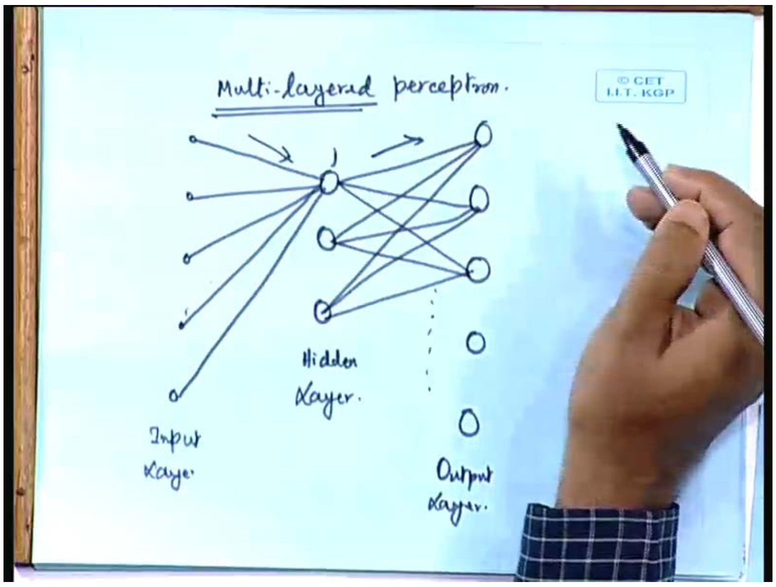
We heard about a subject, a very popular subject called parallel and distributed processing. But, mind you here the name that is given to this book is Parallel Distributed Processing; there is an omission of act. But, the whole thing, that they try to talk about in this book is that neural network is definitely a parallel computational model, no doubt about it. Because, you can employ several neurons and you can do the processing independently.

They are independent, they are indeed distributed, because those neurons, did not be distributed in one single place, the neurons could be distributed in several places, they could be distributed. So, it is the parallel distributed processing, that was proposed by Rumelhart and McClellant in the year 1986, parallel distributed processing book was written.

And that was the first book, after which the concept of multi layered Perceptrons, were popularized. In fact, they are the first people to coin the term of back propagation. In fact, lot of times, I find that the word back propagation and multilayered Perceptrons, they are very much interchangeably used. It should not be very much interchangeably used, because back propagation this word itself has got some concept built into it.

Whereas, coming to think of multilayered Perceptron, the idea is pretty simple, what is meant by multilayered Perceptron is that instead of only the input and the output layer, the way we have it for the single layer Perceptron. We are going to have a layer in between, which is called as the hidden layer. So, multilayered Perceptron architecture would look something like this.

(Refer Slide Time: 35:43)



There will be an output layer, there will be a hidden layer, I might have told, given this diagram sometimes or the other, while discussing other things, I must have talked about this. So, there is nothing great about this diagram, but one thing that these hidden layers are fully interconnected to the layer, that is preceding it. In fact, all the layers starting from the output up to the layer immediately, before the input, it is fully interconnected to the layer, which is preceding it, which means to say, what.

That, if we are talking of this, then it is fully interconnected to all the hidden layer neurons like this is going happen for all of this output layer neurons.. Like this, it goes on; I am not completing this diagram. Again, here this should be connected to all the neurons in the previous layer. Now, in this example, I have shown one input layer, one hidden layer and one output layer.

But, generally speaking, you could have more than one number of hidden layers also, but the very fact that, it is a multilayered Perceptron. Because, it is multilayered Perceptron, it must necessarily have at least one hidden layer. Now, that the thing is that having told about the multilayered Perceptron, I think it is very much necessary to talk about what the term back propagation actually means.

Because, I have seen that some people have got certain misconceptions about the back propagation and I have heard some people wrongly talking. That as if to say, that back propagation as if tries to mean that, like there is a forward signal flow, which will be there. In fact, what happens is that, this input will be having some signals associated with it and these signals will come to the next layer in this case, hidden layer, which will be.

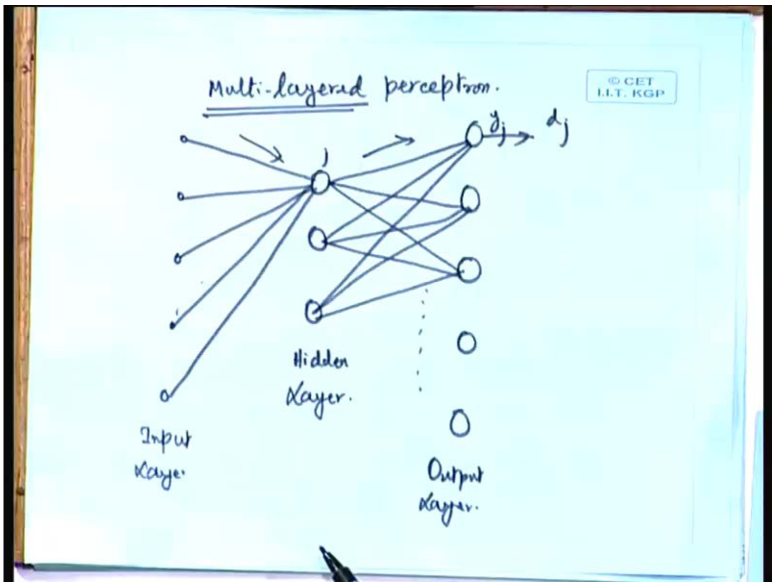
So, there will be some linear weighted inputs, that you will get, which we are going to call as the induced field, that is to say sigma of W j I, if you call this to be the neuron j. Then, sigma W j i, x i, where i is equal to 1 to m, if there are m dimensions in the input. We were describing the linear activation to be that and then we were following it up with the activation function.

Phi of v; that is what we were taking which was in fact doing some kind of a hard limiting and things like that. Now, there is a forward signal flow, in this case, if you observe. The signal will first flow from the input layer to the hidden layer and then, the outputs of this hidden layer will then influence the output neurons. So, as if to say that the signals are propagating in the forward way.

Now, some people wrongly believe that back propagation means, that as if to say you have got a feedback mechanism in the connection and the connections are bidirectional and some signal also flows from output to the input, which is not very correct. In fact, the word back propagation is to be used very cautiously, that here back propagation means, that it is back propagation of errors.

Please note that, it is back propagation of errors, so that in the forward signal flow path in the forward path, we have got only the signal flow and in the reverse path. We are going to have the computation of errors, because one thing is to be understood at this stage. If you are again, bringing in the Analogy of single layer Perceptrons, there try to remember, what we did.

(Refer Slide Time: 40:18)



In single layer Perceptron, we were considering a neuron, which was connected to several inputs and then, it was generating some outputs, let us say y. And then, there was a desired output, which was d and we were feeding as a pattern set, we were feeding the x vector, as well as the d, as the training patterns. And this y minus d or that was the error term for us, using which we were directly modifying the synaptic weights here.

Now, if we try to do the same in the case of multi layered Perceptrons, how are we going to achieve it. Because, here there will be some final output, let us take any one output neuron, there will be some output, let say y or if this is the jth output neuron. Let say, there will be some y j, there will be something which is d j. So, accordingly there will be some error e j, which is y j minus d j.

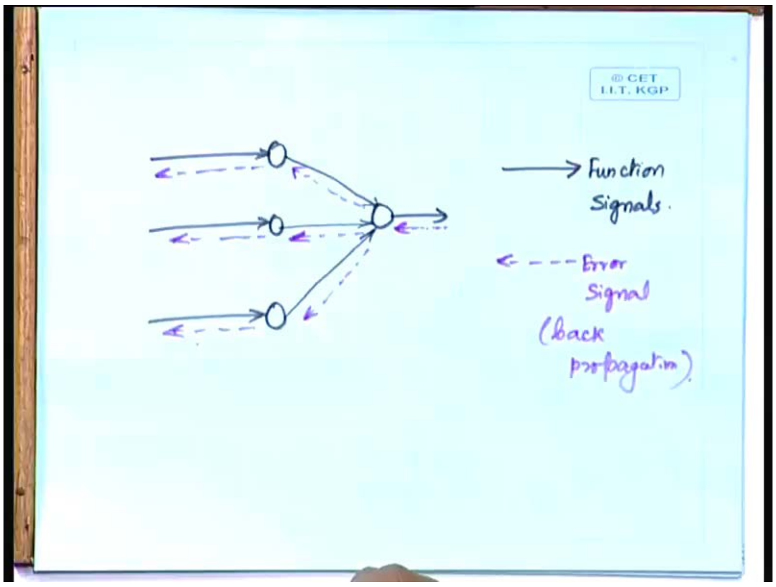
But, that error will can influence or using that error we can update the weights that, are connected between the output and the input, which is immediate layer, which is immediately preceding it. But, if somebody asks me the question that is well and fine you can adjust the weights belonging to this layer, but how about weights belonging to this layer.

I cannot do it, because in order to adjust the weights over here, I have to get an equivalent picture, that as if to say, that I know, that what this output should have been, what the equivalent desired output should have been at this stage. Only, if I had known that, the actual output and the equivalent desired output, which should have been at this stage. Only, then I could have computed the error and only, if I could compute the error, I could adjust the weights at this layer.

And like that, I have to go back to the preceding layers, because there may be more than one hidden layer of neurons. So, like that I have to go to the preceding layers of a neurons and I would find out that, what should be the weight adjustments that are needed in the preceding layers. So, there what we are going to do is, that the error that we are getting, this error should be computed in it is equivalent form in all the previous layers and that is the whole idea of the back propagation of errors.

So, when we talk back propagation, again mind you, be careful in using the term, that it is back propagation of errors.

(Refer Slide Time: 43:27)



In fact, it should look something like this, that let say, that we have got three neurons, let say these are the last hidden layer. The hidden layer that is closest to the output layer, this is an output layer neuron, let us say connected to three hidden layer neurons. So that, here we are getting the signals from these three neurons and then, here we are getting the inputs and we are going to get the outputs, the output will be here.

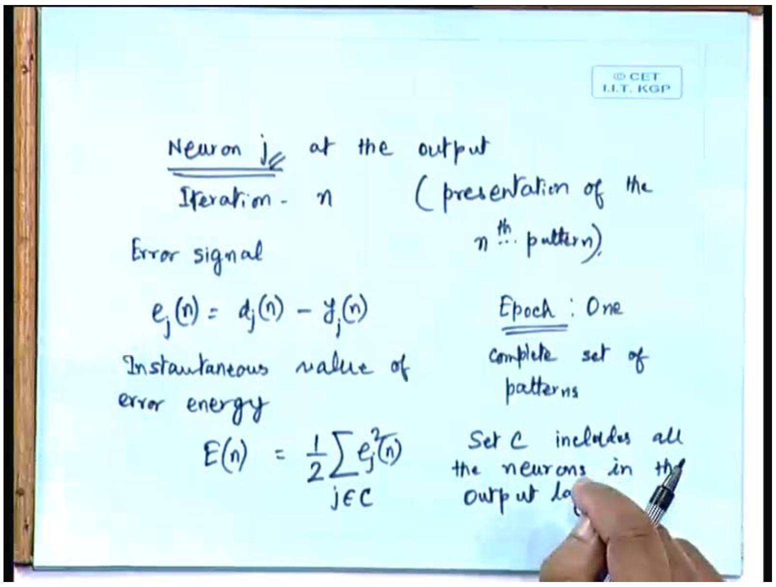
So, all these lines, that I am drawing with from black lines, they are nothing but the function signals. So, these are the function signals and then, the errors signals would be shown like this, so we first compute the error here at the output layer. And then, we have to propagate these errors backward up to the level of the immediate before, immediately preceding hidden layer. And then, once here the errors are computed, then this error should be further back propagated, so this is the flow of the error signal.

So, what I am drawing with this color is the error signal, which is by back propagation, so this is the concept. Now, let us go over to the beginning of the back propagation algorithm. Now, the beginning of back propagation algorithm would look very similar to the weight updating equation or what we called as the delta rule for the case of single layer Perceptron.

The analysis and everything would look very similar to that, but only thing is that we will realize the difference, when we come back from the output layer to the preceding layers. When, we come to that and we calculate the equivalent error signals, there we will understand the difference between the approaches that we had followed so for. Single layer Perceptron and the approach, that we are going to follow for the back propagation.

So, the approach of back propagation would start in a very similar way, so let us consider an output neuron j.

(Refer Slide Time: 46:20)



Neuron j at the output

Iteration - n (presentation of the n^{th} pattern)

Error signal

$e_j(n) = d_j(n) - y_j(n)$

Epoch : One complete set of patterns

Instantaneous value of error energy

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n)$$

Set C includes all the neurons in the output la

So, we consider neuron j at the output and we consider the iteration number to be n, so we consider the iteration n and here iteration n means, it is the presentation of the nth pattern. So, if we take the approach of the l m s algorithm, there what you did you remember, there we were considering the instantaneous error energy and we were minimizing the instantaneous error energy and updating the weight for every n.

That means to say, that we were not waiting for a one complete set of patterns to be presented. In fact, one complete set of patterns is called as Epoch, so there we were not weighting Epoch is one complete set of patterns. So, instead of waiting for one complete set of patterns or an Epoch. We were updating the weights with the presentation of every training sample that, is what we were doing for the case of l m s, because we were considering only the instantaneous error energy.

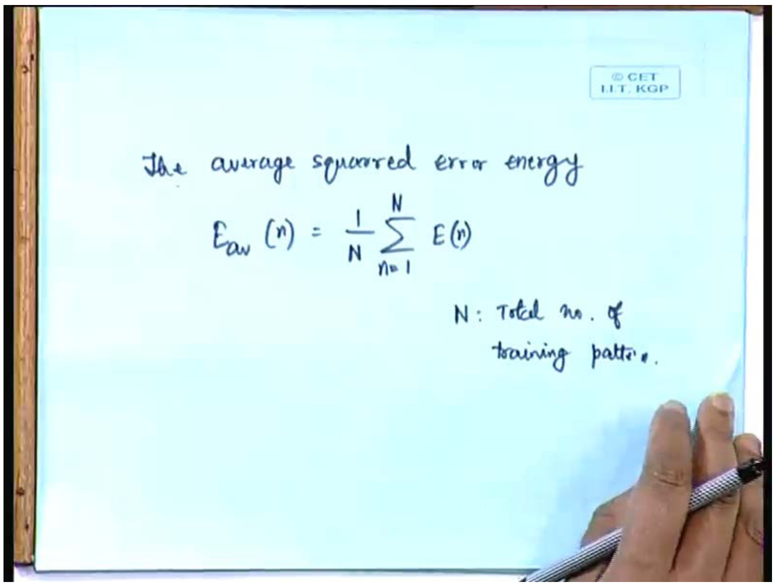
We were only considering, what e j at iteration n square of that, so here also we are going to follow the similar approach as if to say that with every presentation of the pattern, we are going to update the weight. And later on, we can show that whether that becomes equivalent to updating the weight once for all for one complete Epoch. Because, if you are updating the weight after one complete Epoch.

Then, you have to calculate the average error energy that, should be based on the average error energy, whereas doing it for every pattern means based on the instantaneous error energy itself, we can do it. So, now the error signal at the output of the neuron j, could be written very simply. So, the error signal a at the iteration n, we are going to write as e j of n, which should be equal to d j of n minus y j of n.

Now, here, because I am having more than one neurons at the output layers, so that is why I am numbering or indexing the neurons as such. So, that is why we are saying y j d j, so everything is pertaining to the jth neuron and the errors signal is as follows. So, what is going to be the instantaneous value of the error energy, the instantaneous value of the error energy is simply going to be E of n equals half of what e j n square. But, I said there are more than one outputs.

So, definitely we have to sum up, even if we are talking of instantaneous value of the error energy, at the nth iteration or the presentation of nth pattern, whatever errors are existing at the outputs of every neuron or every output neuron and they must be summed up. So, we should sum it up over j, j belonging to the output layer, so if we say j belongs to C, where the set C, set C includes all the neurons in the output layer. So, having defined this, the average squared error energy, if we try to think of the average.

(Refer Slide Time: 51:09)



The average squared error energy

$$E_{av}(n) = \frac{1}{N}\sum_{n=1}^{N} E(n)$$

N : Total no. of training patt.

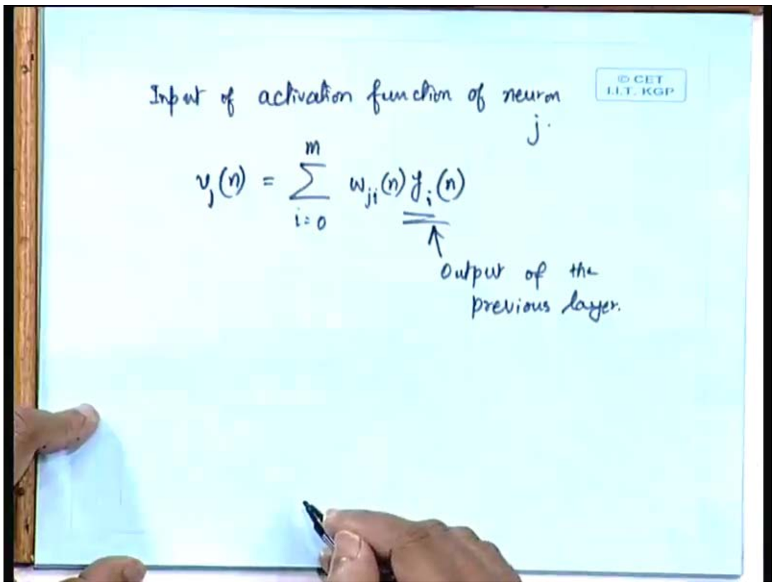
Then, if there are capital N number of total training pattern, let say, if there are n number of capital N number of training patterns. Then, the average squared error energy, that is E av n is equal to 1 upon N summation of E n, E n is what, I have defined just now, the instantaneous value. So, if this instantaneous value of this error, I sum it up for N equal to one to capital N where capital N is the total number of training patterns, that is one that composes one Epoch.

Now, there could be two different approaches, one approach is that on a simple pattern by pattern basis, you update the weight, like the way we did for the l m s algorithm. Or the other approach could be that you calculate this average squared error energy and based on this average squared error energy, you adjust the weight once for all. In fact, is going to be nothing but the minimization of this average squared error energy is just going to be an estimate of what the average weights that, we are going to get out of this.

You see here, at every iteration we minimize, for every pattern presentation, we minimize this E n functional and then we are going to get an updated weight. So, then over N number of patterns, over capital N number of patterns, if we take the average weight, that weigh and this weight will not be exactly the same. The weight that you are getting by minimizing, this E n of that will not be same, there will be some deviations.

So, the weight estimate that your are getting out of this is going to be some estimate of the true weight, that you will get by following the instantaneous approach. In fact, we will there is nothing wrong in following the instantaneous error approach only. So, I can end this class of today, because I am not being able to complete the total analysis, because of lack of time today.

(Refer Slide Time: 53:56)



Input of activation function of neuron j.

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) \underline{\underline{y_i(n)}}$$

$\uparrow$ Output of the previous layer.

In other words, what I can simply say is that, the total output, that I am getting out of this is for the jth neuron if I take, the v j at the nth iteration should be equal to summation of w j i at n times y i of n and you add it up for i is equal to 0 to m. Where, m is going to be the dimensionality, in fact we added i is equal to 0 to m, after embedding the bias terms also into it.

So, this is the complete term, that we are getting as the induced local field at the input of activation function of neuron j and this y is are what output of the previous layer, so these are the output of the previous layer. And in the case of multilayer Perceptron, it is necessarily the hidden layer, the hidden layer that immediately precedes the output layer.

Anyway, with this knowledge, we will continue our discussion, about the derivation of the expressions for the back propagation of errors, which is going to be very important. In fact, that is the underlying theory of the multilayer Perceptron and that we will do in the coming class.

Thank you very much.