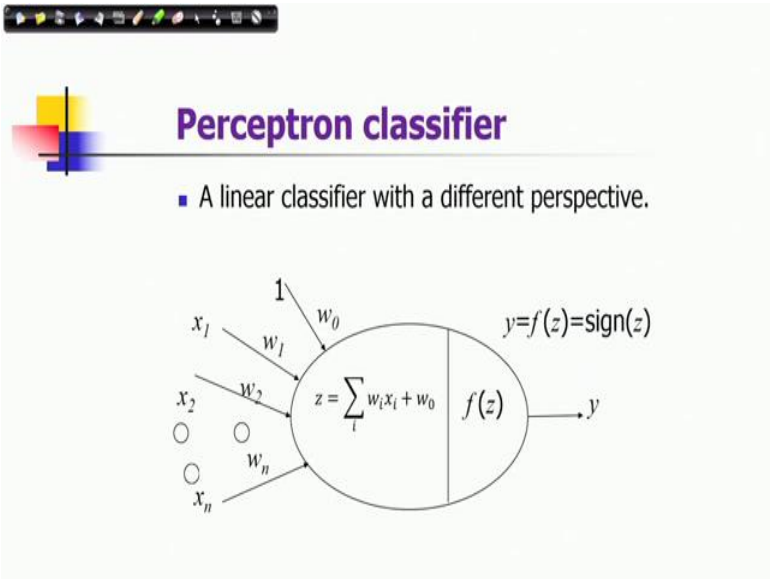
Computer Vision
Prof. Jayanta Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 49
Clustering and Classification Part - IV

(Refer Slide Time: 00:25)



We are discussing about different classification schemes. And today I will discuss about a classifier which is a linear classifier of a different perspective and that is perceptron classifier. So, in a perceptron classifier you can see here that there is a concept of a node and edge representation. So, there is a particular node which represents a node of a classification classifier here. And it has an input, in this case you can see that there are n-dimensional inputs and then in fact the input is also augmented by another dimension one.

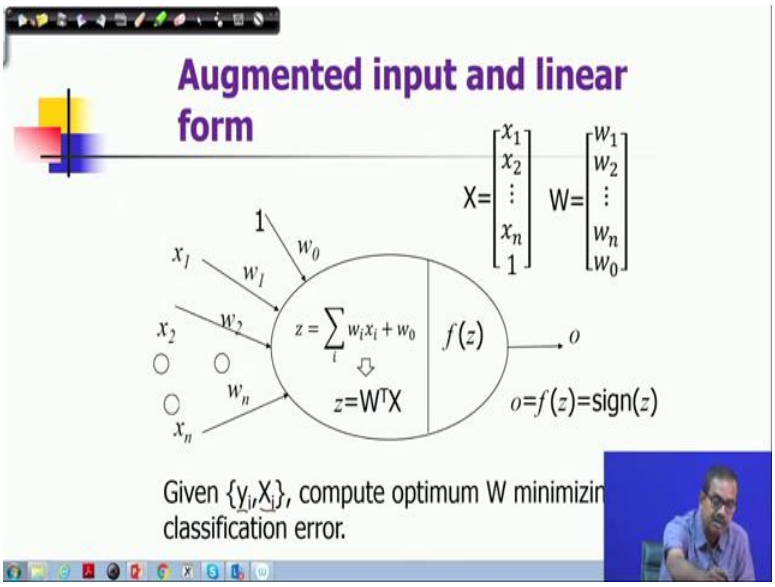
So, this is your input and these are the weights to the particular node. And what we are doing, we are doing weighted sum of this input, and then plus there is a bias. Now, you can also consider as if the input is augmented by another dimension whose value is 1, and then the whole thing can be considered also as a linear classifier.

$$z = \sum_i w_i x_i + w_0$$

We will discuss these things also in subsequent slides that the fact is that now this weighted sum of this weighted input and also add and also it is added with this bias so this value provides also these value is used in a function, and the function value finally, provides you the output of a classifier.

So, depending upon the functional value, the classification output will vary. And in a normal linear classification scheme what we discussed in previous lectures, this function was chosen as a signal functions or sin functions which means if the z is positive, you consider the class level as 1; if z is negative, you can consider class level as - 1, and 0 is little ambiguous. So, for the completeness of the function you can say z is positive for 0 as class level as 1. So, we will see that with this perspective how a perceptron classifier can be understood.

(Refer Slide Time: 02:45)



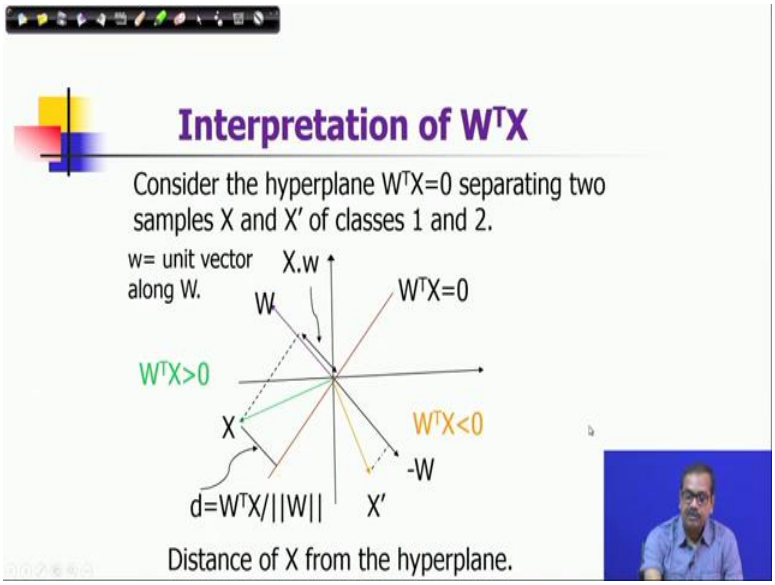
So, as I was mentioning that considering the augmented input, this could be in a linear form. And then we can consider the input as a vector of n + 1 dimensional vector, where the last dimension n + 1th dimension will be kept as always 1. And then and also the weight is also an n + 1 dimensional weights where the bias is included as the n + 1th part as w0. Here as you can see, and then it becomes simply a linear operation.

So, it is it can be expressed as W^TX, which is something like also you can consider from the vector operations as the dot product of the weight and the augmented input X. And this

has been used in a function and finally, output is a functional value of this particular net input which is z. And we are using in this case for example, a sign(z).

So, the problem of classification scheme is that given (y_i, X_i), where y_i is the level of the input and X_i is the corresponding input we have to compute an optimum W which minimizes the classification error. And we have already discussed about this particular type of problem in the previous lecture. So, when you discussed a linear discriminant analysis now with respect to this perceptron classification scheme, let us also revisit that problem.

(Refer Slide Time: 04:36)



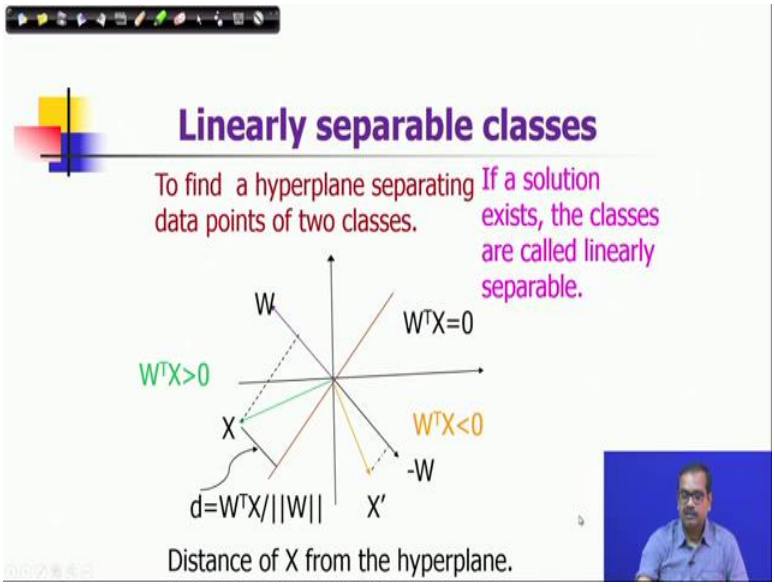
So, in this case, you consider that now this is hyper plane and this $W^TX=0$ is an hyper plane, and it separates two samples X and X' of classes 1 and 2 as you can see. So, in my diagram this is the hyper plane in that is representing as a hyper plane. And you consider the normal to this hyper plane is W. And you note that in this case because of this form this hyper plane has to pass through the origin of the corresponding space, in this case for the simplicity I am showing it in two dimension, but you consider it is a n-dimensional, any arbitrary dimension we have to consider in this case.

Now, consider there is a sample X which lies in one side of the hyper plane, and another sample X' which lies in the other side of the hyper plane. So, hyper plane divides this space into two partitions. So, in this side and if I define W this is a direction of W that vector, so if I take the dot product of these vectors, so this is X.

And so if I take the dot product finally, it is the projection of this vector on this direction that would give you and what it is showing it is basically if you consider that is a unit vector along W, and this is what is X.W, and as you can understand that this should be positive it will form a positive. If it is in the same partition where the vector normal and also the x there lying, then this dot product will be greater than 0. And the interpretation would be that you are measuring the distance of X from this particular hyper plane.

So, the distance from our analytical coordinate geometry, distance function is given in this form, you can see that this is a very standard form how the distance could be completed. Now, if I consider the other sample X' which lies in the other side of the hyper plane, and it is X, it is in our case as I mentioned that X' is of class 2, that means, which is also in a different class then this dot product will be negative because now you are taking dot product with W in that direction, so $W^TX<0$.

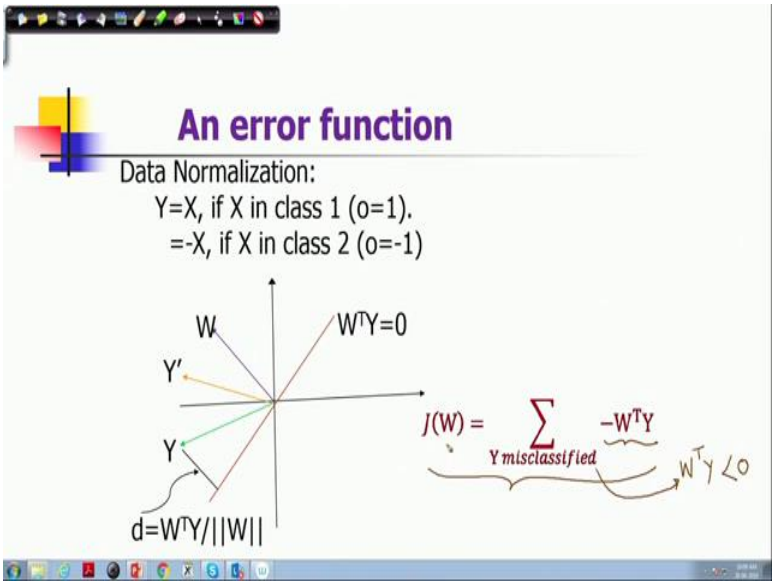
(Refer Slide Time: 07:25)



And so this is an ideal case. So, this hyper plane can discriminate these two samples of two different classes. So, the condition should be therefore a classifier that the samples of class 1 should lie in one side of the hyper plane, whereas the samples of class 2 should lie in the other side of the hyper plane. So, if this condition is satisfied, if your input is given in such a way and if there exists such a hyper plane which can divide the input which can partitions the input in this form, then we say these classes are linearly separable. So, and you can get a solution of hyper plane. So, in that case, our objective would be to find a

hyper plane separating data points of two classes. So, this is what if a solution exists then the classes are called linearly separable.

(Refer Slide Time: 08:24)



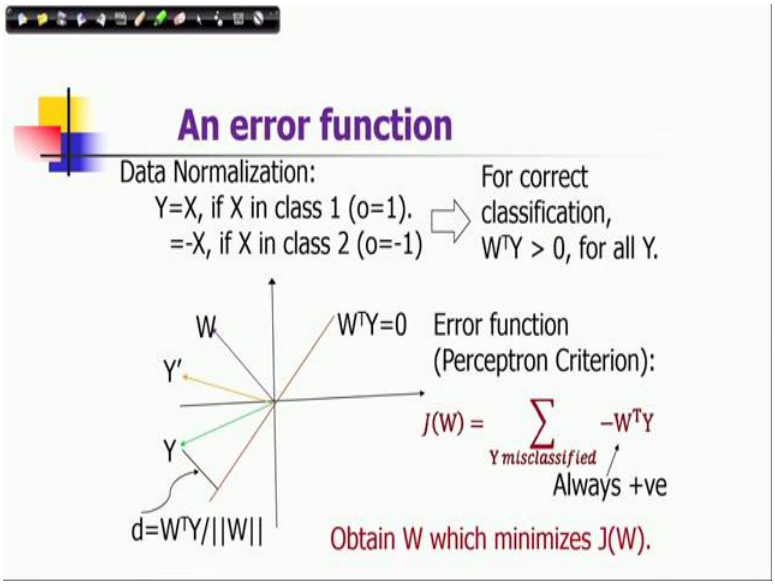
So, we are trying to design an error function. And in this case, you consider the data normalization we will be doing some interesting manipulation of data, so that your problem gets much more simplified. Let us consider the samples or the input samples which belongs to class 1 is X, whereas if it is in class 2 will be inverting this sample will be negating this sample -X. So, in that way intentionally what we are doing even the samples of class 2 also we are bringing in the same hyper plane in the same partition of the hyper plane.

And then with this kind of normalization, so all are input samples which should satisfy the property that all of them should lie in one part of the hyper plane, and we have brought it towards the positive part of the hyper plane, that means, the partition where also the normal also the normal vector also lies in that partition. So, with this kind of operations, they we have satisfied this conditions that all my input sample should be lying in the positive form.

So, with that data normalization we can define an error function where we can see that this is a form of this error function. And if only we consider if Y is misclassified then only we compute this value which means this value has to be always positive because if Y is misclassified with according to your input condition, then for a misclassified sample it would be always negative for a misclassified sample. So, if I take the negation of that, it

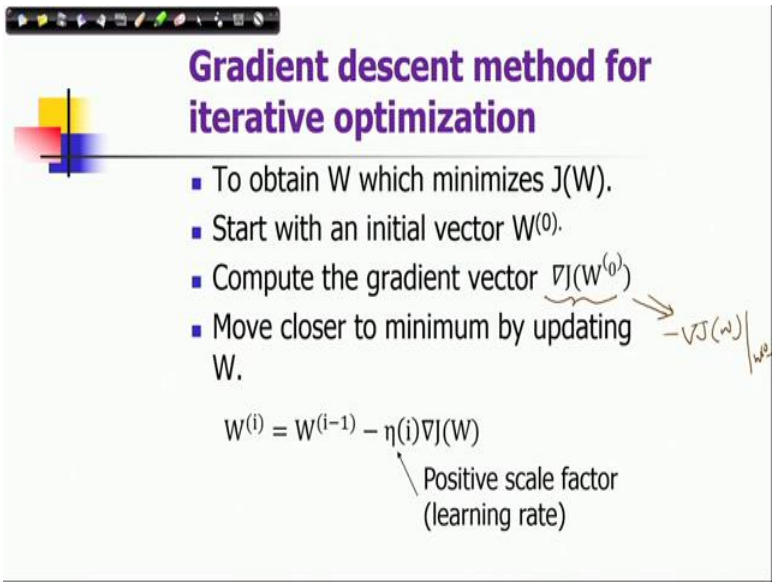
would be positive. So, this error will be always positive. And using this error function, we will be will be designed we will be designing a classifier.

(Refer Slide Time: 10:41)



Let us see how we can do it. So, this is what for correct classification it has to be always positive as we mentioned. And this is an error function which you have defined, and this should be always positive as I mentioned. So, our problem is that to obtain the weight vector which minimizes this particular error function.

(Refer Slide Time: 11:05)

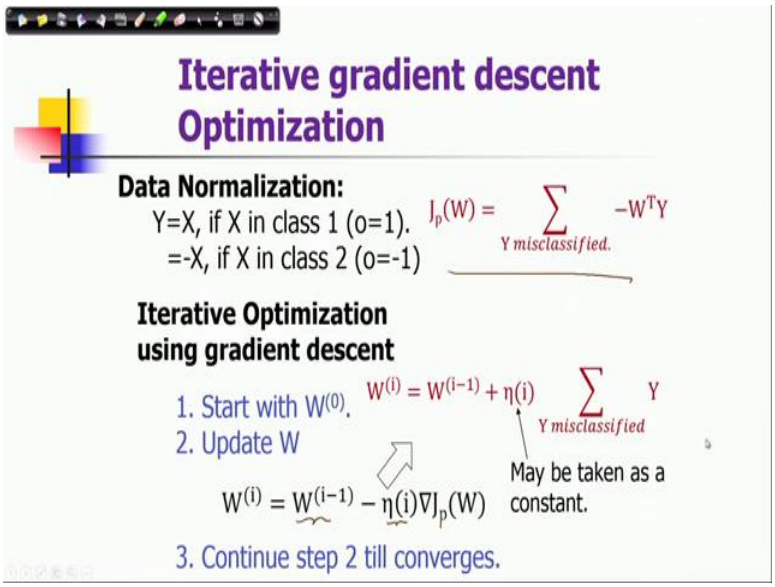


So, we will be using here a gradient descent method for iterative optimization, and the steps should be like this way, we can start with an initial vector which has been shown here, I say W^0 that is the initial form. And then we compute the gradient vector. Suppose, this gradient vector is denoted with this operation we will see how we can compute this gradient vector. So, we are performing gradient over this function. And at the initial, so you note that the functional point is W_0 here. Now, you should move close to the minimum by updating W.

So, you did is following the rules of gradient descent method, you have to move along this steepest gradient directions in the opposite to the steepest gradient directions that means, you should move along $-\nabla J(W^{(0)})$ what has been shown as a symbol say this is a symbol. So, you should move along that $-\nabla J(W)$, which has to be computed at W^0 that is the meaning of this particular symbol.

So, you should move along that, but instead of if what we can do while moving these things will be simply considering there is certain step size by which will be moving it is a proportional factor to that particular value that will be moving. And, this is particular this is called as the positive scale factor or learning it in our weight updation.

(Refer Slide Time: 12:45)



So, we let me further elaborate this particular process of iterative optimization using gradient descent we start with $W^{(0)}$ then update W and W should be updated in this form.

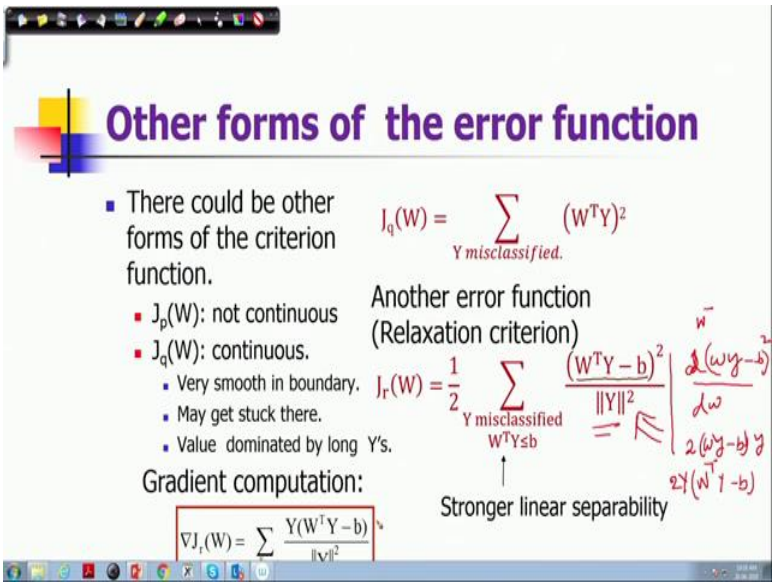
$$W^{(i)} = W^{(i-1)} - \eta(i)\nabla J_p(W)$$

As you can see it is a previous result of the previous iteration of W, and then it has to be moved along the steepest gradient directions multiplied by this scale factor. This is the gradient of this particular function. And we will see how to compute this gradient analytically we can very easily derive these computations, because if I take W^T from here if I take the gradient it would be only sum of Y or sum of - Y.

So, this expression can be modified in this form as you can see that $W^{(i-1)} - \eta(i) \sum_{y\ misclassified} Y$, Y is once again misclassified sample. So, computation is very simple in this regard. What you should do that you start with any W, then you use the classifier to see which samples are getting misclassified, then some of those samples will give you the corresponding update of the weight you should move the weight by adding the sum of those samples with a proportional value.

Now, this value may vary over the steps or you can also use some heuristics to keep it as constant or small. So, you should continue this operation still it converges, so that is what it could be maybe it could be taken as a constant this particular learning rate can be taken as a.

(Refer Slide Time: 14:30)



There could be other forms of criteria function. For example, this is another function where instead of providing it as $-W^TY$, we can consider $(W^TY)^2$. So, the advantage is that the

previous error function what I mentioned that means, it is sum of minus W^TY. Note that all Ys are misclassified only you are taking your defining functions over those values only.

So, if Y is not misclassified that should not come into this functional computations. But this function particularly it is continuous though it has some issues like it is very smooth in the boundary which means there now if you apply gradient descent, you can get arbitrary result, you can get stuck at some location, and also this value is dominated by long Ys. So, some kind of normalization with respect to Y could have been there. So, we can choose another kind of error functions which certain relaxation criteria where this normalization has been considered, and you can see that this is a function where you are using $(W^TY - b)^2$

So, in this case the hyper plane is further moved towards positive side by a distance b. So, it is more stringent criteria for satisfying the linear separability of the classes. And this half is used just to simplify the computations because due to the square term, but the derivative these two will come here. So, 2x(1/2) will get canceled as 1. So, we will also see its analytical form. So, it has a stronger linear separability as I mentioned.

And gradient will be computed in this way. As you can see two half and this squared square term, it gets canceled, and then you can apply simple differential geometries concept of differential geometry here also what you did for the one-dimensional cases one-dimensional functions this can be extended to in many cases particularly when you are when you are using the matrix operations in the linear form, then you will see almost similar rules that applied here. So, this is a function of W as you can see.

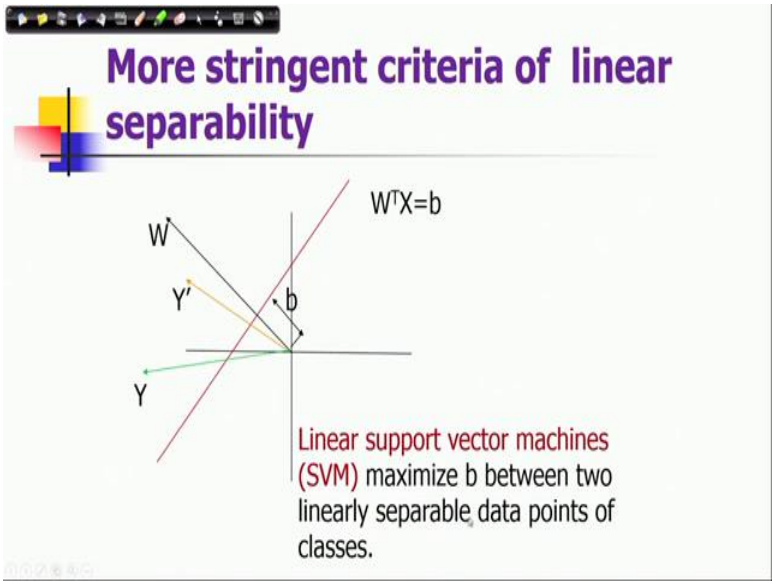
$$\nabla J_r(W) = \sum \frac{Y(W^TY - b)}{||Y||^2}$$

So, if you are deriving over this particular value, then the term Y will come here. So, if I derive this particular value will we can see that like what we did in the in any differential geometry suppose it is a function it is a function of say W.

So, if I perform derivative with respect to W, this value should have been two (WY-b)Y. When you translate it into the matrix form, then you get these operations, then you get this value as you can see why (W^TY-b) that is 2 (YW^TY-b). Of course this is a constant, so this

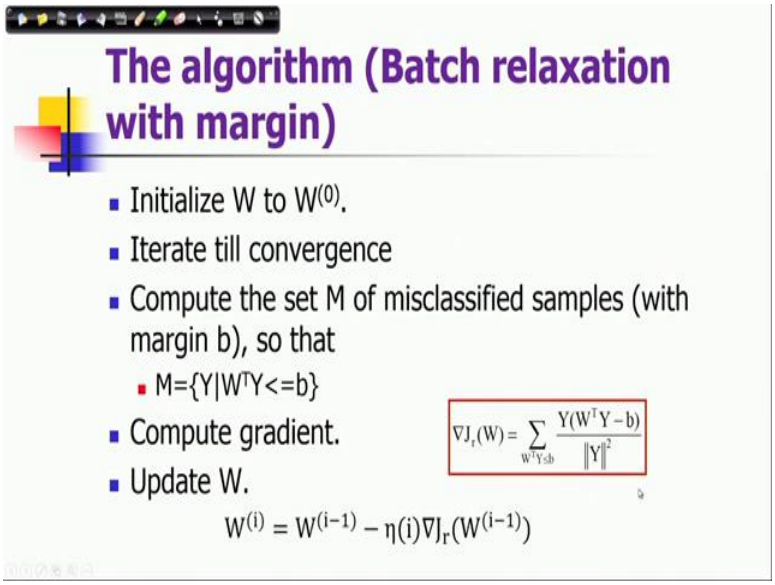
constant term will come here, so that is what you are finding here, so that you can see here that the gradient would be computed in this form.

(Refer Slide Time: 18:52)



So, this particular diagram shows that because of the use of the offset b, you are moving further this separating hyper plane further towards positive partition by distance b and that puts a stringent criteria of a of this linear separability. Just to note here that even support vector machines linear support vector machines that is also that is also a kind of linear classifier and which maximizes this b between two linearly separable data points of classes, so that is the property of a support vector machine.

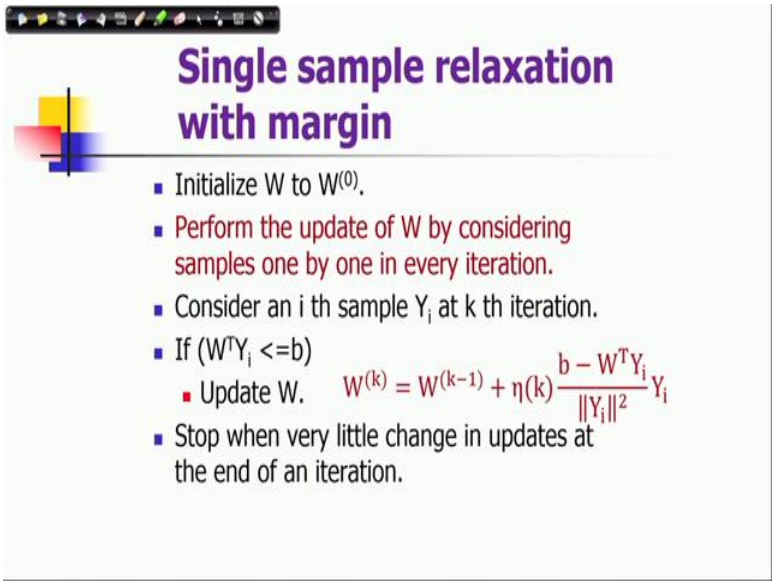
(Refer Slide Time: 19:38)



So, let me summarize this algorithm, and which we denote as batch relaxation with margin. So, in this case we initialize W to W^0, then iterate till convergence. And for that we have to compute a set of misclassified samples where the conditions of classification you know gets violated because of the data normalization all W^TY operation should be positive or rather in this case for stringent classifier, they should be greater than equals b or greater than b.

So, the misclassification is that if it is <=, we consider it as a misclassification. And we are considering those sample Y, and then use those sample you compute the gradient as shown in this particular computational expression, and then you can update W by following this particular rule. So, this is what you should go on doing till you get convergence of your weights.

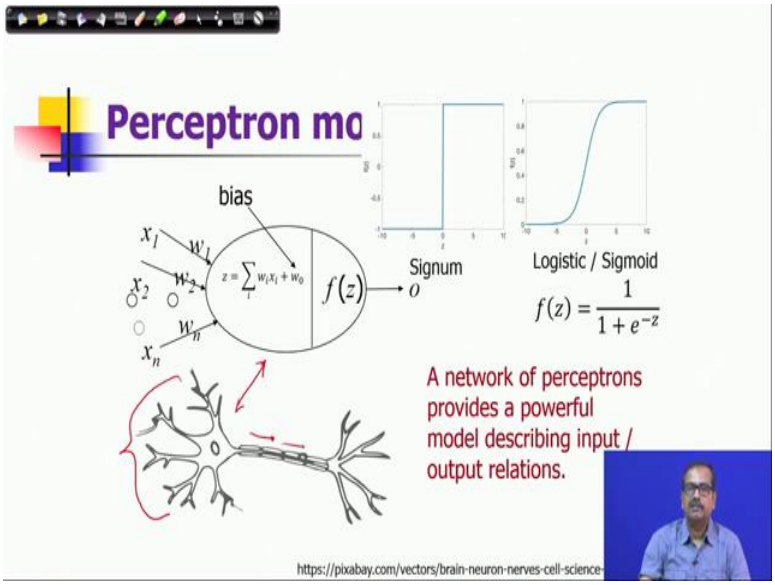
(Refer Slide Time: 20:48)



Now, batch relaxation means now you are performing the classification of every samples in your batch, and then you are computing the gradient function. Now, there is simple simplification of these particular computations, and it has been found you can this is all this also gives the same performance, but it is much simpler and faster instead of waiting instead of considering the whole batch at a time, we can consider immediate updating of W by considering a single sample whenever it is misclassified. So, whenever a sample is misclassified immediately you update the weight W by following the same rule, same classification same gradient computation.

So, this is what you should perform the update of W by considering samples one by one in every iteration, which means if it is misclassified, then you update W. And you go one by one you take all the samples and observe when W is converged. So, this is the expression here now for a single sample computations of updation of W. And you should stop when very little convergence very little change in updates at the end of an iteration.

(Refer Slide Time: 22:12)



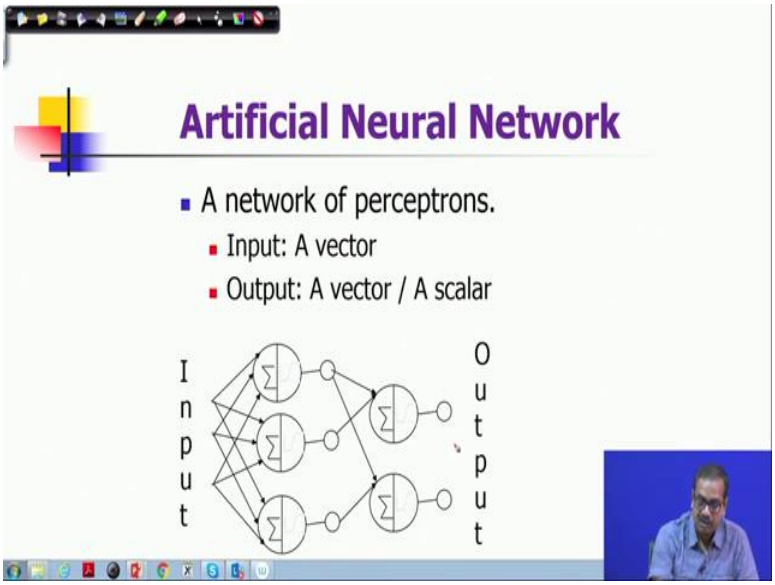
So, actually the perceptron classifier, it models also a neuron what is you know in a in its biological interpretations when you as a biological functions. As you can see in this particular classifier the inputs are given as with the classified the inputs are weighted inputs are taken, and then the output is dependent on as a function of that weighted input. If I consider a biological neuron, you will see that a neuron it consists of several synapses. So, in its input synapses in its input and this could be considered as those exciting synapses, synapses which is coming from which is taken from this input.

And then it processes it this weighted input is processed and then the output is propagated through the corresponding you know path of our nervous system of the neuron. And it goes to the other neurons or the excitation is propagated through this path, and other neurons are connected at the same. So, you can consider the perceptron node is like a single neuron, and it is collecting excitations it getting excitations from different synapses in the input and propagating and generating and output response which propagates through this path and which can be connected further.

So, this particular model is a very strong model. You can consider it as a network of these neurons or perceptrons and you can consider a system that your input and output conditions are or your output is generated through this excitation of these networks there is an input and through this network of neurons finally, the output response is generated and that is this model is known as neural network or artificial neural network model. And various

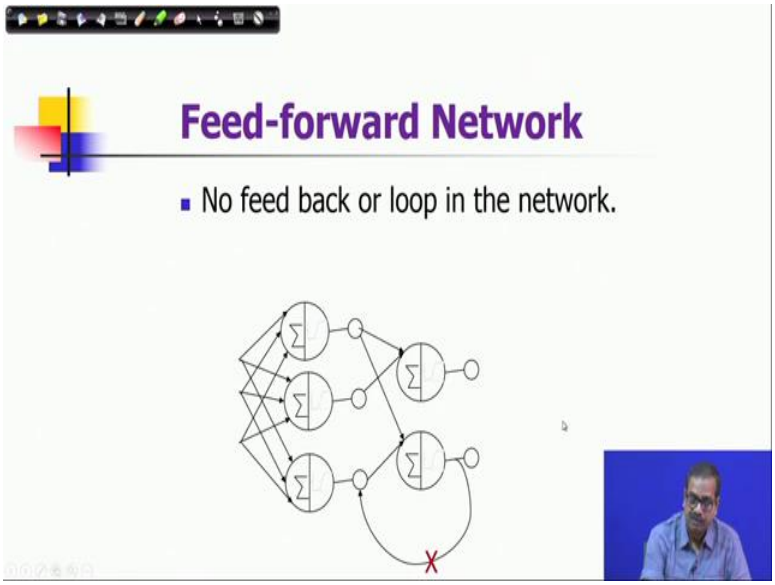
kinds of input-output relationships can be modeled using this generalized model, it is a very powerful model in that respect.

(Refer Slide Time: 24:38)



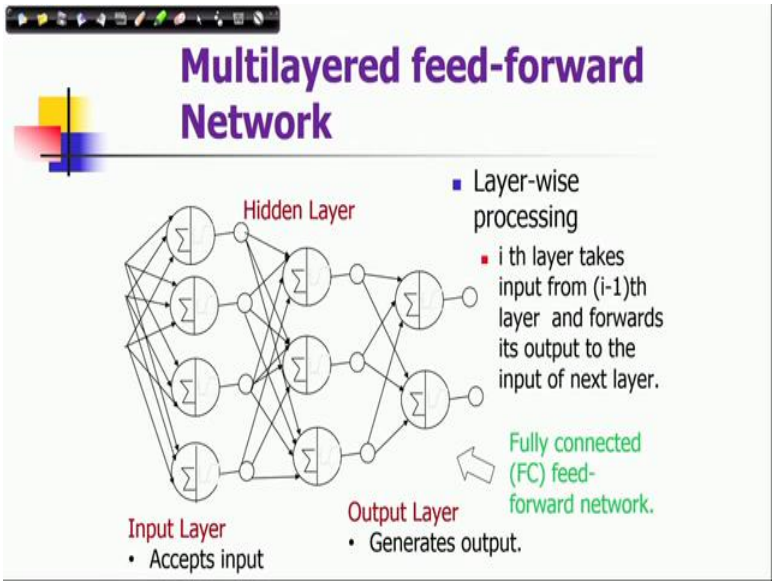
So, this is what we see here in an artificial neural network, it is a network of perceptrons where input is a vector we considered input is a vector and output is also a vector it could be also scalar, that means, you can have several output neurons which is providing you the output or you can consider only single output neuron, then the output becomes a scalar output.

(Refer Slide Time: 25:09)



One of the special case of this kind of network is that they are mean there is no feedback or loop in this network. So, then we call this network is feed-forward network. So, a free loop means from an output you get a feedback to you know one of the one of its input layer sorry input neuron. So, this is kind of loop is not here in this kind of when it is not there in the network then we call this network is feed for feed-forward neural network.

(Refer Slide Time: 25:42)



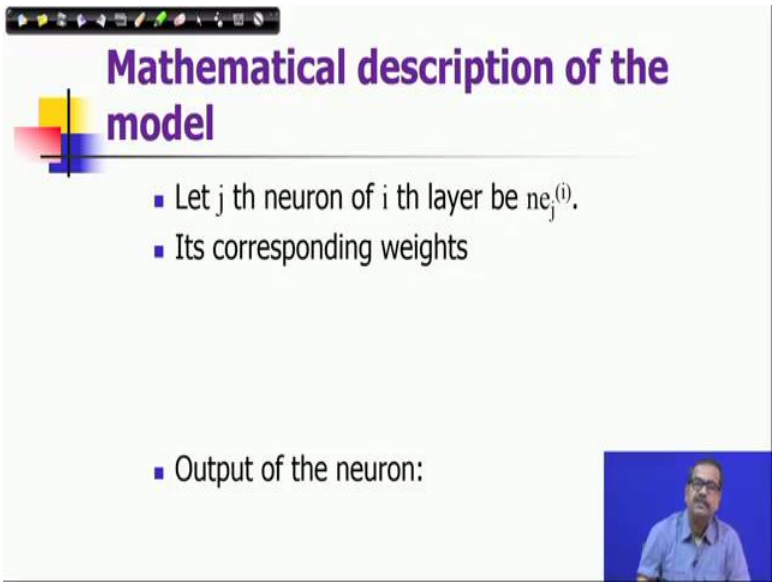
And a very general form of feed-forward neural network is a multilayered feed-forward neural network as has been shown here that there is an input layer which accepts input. So, an input layer which accepts input as we have seen here then there are several intermediate layers those layers are called hidden layer. In this case, we have shown only one particular hidden layer and then there is an output layer where you are collecting the outputs or whether outputs are generated which generates the output.

Now, you can see that in this particular form every layer has certain number of neurons which may vary. And but from a particular layer say from the one input layer to another to the next higher layer now for every neuron the connectivity to all the neurons of the next higher layer is ensured so in this particular model. So, there is a connectivity which means the weighted the there are weights from the responses of a particular neuron, so this weighted response is the input of the input of the neuron of the next layer. So, or every neuron in the next layer gets weighted sum of the outputs of the previous layer and that provides a net input to this layer.

And finally, again as you understand there is also a non-linear function which processes that net input as a weighted sum of the input to that particular neuron and then produces its corresponding output or response which again propagates or which excites the neurons of the other layers. So, this is the kind of model and since all the neurons are fully connected by its previous neurons of its previous layers; so, we call this kind of network also as a fully connected network.

So, as you can see that it is a layer wise processing, and the ith layer takes input from i -1 ith layer and forwards its output to the input of next layer, and this network is called fully connected feed-forward network.

(Refer Slide Time: 28:18)



So, let me take a break here. We will be discussing particularly the mathematical description of this particular model and its analysis so that we can model the we can solve the problems of designing classifiers using neural network or problems of classification is a neural network that we will do in the next lecture.

Thank you very much for listening to this particular lesson.

Keywords: Perceptron classifier, linearly separable, gradient descent, batch relaxation, feed-forward