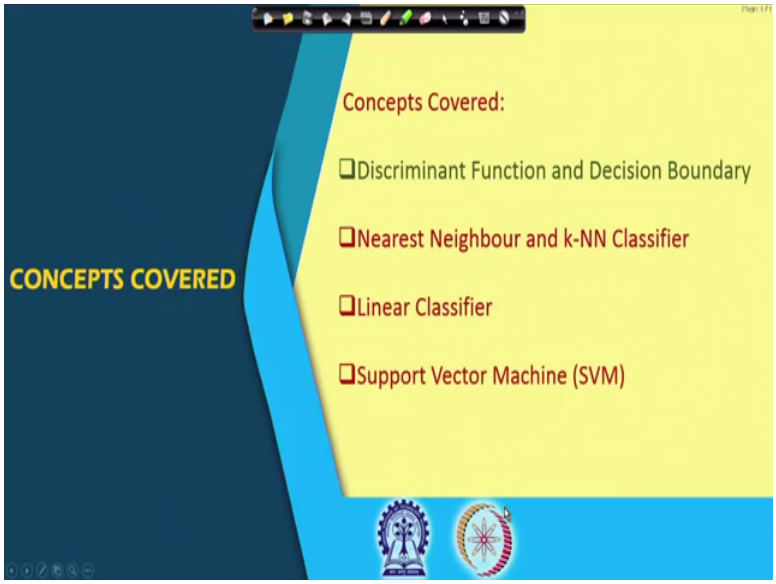
Deep Learning
Prof. Prabir Kumar Biswas
Department Of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 09
Linear Classifier

Hello welcome to the NPTEL online certification course on Deep Learning.

(Refer Slide Time: 00:35)



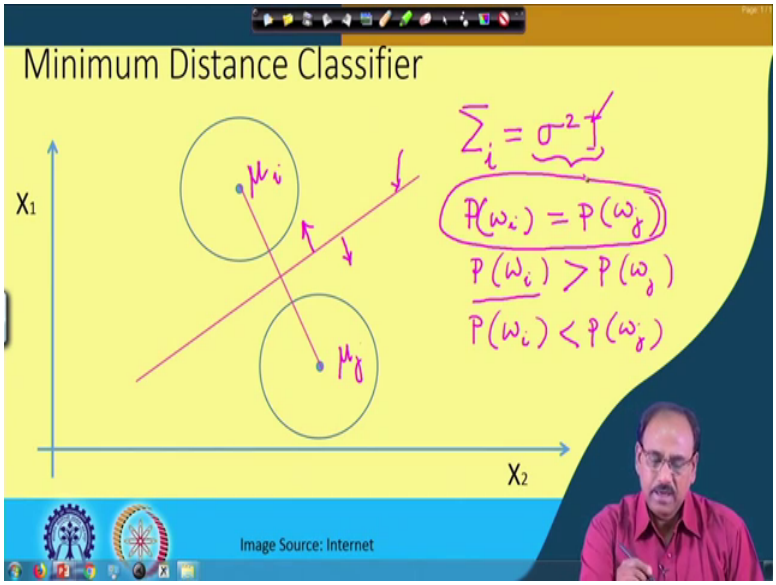
You remember in the previous class we have talked about the discriminant functions and we have also seen the decision boundaries. So, we when we discussed about the discriminant function and then the decision boundary we have assumed the vectors representing the objects that follow certain probably density function of certain distribution. And the distribution that we have assumed in this case was a normal distribution.

So, it was a multivariate normal distribution and based on that based using this multivariate normal distribution, under different assumptions of the covariance matrix we have seen that we can have the discriminant functions which are linear we can have discriminant functions which are quadratic.

And accordingly when we try to compute the decision boundary between the vectors or the patterns belonging to two different classes, the boundary can be either a linear

boundary or the boundary can be a quadratic boundary that depends upon what type of covariance matrix the distribution exhibits. Now, today we will talk about the linear classifier. And we will also talk about the support vector machine, before talking about the linear classifier and the support vector machine we will briefly touch upon another two different types of classifiers which are nearest neighbour classifier and k nearest neighbour or k-NN classifier. So, let us first talk about what is nearest neighbour classifier or nearest neighbour rule.

(Refer Slide Time: 02:20)



So, before going to that you find that in previous two lectures when we talked about the discriminant function which leads to the decision boundary between the two different classes, in a particular case that when the covariance matrix of all the different classes are of the form sigma square I. That means, in this case the different components of the vectors where statically independent and all the components have same variance which is equal to sigma square. So, there the covariance matrix for all the classes are same and which is in the form sigma square I, where this I is an unity matrix.

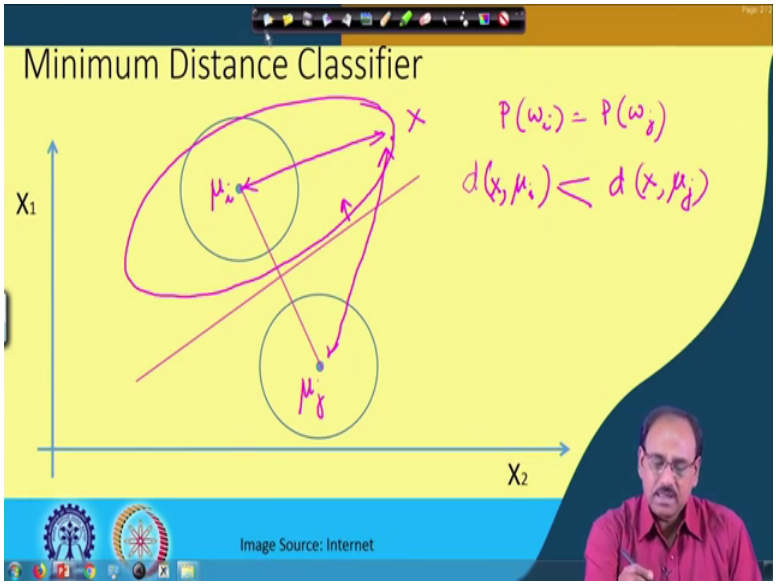
Under this and when the a priori probability of the classes P omega i was equal to P of omega j there we found that the decision boundary between the classes omega i and omega j was a linear boundary. And, not only that it was a perpendicular or orthogonal bisector of the vector of the line joining the main points mu i and mu j. So, if this is mu i and this is mu j then the line joining mu i and mu j is bisected orthogonally by the

decision boundary between these two classes. So, this was one of the case in which I can separate 2 classes by a linear boundary. And of course, in this case if P omega i is greater than P omega j then this decision boundary is shifted towards mu j.

In the sense that for unknown feature vectors our decision will be bias towards omega i because, P of omega i that a priori probability P of omega i is greater than P of omega j. In the other case if P of omega i is less than P of omega j in that case the decision boundary shifts towards omega i. It remains orthogonal to the line joining omega i and omega j, but it shifts towards omega i indicating that what decision for unknown features vectors will be biased in favor of class omega j.

So, particularly in this case when P omega i is equal to P omega j then my decision boundary is orthogonal bisector of the line joining mu i and mu j which clearly indicates that if I have an unknown feature vector say X over here which I need to classify.

(Refer Slide Time: 05:13)

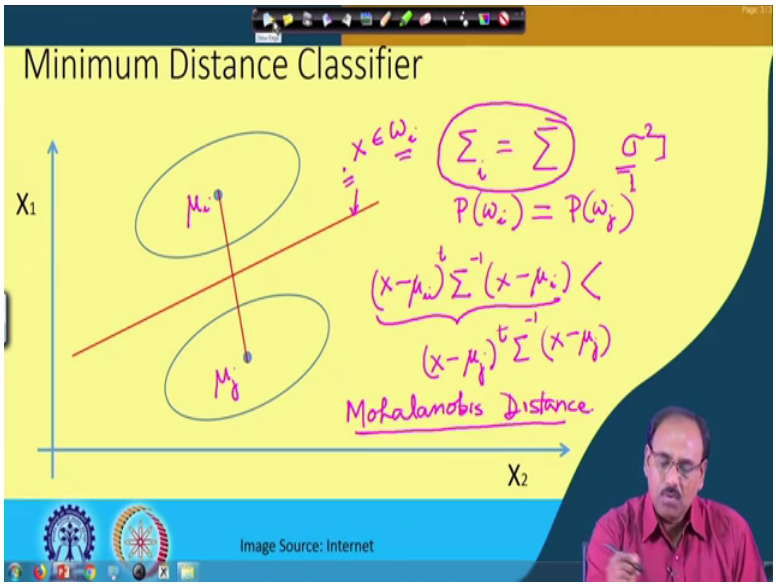


This is the mean of the vectors belonging to class omega i and this is mu j which is mean of the vectors belonging to class omega j. And, here we are assuming that the a priori probabilities P omega i and P omega j they are equal; that means, the classes are equally probable. So, under this situation when I have this unknown feature vector X over here which have to be classified, what I am effectively doing is because this X falls on this side of the boundary. So, it is to be classified to class omega i or in other words what I

am doing is I am trying to compute the distance between mu i and X and I am also computing the distance between mu j and X.

And here the distance between mu i and X which if I represent as d of X mu i and in other case the distance between X and mu j is d of X mu j; you find that d of X mu i is less than d of X mu j. And that is true for any point X lying on this side of the boundary and all this cases this vector will be classified to omega i. So, in other words the classification rule that I am applying is a minimum distance classification rule where, the distance that you are computing is the Euclidian distance between the unknown vector X and the means of the classes.

(Refer Slide Time: 07:10)



Let us see the other case where we have assumed that mu i is of the form mu; that means, the covariance matrix of all the classes are same. But, the components of the feature vector may not be statically independent; that means, the off diagonal elements of the covariance matrix may be non-zero. So, under that situation again under the assumption of equal a priori probability that is P of omega i is equal to omega j. We found that decision boundary is bisector of the line joining mu i and mu j, but the decision boundary is no longer orthogonal to the line joining mu and mu j.

So it is a bisector, but it may not be orthogonal. So, what do we do in this case? Again here if I have a unknown point X on this side, this unknown point X will be classified to class omega i, because it is falling on the side of mu i. Is it a minimum distance

classifier? Yes, again in this case it is a minimum distance classifier because, what we are computing is X minus mu i transpose sigma inverse X minus mu i. And I am also computing X minus mu j transpose sigma inverse X minus mu j. So, if X minus mu i transpose X sigma inverse X minus mu i this is less than X minus mu j transpose sigma inverse X minus mu j then the point X will be classified to class omega i.

And you find that this is also a distance measure, but it is not Euclidian distance any more, but this distance is what is known as Mohalanobis distance ok. And you find that when this covariance matrix sigma is of the form sigma square I and assuming that sigma square is equal to 1 this Mohalanobis distance will be same as Euclidian distance. So, if an in this case where the covariance matrix sigma i of all the classes is same, but the feature components may not be statically independent I still get a minimum distance classifier. But the distance that you compute in this case is not Euclidian distance, but the distance that you have to compute is Mohalanobis distance ok.

(Refer Slide Time: 10:30)

So, with this background now let me talk about what is meant by nearest neighbour rule or nearest neighbour classification. So, as we said before that every object or every signal is represented by a vector of which are vectors whichever with the vectors are computed. It may be computed using some signal processing techniques or given an image I can simply represent this image as a vector by simply concatenating the columns of the image. So, if I have an image of size say m by n, I will represent this by a vector

having m into n number of components. So, that becomes at an m into n dimensional vector.

So, once I represent these as vectors; that means, the signal an image or whatever is represented by a point in that vector space or feature space. So, as it is an m into n dimensional vector so, I am defining an m into n dimensional feature space and every image will be represented by a point in that m into n dimensional feature space. But, here I am taking simplistic view because, I cannot represent an m into n dimensional space on a 2 dimensional plane. So, what I am doing is I am projecting them into 2 dimensional space. So, each of these images that you find in this particular plane they are nothing, but different vectors. So, this image is a vector, this image is a vector; I am simply projecting them into 2 dimensional space X 1 X 2 assuming that X 1 X 2 are the feature vectors..

Now, given this let us see what is a nearest neighbour rule. So, here all this images are the known images I know what is the class from which this image comes. Say for example, I know I have lot of images which are birds, I have a lot of images which are cars, I have lot of images which are dogs ok. Now, given an unknown image this I have to classify or I have to identify what this image is. You know that all those previous images that we had those are the training images using which I have to classify this unknown image. So, one of the approach in which this image can be classified is what I do is I simply take the distance of this unknown image or the vector representing this unknown image from all other images which are known.

And once I compute this vector so, if have say P number of previously known images and I have this unknown image which I want to identify. So, I have to compute P number of distances, distance from every other image which we have in my knowledge space right. And once I do that after that I find out that what which is the image which is nearest to it. So, if you go back to the previous one you find that probably this is the one which is nearest to this unknown image. So, my classification rule is that whichever image is nearest to it, I identify this unknown image to be the image corresponding to that class. So, over here as this was the nearest image so, I identify this unknown image to be one of these images.

And obviously, in this case you find that your classification is not correct because the unknown image that I had was the image of a car whereas, my nearest neighbour rule has

said that it is the image of a dog which is obviously, incorrect. So, what is the problem in this case? The problem is I am trying to find out which is the nearest known image in my knowledge space, that is nearest to this unknown image. So, I am taking my decision based on that nearest image so, my decision is based on only one vector.

And, if that vector is an out layer then obviously, my decision is going to be wrong and that is what has happened in this case. So, an alternative is that instead of considering only one image, if I consider multiple number of images. So, what I will do is I will take multiple number of images. So, if I take k number of nearest images then it becomes a k nearest neighbour rule.

(Refer Slide Time: 15:27)

So, in K nearest neighbour rule what we do is I have as before all these different vectors or images in my feature space. Again, I have this unknown image, I compute the distances from all the know images. And, out of that I consider only few or K number of images which are nearest. And, then among this K number of images, I compute a vote that is whichever class of images is in majority in that K number of images, I consider this unknown image to be classified to that corresponding class. So, in this particular case you find that the out of all these images which are nearest.

So, all these images which where nearest to this unknown image I have two images which belong to bird, I have two images which belong to dog, but I have 1 2 3 and 4, four images of cars. So, this car images are in majority so, I classify this unknown image

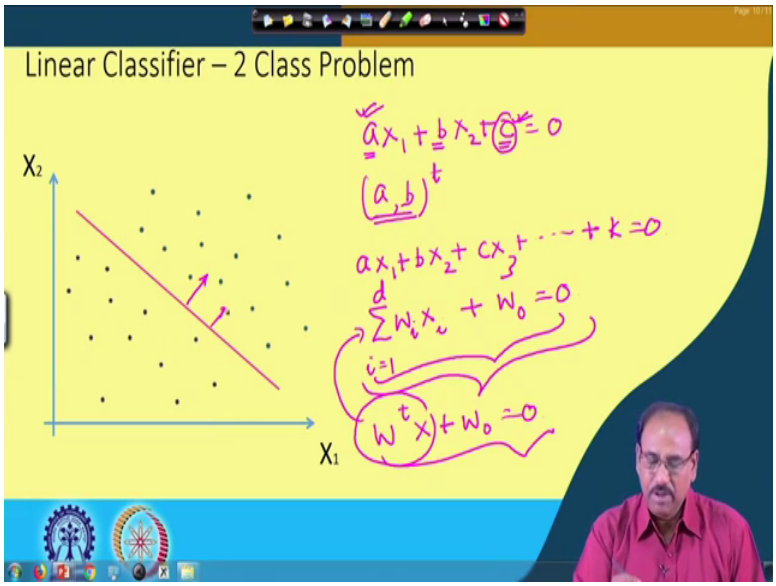
to be one of the car images and my classification in this case is correct. So, K nearest neighbour rule in general gives better result than nearest neighbour rule because, you are taking decision based on multiple number of images or multiple number of feature vectors which takes care of the out layers removal of out layers.

So, this nearest neighbour rules are very simple to apply, but what is the drawback? The drawback is usually in machine learning applications I have lakhs and lakhs of images given for training. So, whether I want to use nearest neighbour or K nearest neighbor, I have to save all those images in the memory. And while taking decision for an unknown image or for an unknown vector I have to compute all those lot of distance values. And, based on the distance values I have to take a decision that to which class this unknown image should be classified and that is obviously, computation extensive.

So, the simplest approach is that instead of trying to do this, you try to find out the decision boundaries which we have also seen earlier. So, given a number of classes the given images belonging into two different classes, if I can compute a decision boundary between the 2 classes. Then for an unknown image, if that image falls on one side of the boundary it would be classified to say class omega i, if it falls on other side of the boundary then it will be classified to class omega j.

And, for that once that boundary is computed I can discard all those training measures of the training vectors, what I need to stored is simply few number of parameters which identifies or which describes that boundary. So, let us see how this can be done.

(Refer Slide Time: 18:38)



So, let us assume in this case take a let us consider this case that all these feature vectors they belong to one class and these are the feature vectors which belong to another class. So, had we used a nearest neighbour rule and give an unknown feature vector over here in order to classify this unknown feature vector, I had to compute the distances from all these feature vectors which are known. And then based on these distance values I had to classify this unknown feature vector to either this class say omega 1 or this class say omega 2.

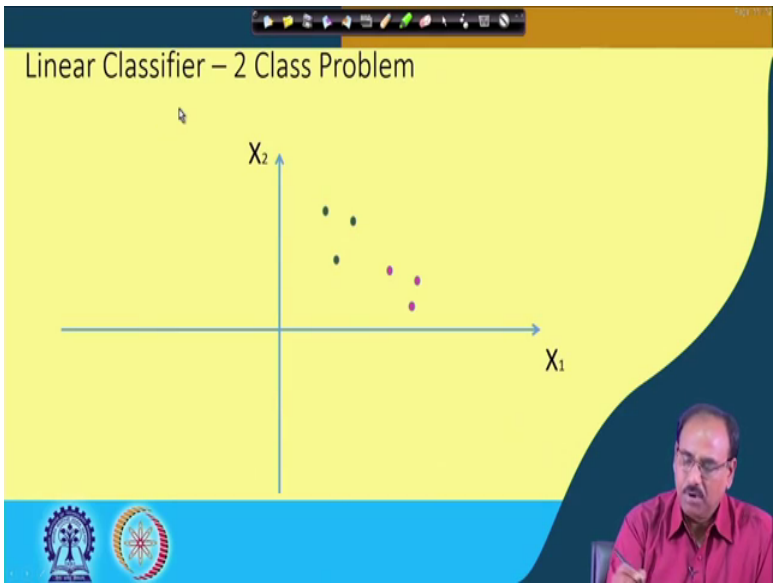
But, now what I am doing is instead of computing those distances or saving all this feature vectors in computer memory I am trying to find out a boundary between these 2 classes. So, assuming that all these features vectors are linearly separable I can separate these 2 classes of feature vectors by a linear boundary. So, in a 2 dimensional case it will be a straight line, in a 3 dimensional case it will be a plane, in multi dimensional case it will be hyperplane. So, given such a linear boundary in this 2 dimensional case you find that the equation of this boundary will be something like a X 1 plus b X 2 plus c is equal to 0.

So, the parameters representing this particular straight line are the parameters a b and c, where we know that a vector a b is orthogonal to this separating line to the line and c represents the position of the line in that is 2 dimensional space. So, if I vary a and b in that case the orientation of the line will be different and if I vary c in that case the

position of the line will be different. So, this is what we have in case of 2 dimensional space, in case of a multi dimensional feature vector the equation will be a X 1 plus b X 2 plus c X 3 plus something like k that will be equal to 0. So, if I have d dimensional feature vectors I can write this in the form W i X i or i varies from 1 to d plus W 0 that equal to 0.

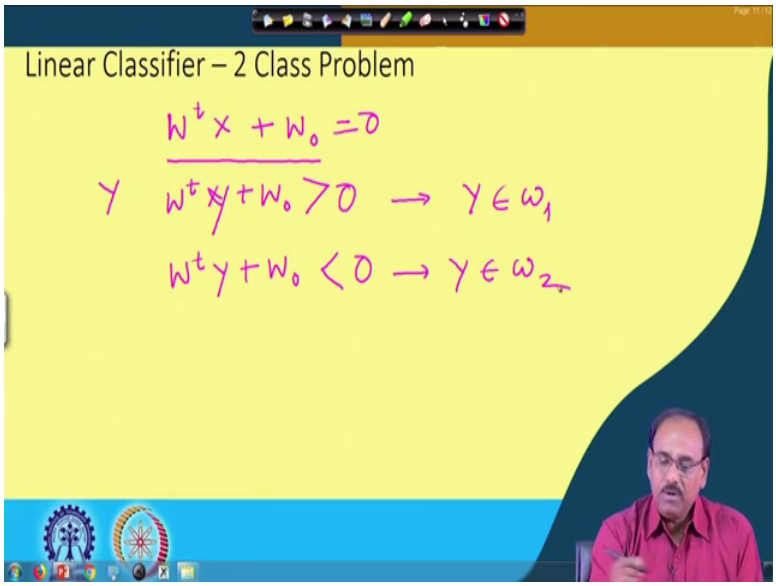
So, you remember that this equation is similar to what we have derived earlier, the decision boundary between two different classes using them normal multivariate distribution which was W transpose X plus W naught that equal to 0. So, this is same as this when this W transpose X is expanded component wise so, this is the expression that I get. So, this is the equation of the this straight line which or the plane which separates the feature vectors belonging 2 classes omega 1 and omega 2 or omega i and omega j. So, given this now I can move further.

(Refer Slide Time: 22:30)



So, what I need to do is given a set of feature vector belonging to two different classes and assuming that the feature vectors are linearly separable, I have to find out a separating plane or hyper plane which separates the feature vectors belonging to these two different classes.

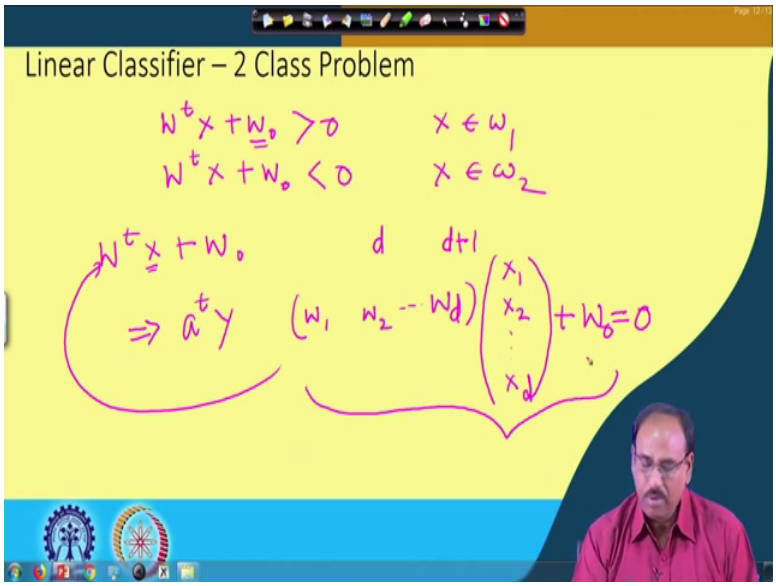
(Refer Slide Time: 22:57)



And the equation of that separating hyper plane is simply of the form W transpose X plus W naught is equal to 0. And you remember that such a plane divides the feature space into two half spaces; one of the half space is positive and the other half space is negative. So, if we say that the feature of vectors taken from class omega 1, they belong to the positive half space and the features vectors taken from class omega 2 they fall in the negative half space..

And using this once I design such a classifier of the such a separating plane then for an unknown feature vector say Y; if W transpose Y plus W naught becomes greater than 0, then we take a decision that W should belong to class omega 1. Because, Y is sorry this is Y not X that because, Y is falling on the positive half space whereas, if we find the W transpose Y plus W naught becomes negative if it is less than 0, then our decision will be that Y should belong to class omega 2.

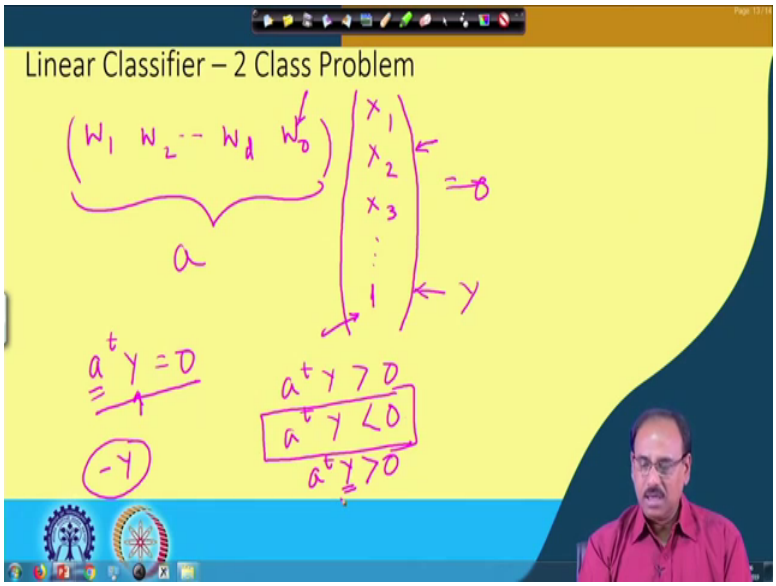
(Refer Slide Time: 24:22)



So, what we have is for all the known samples or the training samples I if the training samples is taken from class omega 1, I should have W transpose X plus W naught greater than 0. When X belongs to class omega 1 and W transpose X plus W naught should be less than 0 when X belongs to class omega 2. So, whichever W when vectors W and the bar stem W naught I choose that must satisfy these relations, these inequalities for the samples taken from class omega 1 and the samples taken from class omega 2 right.

Now, we find that this particular expression W transpose X plus W naught, I can put this in an unified representation in the form that I can write this as a transpose Y. How I can do it? I have to append an additional dimension, I have to append an additional dimension into this X. So, if X is of dimension d I have append a 1 to this dimension to this X and make it of dimensional d plus 1. So, the way it is done is suppose it is a d dimensional vector so, this W transpose t W is also a d dimensional vector. So, I have W 1 W 2 up to W d this multiplied by X 1 X 2 up to X d. So, this is the expression which gives me W transpose X plus W naught, this equating to 0 gives me the equation of the separating plane.

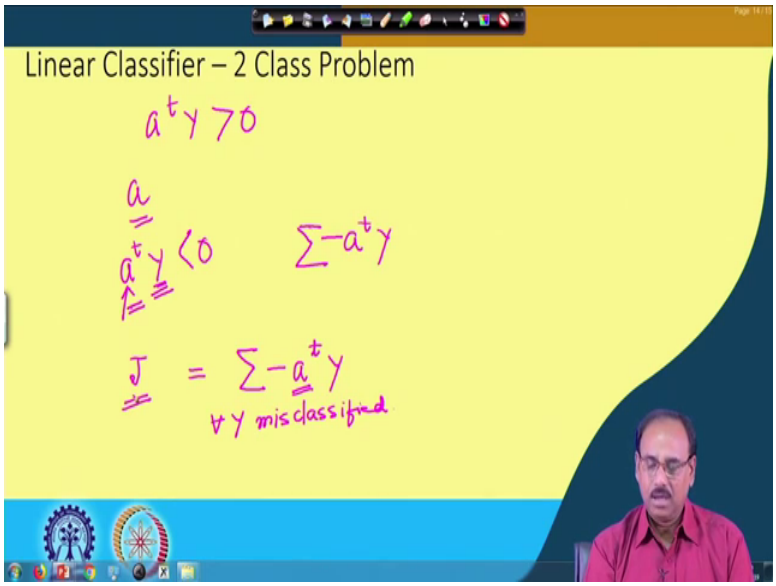
(Refer Slide Time: 26:27)



Now, this equation I can rewrite in the form W 1 W 2 W d W 0 X 1 X 2 X 3 up to 1 this equal to 0; so, I can call this as a and this as Y right. So, my equation simply becomes equation of the plane as a transpose Y equal to 0, where a is W appended by W naught and Y is vector X appended by an additional component which is equal to 1. So, I can represent that equation W transpose X plus W naught equal to 0 as W transpose Y equal to 0.

And as before if Y belongs to class omega 1 I have to have a transpose Y greater than 0, if Y belongs to class omega 2 I have to have a transpose Y less than 0. What I can do is for all the Y which are taken from class omega 2 I simply negate them. So, for them instead of considering Y, if I consider minus Y for all Y which are taken from class omega 2. And if I do this then if Y is correctly classified by a I will have simply a transpose Y greater than 0, where Y if it is taken from class omega 2 it is negated Y.

(Refer Slide Time: 28:14)



So, given this I have now uniform criteria of correct classification that a transpose Y has to be greater than 0 for all the training samples Y whether they are taken from class omega 1 or they are taken from class omega 2. Because, for all the Y's which are taken from class omega 2 they have been negated. So, my design approach can be that I can start with any a arbitrary again, but if I find so with this a I test all the training samples.

So, as long as for every training sample a transpose Y remains greater than 0 I know this a correctly classifies all those training samples. But, for if for any training sample I find that a transpose Y becomes less than 0, then I know that this Y has not been correctly classified or it has been misclassified by this vector a.

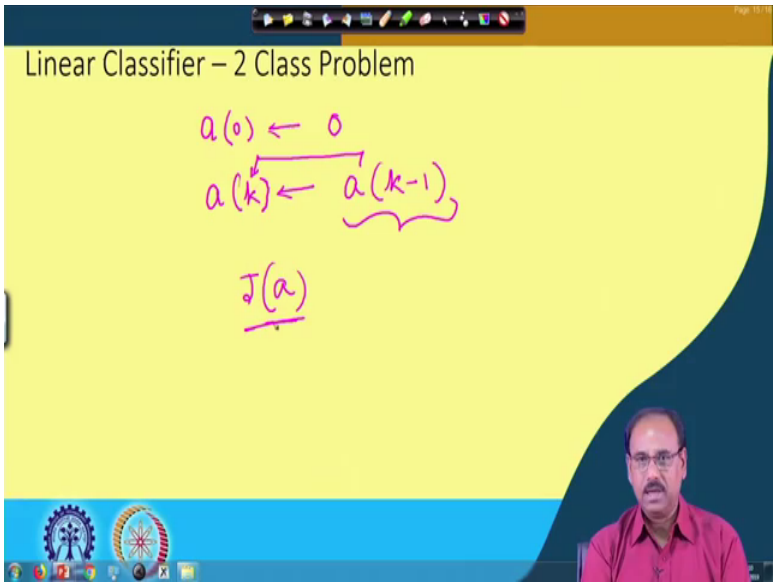
So, what I have to do is in this particular case this a has to be modified. So, in order to do this what I can do is I can compute some sort of error terms, that is for every vector which is misclassified I will compute an error. So, the error term because Y will be misclassified, if a transpose Y is less than 0 I can simply compute the error term as minus a transpose Y. So, whenever a transpose Y is less than 0 minus a transpose Y, we have a positive term.

So, if I check all the feature vectors all the training vectors which are misclassified by this a, I collect all of them compute minus a transpose Y for all those feature vectors which are misclassified and take the sum of all of them right. So, I compute an error say

J which is equal to minus a transpose Y, for all Y which are misclassified and J will be equal to 0 if all the samples are correctly classified.

So, we find that if I have the misclassified samples, this sample this sum of minus a transpose Y is going to be positive. And if all the samples are correctly classified, all the vectors are correctly classified then the value of J will be equal to 0. So, while designing this a or while designing the boundary between the linear boundary between two different classes or approach should be that we should be able to or we should try to reduce this error term J ok. And for that the kind of approach that can be taken is something like this.

(Refer Slide Time: 31:30)



So, I have this a I will put an initial a to be equal to 0 and then gradually in the k th step I should be able to a get a k from a of k minus 1 that is the previous value of a such that I move from a k plus 1 to a k minus 1 to a k in such a way that this movement will try to reduce the error J a; if I represent the error as a function of a that when I move from j a k minus 1 to a k the error a should be reduced. So, we will talk about this more in our next lecture.

Thank you.