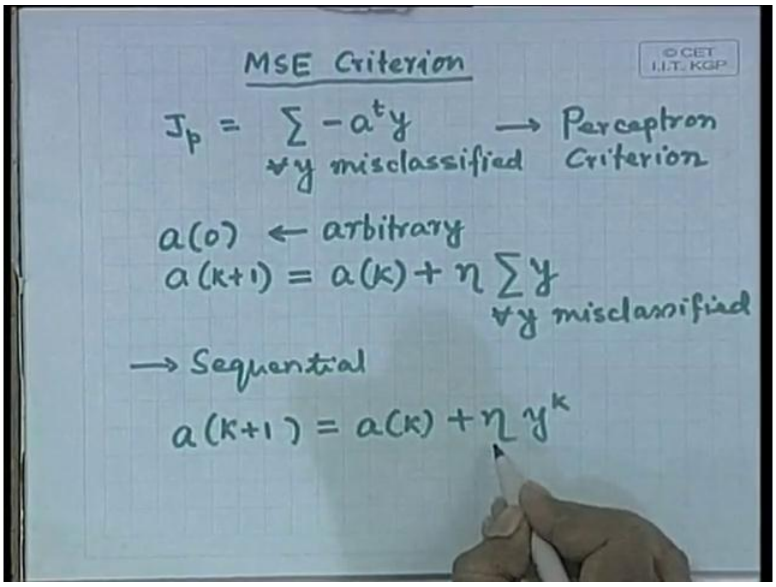
Pattern Recognition and Applications

Prof. P.K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Khargpur

Lecture - 22
Brief overview of the course

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)



MSE Criterion

$$J_p = \sum_{\forall y \text{ misclassified}} -a^t y \longrightarrow \text{Perceptron Criterion}$$

$a(0) \leftarrow$ arbitrary

$$a(k+1) = a(k) + \eta \sum_{\forall y \text{ misclassified}} y$$

$\longrightarrow$ Sequential

$$a(k+1) = a(k) + \eta\, y^k$$

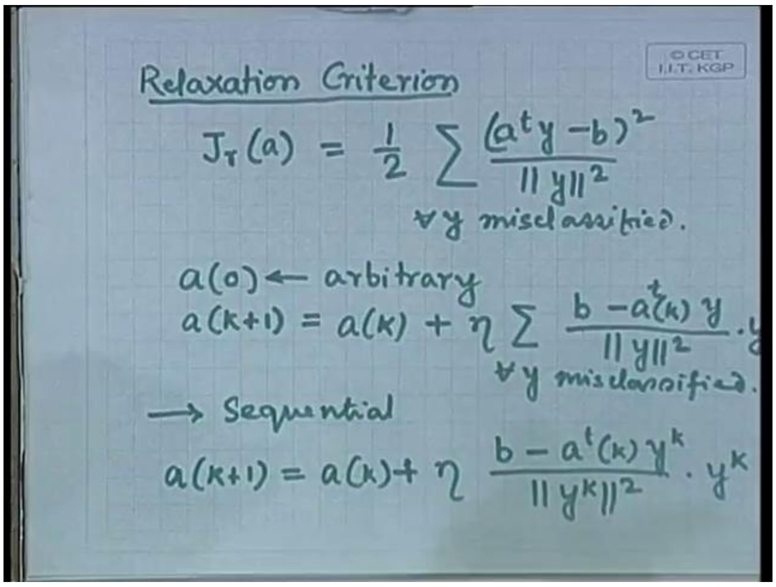
You define a criteria function which is given by J p which is nothing but minus a transpose y, for all y which are misclassified. So, this was the perceptron criterion function and our m was that we wanted to find out the weight vector a, which will actually classify all the training samples properly. 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 decision rule or the perceptronal algorithm which was given like this that a is 0 was initially selected arbitrarily and then in every iteration a K plus 1 was obtained from the previous value a K following the gradient descent procedure. It was simply a K plus eta times summation of 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. Here, 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 or 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 plus 1 will be a K plus eta times y K, where this y K is the sample which is misclassified by the weight vector a K and eta. 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.

$a(0) \leftarrow$ arbitrary

$$a(k+1) = a(k) + \eta \sum \frac{b - a^t(k)\, y}{\|y\|^2} \cdot y$$

$\forall y$ misclassified.

$\longrightarrow$ Sequential

$$a(k+1) = a(k) + \eta \frac{b - a^t(k)\, y^k}{\|y^k\|^2} \cdot y^k$$

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.

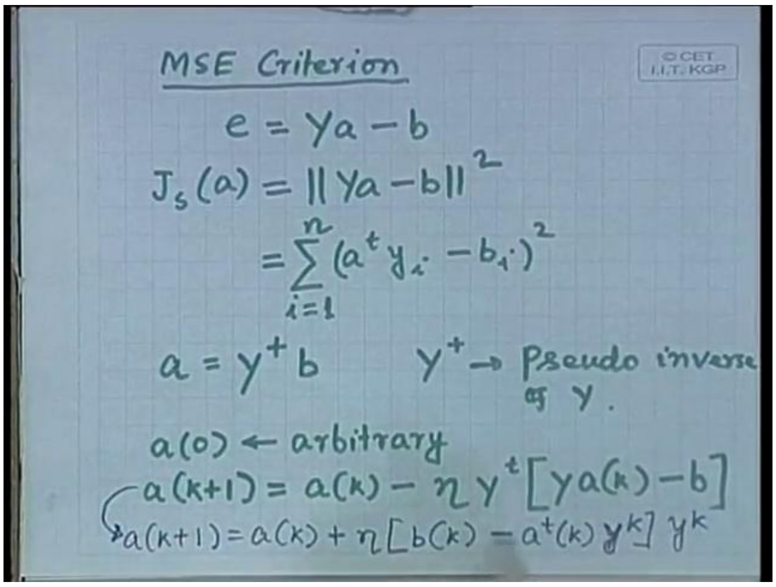
So, that problem is solved by this perceptron criterion y this relaxation criterion and this relaxation criterion the criterion function was modified as J r a which is half of a transpose y minus b square upon modulus of y square, again. Here, this vector y is misclassified, so it has to be done for all y misclassified, but our design rule remain 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 arbitrarily and, so we have this algorithm relaxational algorithm which is given as a 0 is chosen arbitrarily.

Then we have this iterative approach that is a K plus 1 will be given by a K plus eta times summation of b minus a K transpose y upon mod of y square into y for all y misclassified. Then we have seen that the corresponding sequential version where the updation rule is modified as a K plus 1 is equal to a K plus eta times b minus a K transpose y K upon mod of y K square into 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 omega 1 and omega 2, I should be able to draw a boundary between these two classes but 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 omega 1 and omega 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 ionic solution and the algorithm will never converse, so in such cases we have this minimum squared error algorithm.

(Refer Slide Time: 08:20)



MSE Criterion

$$e = Ya - b$$

$$J_s(a) = \|Ya - b\|^2$$

$$= \sum_{i=1}^{n} (a^t y_i - b_i)^2$$

$a = Y^+ b$ $\quad Y^+ \rightarrow$ Pseudo inverse of Y.

$a(0) \leftarrow$ arbitrary

$$a(k+1) = a(k) - \eta Y^t [Ya(k) - b]$$

$$a(k+1) = a(k) + \eta [b(k) - a^t(k) y^k] y^k$$

M S E criterion function, but the criterion function is based on an error measure where error is nothing but Y a minus b where y is the collection of all the training samples, a is the weight vector and b is called a merging vector. So, for these M S E criteria the aim is to reduce this square of this error minimize square of this error. So, accordingly you define the squared error criteria which is given by J s as a function of a, which is nothing but mod of Y a minus b squared. I can expand it as a transpose y i minus b i square of this take the summation over all the samples y is equal to 1, 2 n I have n number of training samples.

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 is equal to y plus b where this y plus is actually pseudo inverse of y. 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.

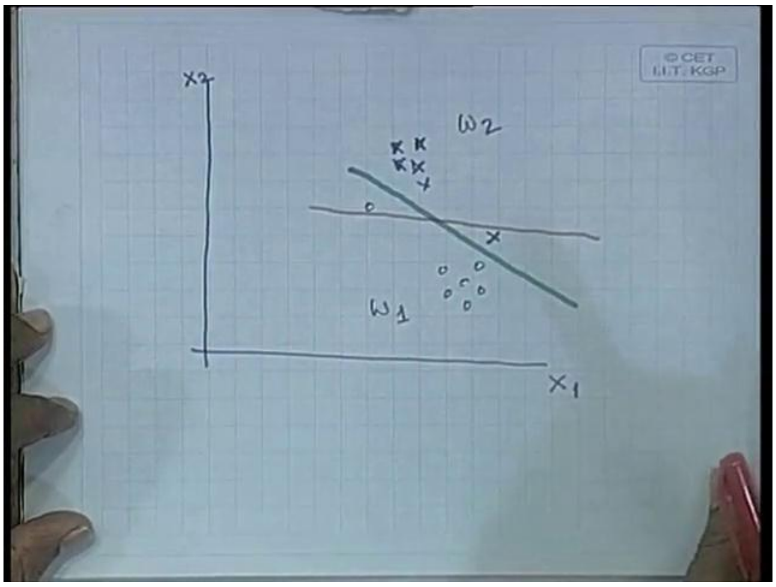
That is perceptron criterion relaxation criterion, here again or initial weight vector is 0 this will be chosen arbitrarily. In every iteration step a, a will be modified as a K plus 1 is equal to a K minus eta times transpose y a K minus b, but still you find that, here we are considering all the samples together to find out the solution vector. Here, 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 of the iterative algorithm.

The iteration step will be modified as a K plus 1 is equal to a K plus eta times b K minus K transpose y K into y K. So, find that we can go for a sequential algorithm from this updation step 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. The relaxation criterion is useful when the classes are linearly separable and if the classes are not linear linearly separable in that case we can go for this minimum based error criterion function.

But, if I have a problem where 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 criteria function, 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.

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)

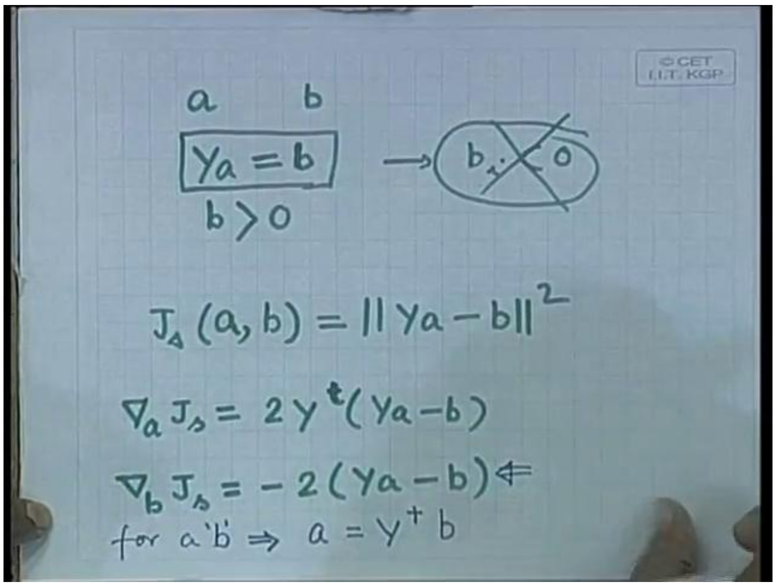


Even if I have the samples belonging to 2 classes which are given like this suppose this is a set of samples which actually belongs to say class omega 1 and I can have a set of samples over here. Let me use different symbol something like this where these samples actually belongs to class omega 2, here you find that I can easily, so these are mine two features x 1 and x 2.

So, I have a two dimension feature vector, so 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. It is 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 M S E or minimum squared error based criteria function you need to solve this kind of problem, where 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)



$a \qquad b$

$\boxed{Ya = b} \longrightarrow \cancel{b_i < 0}$

$b > 0$

$J_s(a, b) = \|Ya - b\|^2$

$\nabla_a J_s = 2Y^t(Ya - b)$

$\nabla_b J_s = -2(Ya - b) \Leftarrow$

for $a'b' \Rightarrow a = Y^+ b$

Weight vector a, and some weight vector b where y a minus b or Y a will be is equal to b where this b is greater than 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 a weight vector a and a merging vector b where this equation will be satisfied.

That is Y a equal to b, so when I say b is greater than 0 that means every component of b is greater than 0 because b are emerging 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 emerging vector b is greater than 0, then every component of this emerging 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 that 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 to maintain that ever component of b will be greater than 0. That can be satisfied if we start with b, with ever 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 skill factor I have Y a is equal to b. So, what I am saying is ill 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 where component of b say b i may be less than 0. So, this is what I will not permit ill always increment the value of b and the scale factor that is incorporated, that will be reflected in the corresponding value of x in the weight vector is that.

So, when I do that, I will redefine my criteria unction 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 merging vector. However, the definition will demand the same that J is a b which is y a minus b square fixed that was fixed in M S E algorithm. Our assumption was the merging vector was fixed and for a given merging vector, we are trying to find out a value of weight vector. So, the problem that we have faced that whenever we fix the merging 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.

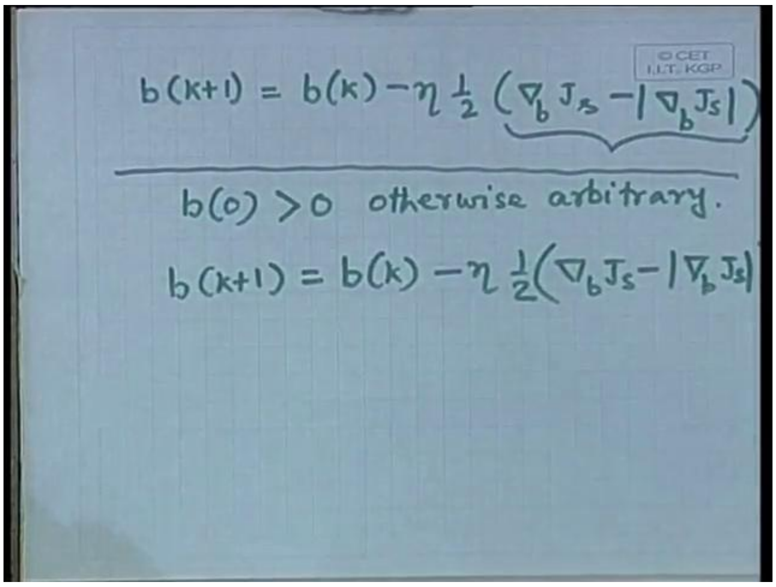
So, this boundary ensures that your sum of square error will be minimum, but this does not ensure that the classes are actually linearly separated. So, i-th classes are linearly separate separable, then we have said that we can have a solution of this form Y b minus is equal to b, where b is greater than 0. So, we will take the different merging 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 merging 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 is equal to Y a minus b take the square of the modulus of this error. 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 is the solution. The gradient with respect to a is what we have seen earlier the 2 y transpose into Y a minus 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 minus 2 Y a minus b know, 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 y pseudo inverse bore, 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 non singular because if the pseudo inverse is singular we cannot have if y transpose, y inverse is singular then we cannot compute the pseudo inverse.

So, for the computation of a i that we can go for this, if you assure that y transpose inverse is into 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. 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)



$$b(k+1) = b(k) - \eta \frac{1}{2}\left(\underbrace{\nabla_b J_s - |\nabla_b J_s|}\right)$$

$b(0) > 0$ 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 plus 1 which is same as the same gradient descent procedure became minus eta times that is rate of convergence of gradient of J s with respect to b minus modulus of gradient of J s with respect to b. So, here you find that because to obtain b K plus 1, we are subtracting from b K this quantity, so if gradient of J s with respect to b is positive. Then what we are able to ensuring s that this total quantity will be equal to 0 where as if gradient of J s with respect to b is negative.

So, here it is minus, here it is also minus, so I get a negative quantity which is twice of gradient of J s with respect to b. 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 plus 1 from b K to b K, I am adding a quantity I am not subtracting any quantity from b K to get b K plus 1. So, that insures that you are b K plus 1 can be utmost greater than b K it can never b less than b K.

So, by this step we are ensuring that none of the elements of the merging 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 merging 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 greater than 0, but otherwise arbitrary.

Student: Sir if we had grad b, J s modulus is it inner orbit?

Gradient of.

Student: Is magnitude of every component of gradient

It is magnitude of every component of gradient vector what I am saying is every component it is not modulus

Student: It is a vector.

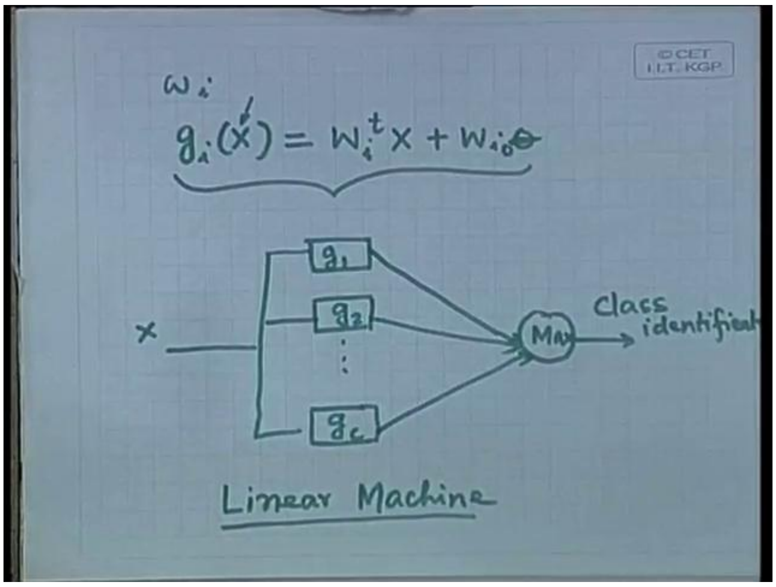
It is a vector, it is a vector only the positive components.

Student: The vector is minus b1 plus b2.

It will be class b 2, b 3 yes, is that so what the iterative algorithm will be something like this that b 0 will be greater than which is initially chosen. But, otherwise it is arbitrary and at every step this b K plus 1 will be b K minus eta times half gradient of b J s minus modulus of gradient J s with respect to b. 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 part that we have discussed in our last class.

By doing this because we are ensuring that b will always be grater, will always remain greater than 0 it is quite likely that it will even for lineally 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 criteria in all these cases. We have assumed that we have just two diff classes omega 1 and omega 2 and our criteria was that if a transpose y is greater than 0. The sample belongs to one class if it is less than 0 it belongs to another class or if a transpose y is greater than b it belongs to one class, if a transpose y is less than b it belongs to another class. Now, the question is how to generalize this in a multi category case using 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 mission that we have said earlier.

(Refer Slide Time: 29:12)



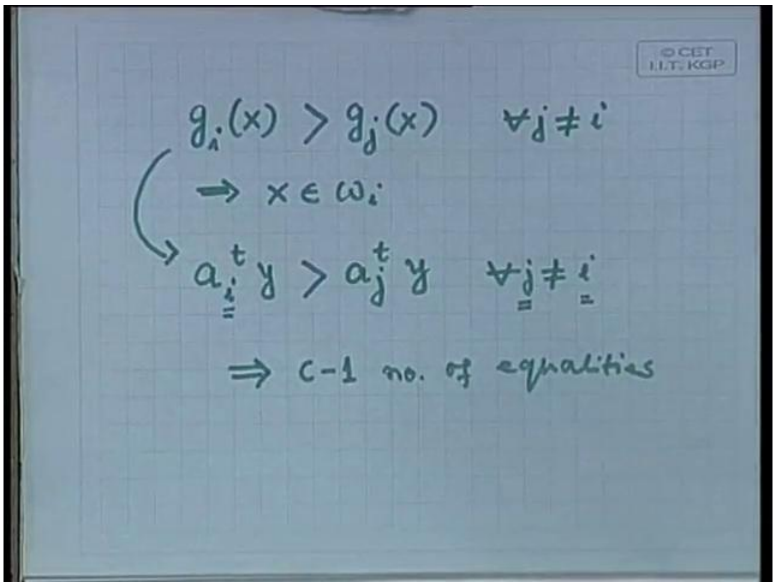
That is for every class omega i we have a discriminant function defined as j i x where this j i x is nothing but W i transpose x plus W i not sorry, so the linear equation or the linear discriminate function for every class i. So, for given any unknown sample x I have to compute this discriminant function, 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 x, 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 discriminate function g c then given a vector x the vector x will be fed to the discriminate function of all the classes then I have a maximum selector all these are ports come to this maximum selector. Then maximum selected depending upon whichever discriminate function gives you the maximum value this gives you the class identification. This is what we have said it is a linear machine and we have seen earlier that following the base decision theory if I know what is the parametric form of the probability functions, of the samples belonging to different classes.

Then following this minimum error classification or minimum r is j classification we can obtain the classifiers or we can obtain the discriminate functions for different classes. Then we have discussed about different situation when this discriminate function can be a linear discriminate function or when the discriminate function will be a quadratic discriminate function.

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 procedures 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)



$g_i(x) > g_j(x) \quad \forall j \neq i$

$\Rightarrow x \in \omega_i$

$\underline{\underline{a_i^t}}\, y > a_j^t\, y \quad \forall \underline{\underline{j}} \neq \underline{\underline{i}}$

$\Rightarrow c-1$ no. of equalities

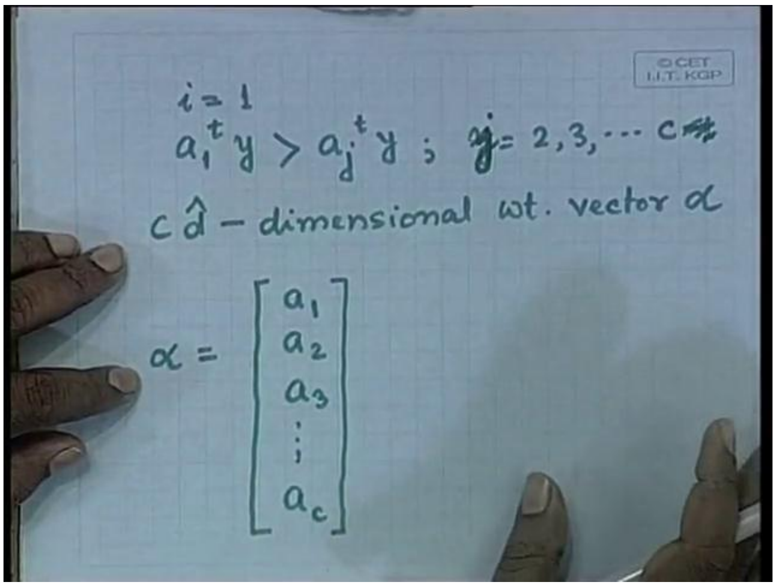
That if I get that g i x where x is the sample vector g i x is the discriminate function corresponding to the class omega i. So, if g i x is greater than g j x for all j not equal to i this indicates that x belongs to class omega i because g i x is maximum among all the discriminate functions. So, this indicates that the sample x belongs to class omega i or in other word if I expand this it simply becomes a i transpose y, where this y is the augmented vector x by adding 1 to it.

So, a i transpose y is greater than a j transpose y, again for all j not equal to i, so that is the condition that this sample x or sample y augmented sample y that belongs to plus omega a. 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 is not equal to 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 minus one number of values, so that one which is left out that is equal to i.

So, this inequality that a i transpose y greater than a j transpose y for all j not equal to i this is actually equivalent to c minus 1 number of inequalities is that, so in all those inequalities I have to have this particular condition 2 that suppose i is equal to 1. So, a 1 transpose y must be greater than a 2 transpose y, a 1 transpose y must be greater than a 3 transpose y, a 1 transpose y must be greater than a 4 transpose y and so on up to a transpose y must be greater than a c transpose y.

Then only I can say that this sample y or the corresponding sample x that belongs to class omega 1, so this equation actually corresponds to for all values of j c minus one number of inequalities. So, whether it is possible to have an unified approach to take care of this c minus 1 number of inequalities simultaneously and give me c number of weight vectors a one a two a three up to a c, so over here you find that.

(Refer Slide Time: 36:31)



$i = 1$

$a_1^t y > a_j^t y \;;\; j = 2, 3, \cdots c$

$c\hat{d}$ – dimensional wt. vector α

$$\alpha = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_c \end{bmatrix}$$

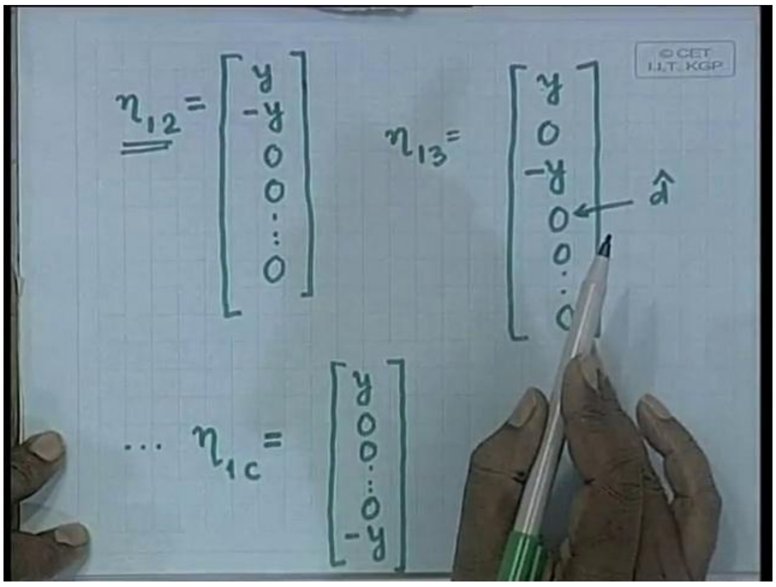
Let us take a specific case that is equal to 1, so I want that a 1 transpose y must be greater than a j transpose y, for y varying from 2, 3 up to c minus 1 later on we will generalize. So, I take a particular value of I which is equal to 1, so you find that all these c minus 1 number of inequalities can be put in an unified approach if I take c into d hat dimensional weight vector.

Student: The total number of equalities is c minus 1.

c minus 1, c minus 1 yes, sorry 2 to c correct, it is 2 to c not c minus 1 total number of inequalities is c minus 1, so all these inequalities can be put in an unified form in a single expression. If I consider a c d hat 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 plus 1 dimensional, so that d plus 1, we are talking we are taking as d hat. So, I take a d c into d hat dimensional weight vector let us say this weight vector is alpha, now what is this alpha this alpha will be nothing but a 1, a 2, a 3 up to a c 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 d hat dimensional I have c number of such weight vectors. So, that overall dimension of this vector alpha is c into d hat, thank you j equal to 2 to c, so this overall vector that becomes c into d hat dimensional. So, we can think that these particular weight vectors if I have this condition that a 1 transpose y greater than a j transpose y for all values of j varying from 2 to c. So, I can equivalently say that effectively this c into d hat dimensional weight vector alpha will properly classify all the c minus 1 number of feature vectors, but the feature vectors will be of the form.

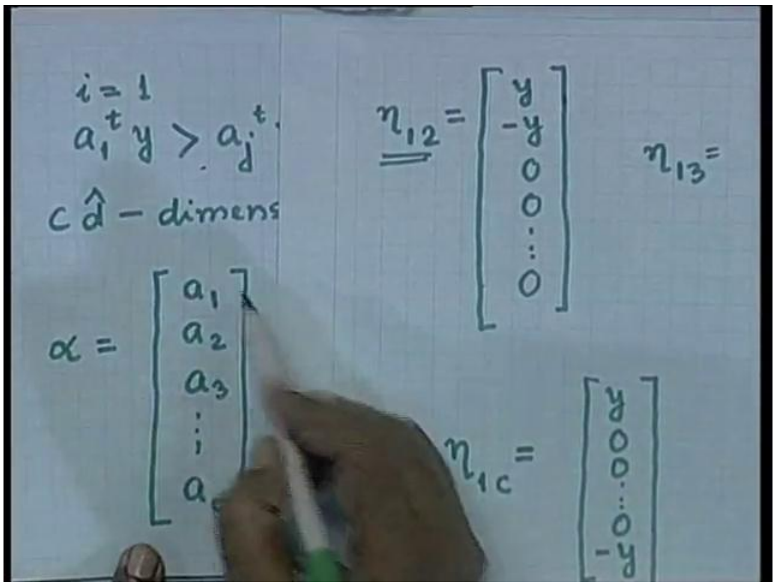
(Refer Slide Time: 40:08)



$\underline{\underline{\eta_{12}}} = \begin{bmatrix} y \\ -y \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$
$\eta_{13} = \begin{bmatrix} y \\ 0 \\ -y \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$
$\hat{d}$
$\cdots \eta_{1c} = \begin{bmatrix} y \\ 0 \\ 0 \\ \vdots \\ 0 \\ -y \end{bmatrix}$
© CET I.I.T. KGP

Say let me put it as eta one two which is nothing but y minus y 0, 0, 0 eta 1, 3 will be y 0, minus y 0, 0, 0 it will continue like this eta 1 c will be is equal to y 0, 0, 0 minus y. But, this 0 actually means that this is a column vector of dimension d hat y is obviously a column vector of dimension, d hat minus 1 y is also a column vector of dimension, d hat 0 is also a column vector of dimension d hat at every component is equal to 0. So, as there are c number of such vectors, every vector having a dimensionality of d hat, so the overall dimension of these vectors eta 1, 2 or eta 1, 3 up to eta 1 c each of them are of dimension c in to d i.

(Refer Slide Time: 42:08)



$i = 1$
$a_1^t y > a_j^t$
$c\hat{d}$ - dimens
$\alpha = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_c \end{bmatrix}$
$\eta_{12} = \begin{bmatrix} y \\ -y \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$
$\eta_{13} =$
$\eta_{1c} = \begin{bmatrix} y \\ 0 \\ 0 \\ \vdots \\ 0 \\ -y \end{bmatrix}$

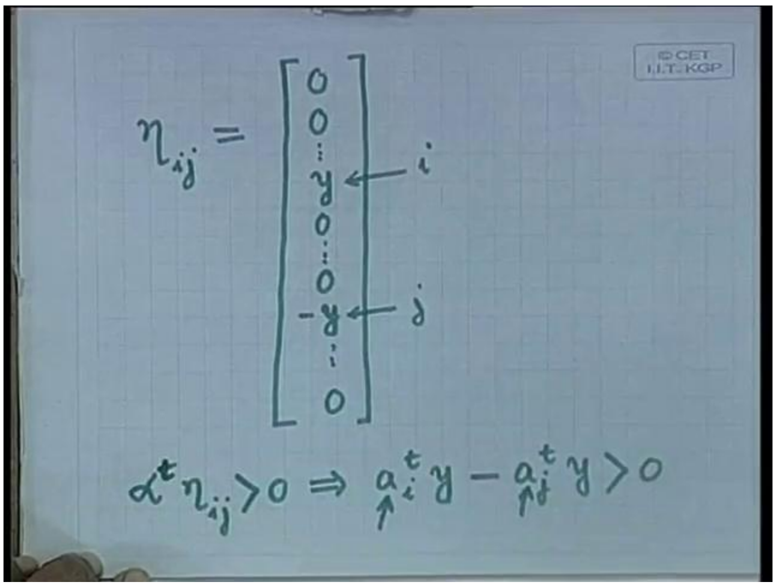
So, this inequality that a 1 transpose y greater than a j transpose y for j varying from 2 to c, it is equivalent to say alpha transpose eta 1 2 is greater than 0 is it not, if I put these two side by side. So, let us take these two vectors side by side alpha is y a 1, a 2, a 3 up to a c eta 1 2 is y minus y, then all components to 0. So, if I take alpha transpose eta 1 2 that computation will be equivalent to a 1 transpose y minus a 2 transpose y rest of the components in eta 1, 2, 0.

So, that corresponding vector multiplication will have null effect will not have a any effect on the result is that, so if I take alpha transpose eta 1 2 that is simply a 1 transpose y minus a 2 transpose y. Similarly, alpha transpose eta 1 3 that will be simply a 1 transpose y minus a 3 transpose y is that, similarly alpha transpose eta 1 c it will be a simply a 1 transpose y minus a c transpose y. So, this condition that a 1 transpose y greater than a j transpose y or a 1 transpose y minus a j transpose y greater than 0 for all the values of j for y to be classified to plus omega 1. That is equivalent to saying that this weight factor alpha properly classifies all these c minus 1 number of feature vectors.

But, 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. 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 minus 1 number of feature vectors, but the dimensionality of the feature vectors each of this feature vectors is also being implemented by a factor c initially y vector y is of dimensionality d hat.

Now, each of them are of dimension c in to d hat, so the dimensionality is increasing by the number of pluses that we have and the number of vectors for every original vector is also being increased by a factor c into c minus 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 is equal to 1.

(Refer Slide Time: 45:49)



© CET I.I.T. KGP

$$\eta_{ij} = \begin{array}{c} \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ y \\ 0 \\ \vdots \\ 0 \\ -y \\ \vdots \\ 0 \end{array}\right] \end{array} \begin{array}{l} \\ \\ \\ \leftarrow i \\ \\ \\ \\ \leftarrow j \\ \\ \\ \end{array}$$

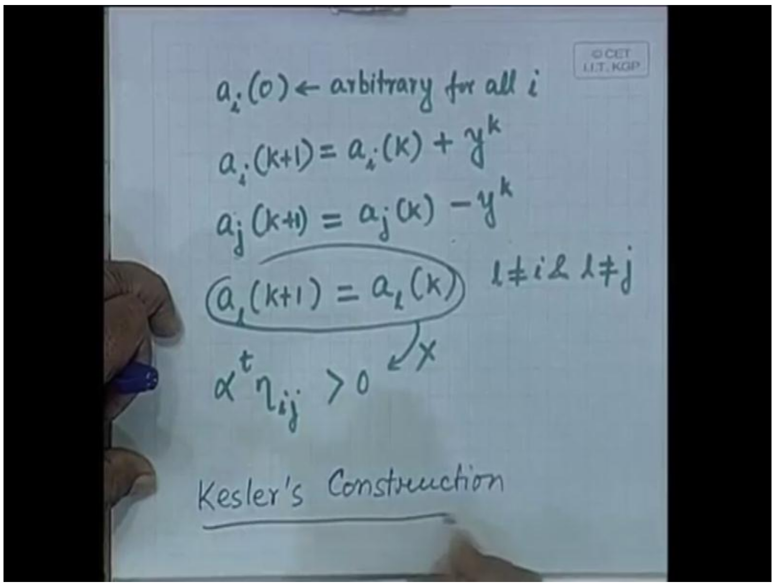
$$\alpha^t \eta_{ij} > 0 \Rightarrow \underset{\uparrow}{a_i^t} y - \underset{\uparrow}{a_j^t} y > 0$$

In general, a feature vector eta i j will be something like this that I will have this vectors having component 0 i-th component will be equal to y j-th component will be minus y. Then for this 0 again this is my i-th component and this is my j-th component is that and this alpha transpose eta i j greater than 0 that is equivalent to if you follow from here it will simply be a i transpose y minus a j transpose y that will be greater than 0. So, if this condition is satisfied for all values of j not is equal to i then we say that the sample y actually belongs to class omega i it does not belong to class omega j.

So, when I have this sort of formulation that alpha transpose eta i j for every such inequality I actually have the weight vectors corresponding to discriminate functions of two different classes. Omega i and omega j involved in it and for proper classification of any sample, y i must have that it alpha transpose, eta i j must be greater than 0 that means a i transpose y minus a j transpose y must be greater than 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.

But, I will also have to modify a j because both of them are responsible in earlier case because the 2 plus 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 omega i and the other one of that discriminate function of class omega j. So, any time any sample is misclassified by alpha I have to what modify both these weight vectors a i and a j involved that particular eta i j, 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)



© CET I.I.T. KGP

$a_i(0) \leftarrow$ arbitrary for all i

$a_i(k+1) = a_i(k) + y^k$

$a_j(k+1) = a_j(k) - y^k$

$\boxed{a_l(k+1) = a_l(k)}$ $l \neq i$ & $l \neq j$ ✗

$\alpha^t \eta_{ij} > 0$

<u>Kesler's Construction</u>

That a i 0 is arbitrary for all i that is for every class I assume an arbitrary weight vector initially and then for any vector which is misclassified. So, with the help of this initial weight vector i have to form initial alpha which is just concatenation of all this different weight vectors. Using that I have to go for classification of the samples are the samples to be modified in this form whenever I know that our sample belongs to class omega i.

So, for every such i I have to generate eta i j, but the i-th component will be equal to i and the j-th component will be minus y. So, I have to comp generate eta i 1, I have to generate eta i 2, I have to generate eta i 3, I have to generate up to eta i c. So, for each of these samples I have to generate c minus 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 alpha.

If any of those samples is not properly classified by alpha then I have to modify both a i as well as a j, so that weight updation rule will be a i K plus 1 will be is equal to a i K plus y K. This is the sample which is misclassified and this is what we have obtained using our percentile criteria this expression is same. So, we are incrementing a i in particular direction a j has to be incremented in a negated direction because we want to analyze a j.

Else we want to revert the i and a j K plus 1 for this the weight updation rule will be this will be simply a j K minus y K, so I find that only two weight vectors. So, updating one is a i and a j for every eta i j which is misclassified and all other weight vectors will say that a l K plus 1 will remain as a l K where l is not equal to i and l is not equal to 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 percentile 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 analyzed all other weight vectors other than a i and a j, they remain unchanged because these weight vectors really does not take part to tell us. Whether alpha transpose eta i j is greater than 0 or not these weight vectors do not tell us anything about this only the weight vectors say i and i j decide whether this condition will be satisfied or not is that. So, we are keeping a l K plus 1 is equal to 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 with vector alpha.

For every sample y we are constructing alpha eta i j 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 transpose y minus a j transpose y greater than 0, so increment one reduce the other one. So, that this condition is reached quite fast is that if you do not do this will get 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, Khargpur

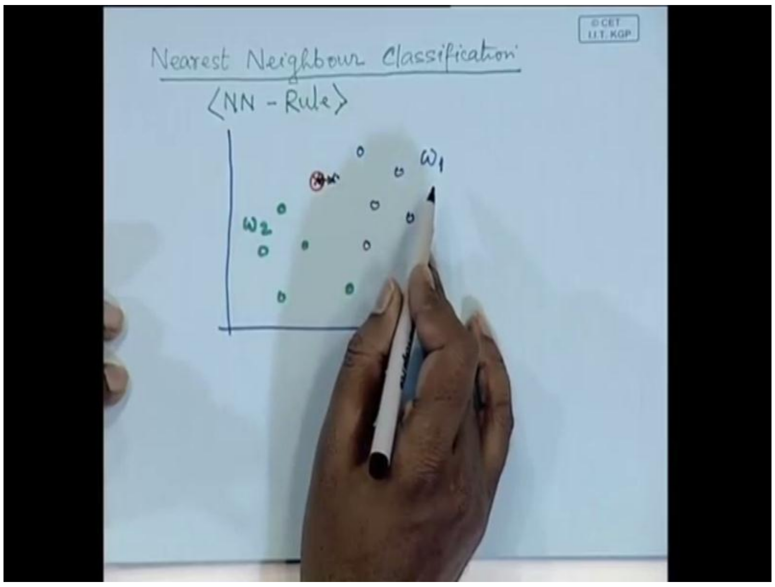
Lecture no. # 17

Neural Network for Pattern Recognition

Good morning, so today we are going to start our discussion on use of neural network of pattern recognition, so far what we have discussed. Initially we started discussing about the base 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.

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)



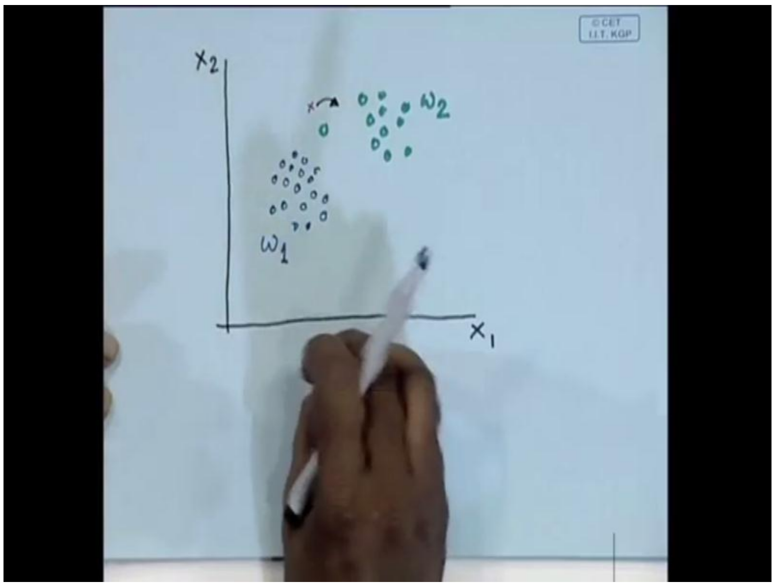
Nearest neighbor classification or it is also called N N rule, so the only use this nearest neighbor classification technique what you do is instead of trying to find, design any classifier or any discriminate function. For any of the classes what I have is, I have a set of samples that samples coming from different classes, so if it is a 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 on all other training sample 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 omega 1 and these are the samples which belong to class omega

2, I just find out that which sample is nearest to this unknown sample. So, 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 omega 1. So, I simply classify this unknown sample to class omega 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.

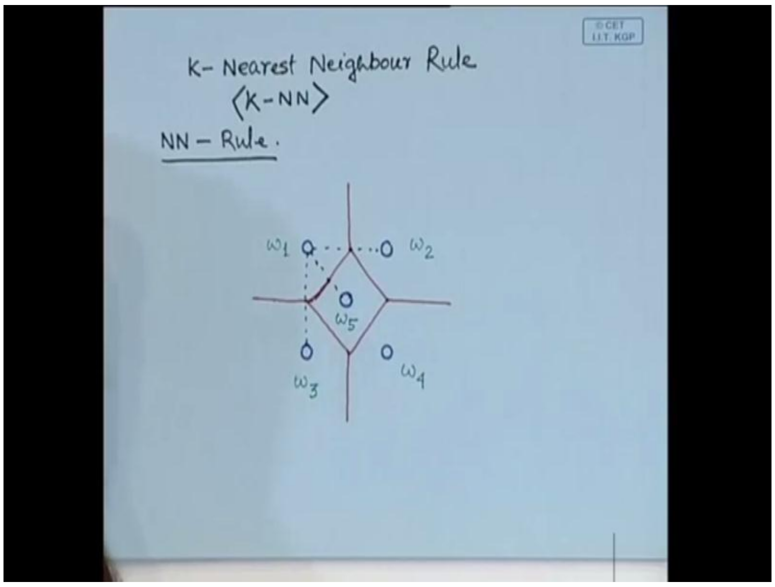
(Refer Slide Time: 01:05:54)



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 omega 1 and these are the training samples taken from class omega 2 and in over here if the unknown sample is somewhere at this position. So, find that as before this unknown sample will have a minimum distance from this training sample which belongs to class omega 2. So, I will classify this sample to belong to class omega 2, however you find that here I have a single point a single sample from class omega 2.

Whereas, I have more number of samples from class omega 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 the class. So, a solution to this is that instead of taking a single point which is nearest to the unknown sample, you take a piece 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 N N 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.

Student: Sir, how will that is K number of samples from omega 1 class or omega 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 omega 1 or it belongs to class omega 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. 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 coming to this same example. Say, this is the position of the unknown sample and these are the samples which are from class omega 1, these are the samples from class omega 2. 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 omega 1 or it is coming from class omega 2 irrespective of that I have to find out K number of samples which are nearest to this unknown sample.

So, suppose I decide this value of K is equal to 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 omega 1 and I have only one sample which is coming from class omega 2.

So, it is class omega 1 which is maximally represented in those 5 training samples, so I have to classify this unknown sample x, to class omega 1 rather than class omega 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 omega 2 or as if I take 5 nearest samples and out of those 5 I find out which class is maximally represented.

Here, it is class omega 1 which has 4 samples output, the 5 or as class omega 2 has got only 1 sample out of 5, so accordingly this unknown sample is classified as class omega 1 rather than class omega 2. Now, naturally if I go for the nearest neighbor classification where value of K is equal to 1, the error rate or the errors bound will obviously more than error bound that you get by using base decision. Base minimum error classifier because in case of base minimum error classifier we classify an unknown sample to a class for which the probability of a particular class is maximum.

In case of a single 1, N N 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 base minimum error classifier or else. 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 N N classification approach is the error bound of base 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 tends to be in 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 N N classifier approach is that of the base minimum error.

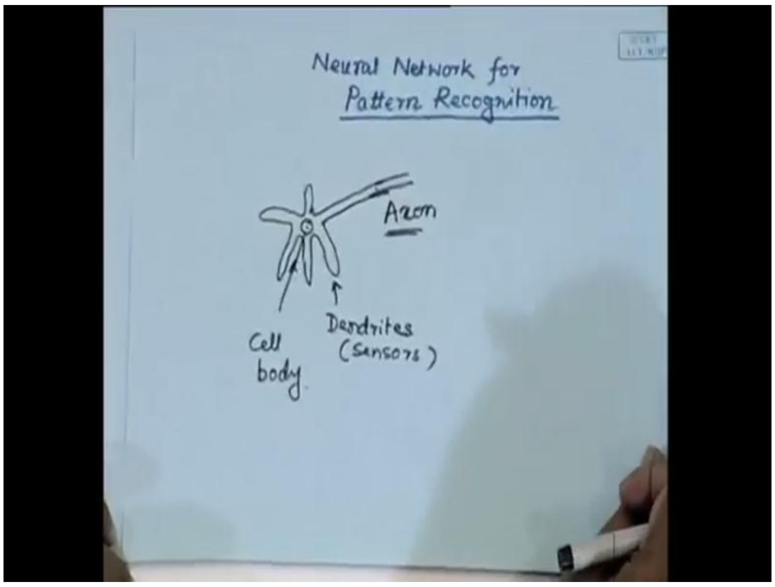
Classifier approach is that of the base minimum error classifier, now following this if I go for simply nearest neighbor rule, I should be able to find out what is the decision boundary between different classes. So, let us take a very simple example something like this say I have a number of samples 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 omega 1, this sample belongs to class omega 2, this sample belongs to class omega 3, this one belongs to class omega 4 and say this one belongs to class omega 5. Now, following nearest neighbor rule of I want to find out what is the decision bounded which separates all this different classes.

So, 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 or 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 or those 2 representatives.

So, following that you find that the decision boundary between class omega 1 and omega 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 omega 1, omega 2 and it will be extended on this side. Now, coming to the decision boundary between class omega 1 and omega 5, that will also be an orthogonal bisector of the line joining omega 1 and omega 5, so this decision boundary will be simply this. Similarly, the decision boundary between classes omega 1 and omega 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 omega 3 and omega 5 will be this between omega 5 and omega 4. It will be this between omega 2 and omega 4 the decision boundary will be this between omega 3 and omega 4 the decision boundary will be this. Between omega 2 and omega 5 the decision boundary will be like this is it 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. So, these are the different non parametric techniques for classified design or for classification of unknown samples to one of the known classes.

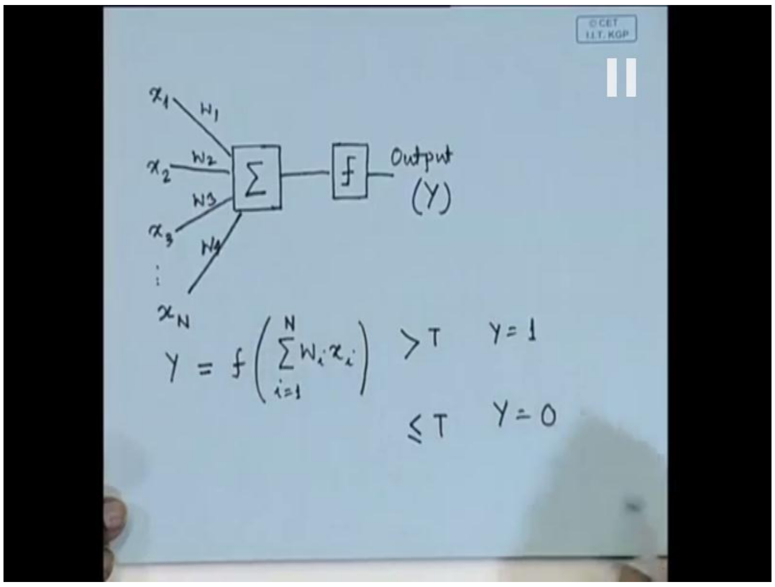
(Refer Slide Time: 01:20:54)



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 of 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 5, 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 this 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 in information to or different muscles through a set of such neurons. 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 functional is 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)



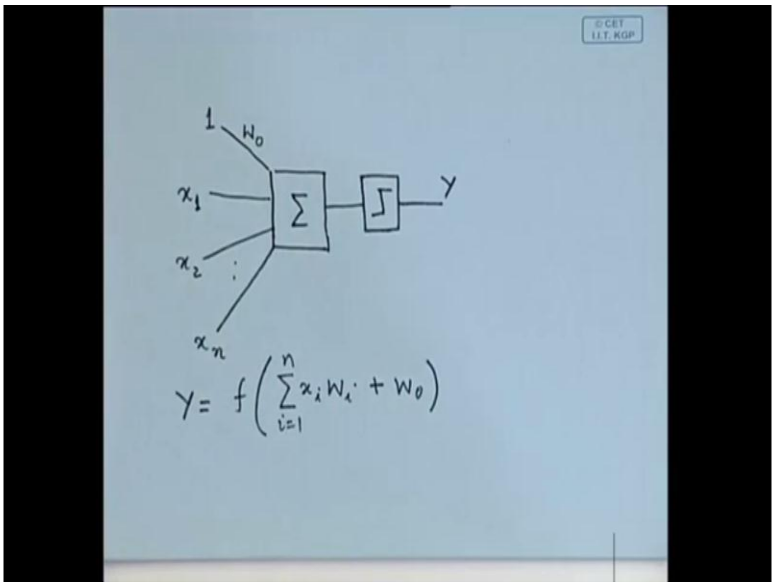
Say I have to have something like a cell body and then I have to have a set of inputs which act as dendrites and then I have to have an output which will act as the axon. So, what I have is I give the feature vector as input to this neuron and our feature vectors are having different components x 1, x 2, x 3 up to say x N. If I have N dimensional feature vector then the cell body performs a weighted summation of these different feature components and weighted sum is read as an input of a non linear function. In most of the cases this non linear functions is some sort of threshold function or it may be account inverse non linear function as well as different types of neural networks used, different types of non linear.

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 summation of 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 simple threshold function then what I will have is if this non linear 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 were designed earlier.

That is we have said that g i x is greater than 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 the summation of W x i, i varying from 1 to N. Where my feature vector is N dimension and if this function is greater than threshold I said y is equal to 1 indicating that this x belongs to 1 class. If this is less than or equal to threshold I set Y is equal to 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 becomes equal to 0, I will not take any axon, whereas if it is less than 0, then only ill set y equals to 0.

So, accordingly this function will be different from threshold function will be equal to will be different where if this weight age sum becomes equal to 0, then I will not take any axon. However, in case of linear 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. This purpose of this modified feature vector is that I can incorporate a bass weight with a different corresponding to different classes, that is W i 0, I can also incorporate the same bias within a linear model what I do is I can simply.

(Refer Slide Time: 01:28:52)



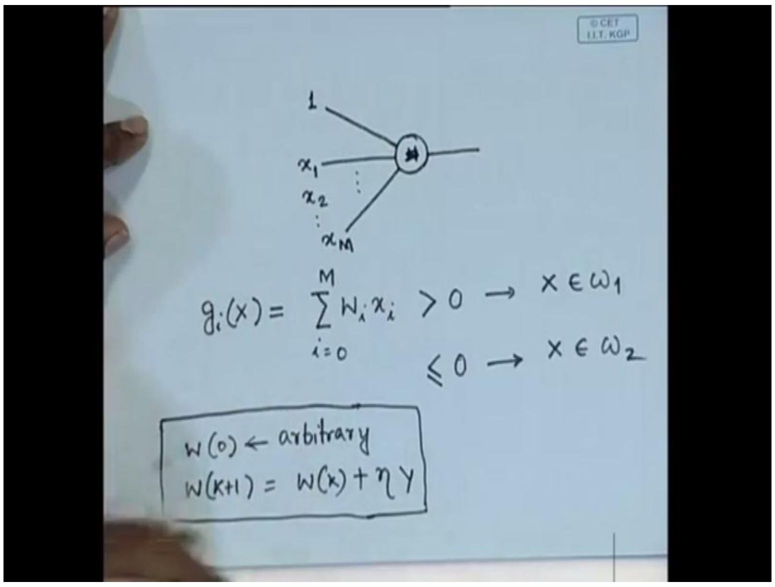
So, let me draw the same neural model where I have this addition and I have different inputs, one of the inputs I can put equal to 1 constantly and the weight given to this corresponding input can be W not. Whereas, to other inputs I have the different components of the feature vectors that is x 1, x 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. So, where here you find that output of this is summation of x i into W i, i varying from 1 to N plus W not 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 percent criteria relaxation criteria mean square derive criteria minimum square derive 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 a W, i-th 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 perceptron, so in the simplest case I can have a single output. So, single perceptron which is nothing but a single neuron like this, so in case of a single output single perceptron, the neural network is simply this.

(Refer Slide Time: 01:32:29)



1
x_1
x_2
x_M
$g_i(x) = \sum_{i=0}^{M} W_i x_i > 0 \rightarrow X \in \omega_1$
$\leq 0 \rightarrow X \in \omega_2$
$W(0) \leftarrow$ arbitrary
$W(k+1) = W(k) + \eta Y$
© CET I.I.T. KGP

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. So, I am, so node gives me this particular output for a given input, so a single output single perceptron is a single node something like this.

So, which is a 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 bass 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. Here, my g i x what this node compute is simply summation of 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 omega 1.

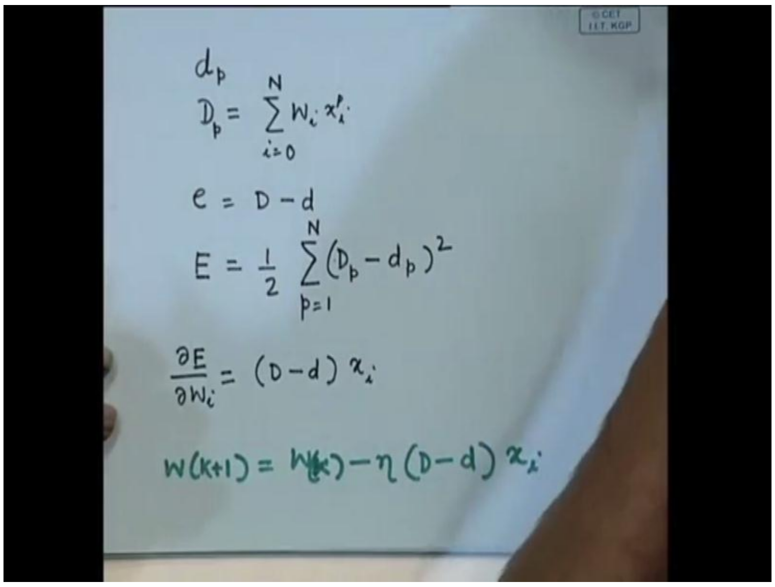
If this is less than or let us include equal also if it is less than or equal to 0, we say that the sample x belongs to class omega 2. Now, in case of perceptron criteria we have said that this weight updation rule was that initially we decide W 0 to be arbitrary when we had the perceptron criteria initially decided that W 0 the weight. Initial weight is chosen arbitrary then at every iteration stage the weight was chosen as W K plus 1 was equal to W K plus eta times 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 cross levels of the training samples then for attaining sample, I know what should be the corresponding output.

Say if the, if a training sample x belongs to class omega 1, I know that output should be equal to 1, if the training sample belongs to class omega 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 weight 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 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 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

$$D_p = \sum_{i=0}^{N} W_i x_i^p$$

$$e = D - d$$

$$E = \frac{1}{2} \sum_{p=1}^{N} (D_p - d_p)^2$$

$$\frac{\partial E}{\partial W_i} = (D - d)\, x_i$$

$$W(k+1) = W(k) - \eta\, (D - d)\, x_i$$

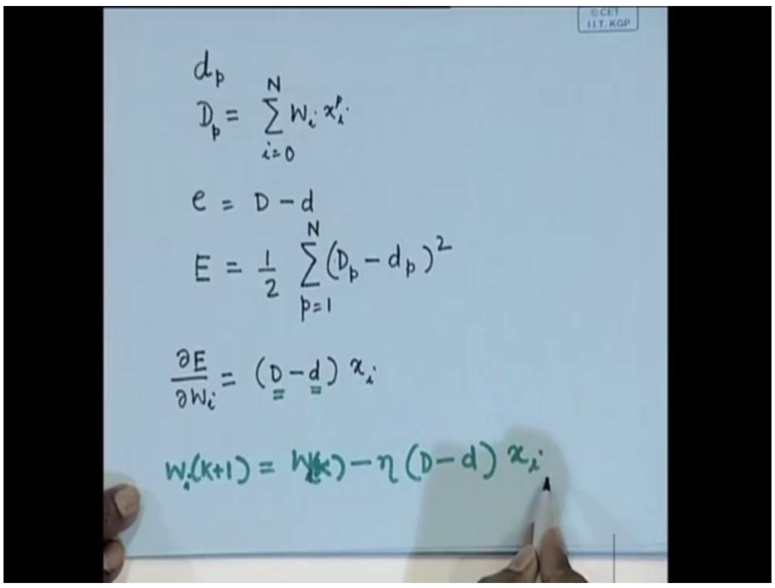
If I know that my output the target output is say d or as the actual output that is get is nothing but summation of W i x i where i varies from 0 to N. So, this is my target output, this is the actual output that I am getting, so the error is nothing but D minus d, I can have sum of squared error from this. So, I can say that sum of squared error, let me write it like sum of E is nothing but, half of summation d let me say that this is for p-th sample. So, the p, so I will have D p minus d p where p varies from 1 to N or N is the total number of training samples, so every sample i have an error take the summation of that error summation of square p f.

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. So, the simplest approach that can be taken is you try to minimize this by differentiating this with respect to W and as W has got M plus 1 number of components. So, by differentiating this with respect to W and equating that to 0, I will get M plus 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 that 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. For that what we have to do is we have to go for gradient decent procedure pardon 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 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 it with respect to or weight vector W. So, if I take the gradient of this say dell E upon say del W i which is the i-th component of the weight vector where this i varies from 0 to M as there are N number of M plus 1 number of components. Including that one that we have added as bias, so this dell E dell W i if you compute this it will simply become D minus d into x i. If you differentiate this with respect to W i then what i get is this D minus d is the actual output that I get lower case d is the target output, so D minus d is the error that I can easily compute into x i.

(Refer Slide Time: 01:42:49)



©CET I.I.T. KGP

d_p

$$D_p = \sum_{i=0}^{N} W_i x_i^p$$

$$e = D - d$$

$$E = \frac{1}{2}\sum_{p=1}^{N}(D_p - d_p)^2$$

$$\frac{\partial E}{\partial W_i} = (\underline{\underline{D}} - \underline{\underline{d}})\, x_i$$

$$W_i(k+1) = W_i(k) - \eta\,(D-d)\, x_i$$

So, accordingly or weight updation rule will be W K plus 1 is equal to W K following that gradient procedure minus eta times D minus d into 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 plus 1 is equal to W K plus eta times y where y is the vector. So, the entire weight vector was modified simultaneously and in this case it is that, know 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 into x i. So, what is the differentiation dell E dell W i is nothing but if I expand this whole term because this is square, I will have 2 into this into dell W i of D p minus d p summation.

Student: Sir, this for p-th sample and for p-th sample.

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 minus d.

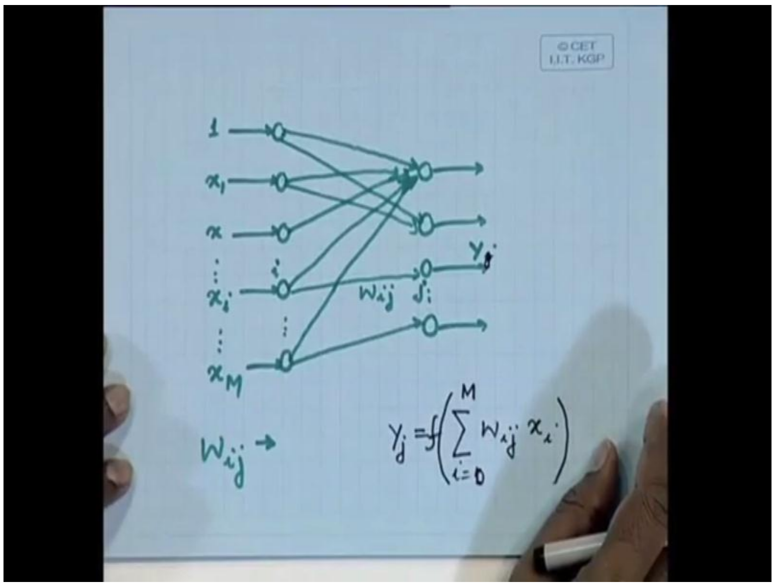
Student: So, in that above equation there is the possibility that p-th.

This is 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 or the weight updation rule was this over here the weight updation rule is W K plus 1 is equal to W K minus eta times D.

But, sorry this will be about i-th component because I am talking about i-th component W i K plus 1 is equal to W i K minus eta D minus d into x i where x i is the i-th component is that. So, this is what I have in case of a single output single perceptron this 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 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 plus if the output belongs to becomes 0 I will say the sample belongs to another class. But, if I have multiple plus 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 fill 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 is the feature components. These are fed to input layer which is simply crossed 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 and these are my outputs, so I represent the connection weights as W i j. So, this W i j 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 i j 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.

So, this is represented by W i j, so accordingly you find that if this output i say Y j, so this Y j will be nothing but Y j summation of W i j x i, i varying from 1 to or 0 to 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 i j, so the output layer what I get is W i j into x i. This also inputs the bias because I have started i equal to 0 M, so i equal to 0 means this W 0 j this connection component will actually correspond to the bias or bias way.

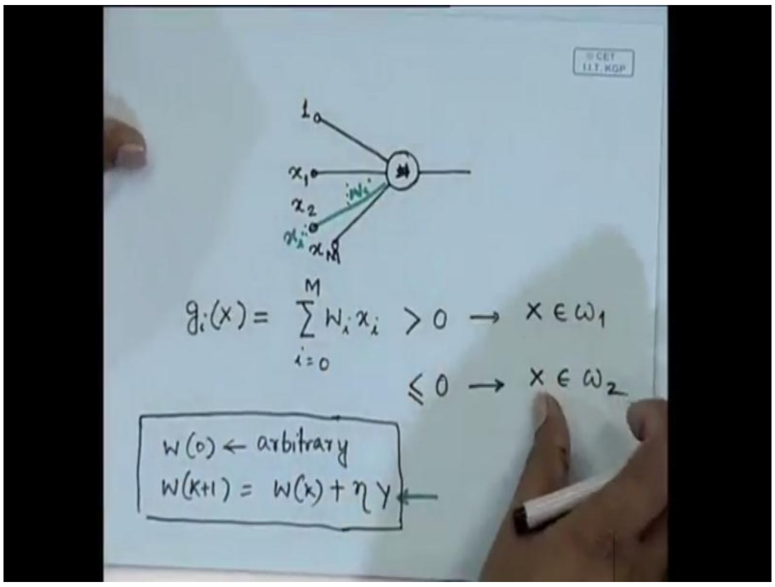
So, I have this Y j which is a non linear function of sum of W i j into x i where i varies from 0 to M in some cases. As we said earlier that this non linearity can be a threshold function, in other cases the non linearity can also be a sigmoidal function, but the sigmoidal function is.

Student: Over here input in node is only single, it may be multiple links.

Pardon

Student: In the first column of the notes why there is single input is having, in previous case we have multiple are there.

(Refer Slide Time: 01:52:00)


1_0
x_1
x_2
x_i
x_M
w_i
$g_i(x) = \sum_{i=0}^{M} w_i x_i > 0 \rightarrow x \in \omega_1$
$\leq 0 \rightarrow x \in \omega_2$
$w(0) \leftarrow$ arbitrary
$w(k+1) = w(k) + \eta y$

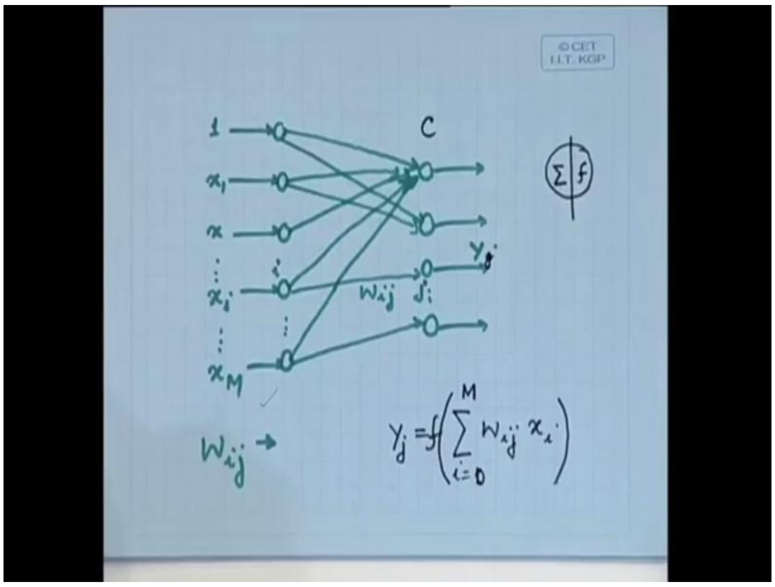
You are talking about this one, here this is my say actually each of these are actually nodes because when I am feeding this inputs where do I feed these are actually nodes which are unit again nodes. That means whatever is fed to the input that is simply passed to output when I say that the number of layers this input layer is not really specified because this comes by default.

Student: 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 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 weight. So, accordingly the input to each of the output layer node that is input to the j-th output layer node is this part of W i j 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 followed by for which we have defined I have an adore unit followed by a non linearity. So, I can say that each of these nodes actually are broken into 2 parts, one part computes the summation which is this followed by non linearity f that is not layer 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 liner equation of this form.

This linear equation is equation of single straight line is that if the point when it is greater than 0 means a point which risen 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 line, 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 W i j into x i, i varying from 0 to M for a particular j it gives me equation of one straight line.

For another value of j it gives me equation another straight line, so when I have, say on this output layer that if I have a c class problem I have C number of output layer nodes every node corresponding to one class. 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 4 are given input which of these output layer nodes gives you the maximum output.

This 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 coming to a linear seperability because each of them are actually representing your linear equations. So, it assumes that the classes are linearly separable what you do in 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 cases. So, piece wise linear approximation is required, so I will come to that later on, let us stop here today.

Thank you.