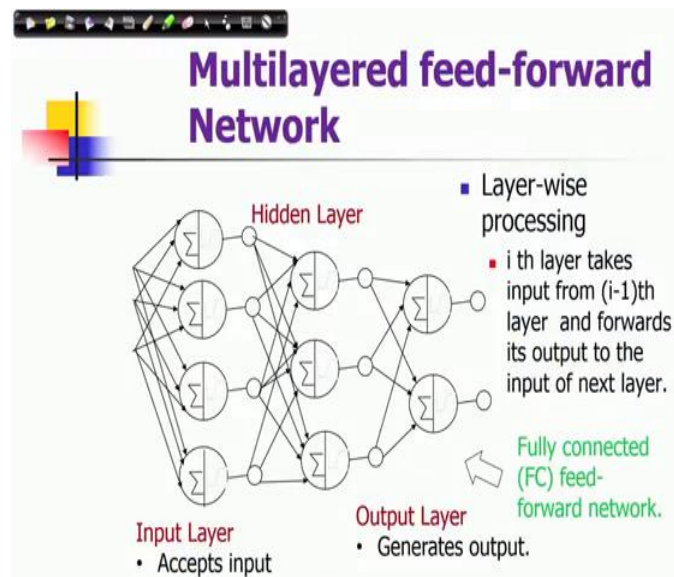


Computer Vision
Prof. Jayanta Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

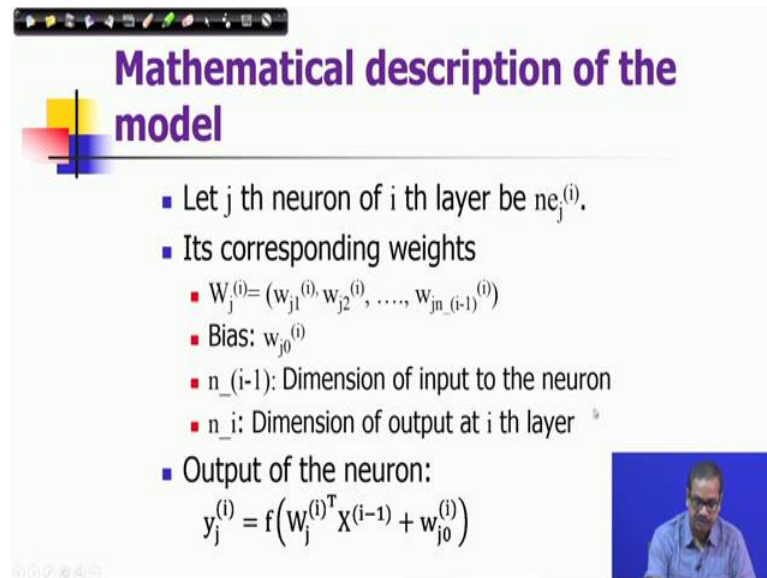
Lecture - 50
Clustering and Classification Part – V

(Refer Slide Time: 00:22)



We are discussing about artificial neural network and particularly it is a multilayer feed forward neural network, that we will be considering and this particular network will be used to model a classifier. So, let us see how we can model a multilayer feed forward neural network. We will be concentrating on a latest model particular neuron of a layer or a particular layer which means let us consider a j th neuron of i th layer.

(Refer Slide Time: 00:47)



Mathematical description of the model

- Let j th neuron of i th layer be $ne_j^{(i)}$.
- Its corresponding weights
 - $W_j^{(i)} = (w_{j1}^{(i)}, w_{j2}^{(i)}, \dots, w_{jn_{(i-1)}}^{(i)})$
 - Bias: $w_{j0}^{(i)}$
 - $n_{(i-1)}$: Dimension of input to the neuron
 - n_i : Dimension of output at i th layer
- Output of the neuron:
$$y_j^{(i)} = f\left(W_j^{(i)T} X^{(i-1)} + w_{j0}^{(i)}\right)$$

And so, to model this neuron it has its input weights. So, you consider that a weight is a vector and as it is connected to all the outputs of the all the neurons of previous layers. And, if I consider that number of output is given by n_{i-1} . So, where n_{i-1} is a number of neurons at $(i-1)^{th}$ layer so and the bias is considered it as $w_{j0}^{(i)}$. So, this is the dimension of input to this neuron which means also the number of neurons in the output layer for a fully connected network.

And, as I mentioned dimension about output at the i th layer which means that is a number of neurons at the i th layer. So, the output of the neuron can be described in the mathematical form in this way, that it is considered first to take a net input from its input. That means, $X^{(i-1)}$ is actually the vector generated by $(i-1)^{th}$ layer of dimension n_{i-1} . Now, you have the product you have the weighted combination of this input.

And, then it is added with a biasing term which has been considered here and then that becomes a net input to the function which will be a non-linear function which will be generating the output response.

(Refer Slide Time: 02:38)

Mathematical description of the model

- Output of j th neuron in i th layer:
$$y_j^{(i)} = f(W_j^{(i)T} X^{(i-1)} + w_{j0}^{(i)})$$
- Input output relation in i th layer

$$Z^{(i)} = \begin{bmatrix} W_1^{(i)T} \\ W_2^{(i)T} \\ \vdots \\ W_{n_i}^{(i)T} \end{bmatrix} X^{(i-1)} + \begin{bmatrix} w_{10}^{(i)} \\ w_{20}^{(i)} \\ \vdots \\ w_{n_i 0}^{(i)} \end{bmatrix}$$

The diagram shows the weight matrix $W^{(i)}$ pointing to the first column of the matrix in the equation above. The bias vector $b^{(i)}$ points to the second column of the matrix in the equation above. A small video inset of a man is visible in the bottom right corner of the slide.

So, if I consider the input output relation in the i th layer then that can be described in this form, you have in the i th layer you have once again n_i number of output neuron. So, for each one there is a weight vector which is connecting all the neurons of the previous layer. So, this is how this relationship can be expressed as you can see that each input with its weight vectors, it is multiplied and it is making a weighted sum with respect to the components of the previous layer and then it is generating the output of the i th layer.

So, note that the dimension here is of n_i this output is n_i , this is a dimension and these are the biases. So, we can express in brief we can note denoted that this is a weight matrix of i th layer and this is corresponding the bias vector at the i th layer. So, these are the two parameters which we are related to the i th layer.

(Refer Slide Time: 04:00)

Input output relation

- Output of j th neuron in i th layer:

$$y_j^{(i)} = f(W_j^{(i)T} X^{(i-1)} + w_{j0}^{(i)})$$
- Input output relation in i th layer

$$Y^{(i-1)} = f(W^{(i)} X^{(i-1)} + b^{(i)}) \equiv \begin{bmatrix} y_1^{(i)} \\ y_2^{(i)} \\ \vdots \\ y_{n_j}^{(i)} \end{bmatrix}$$

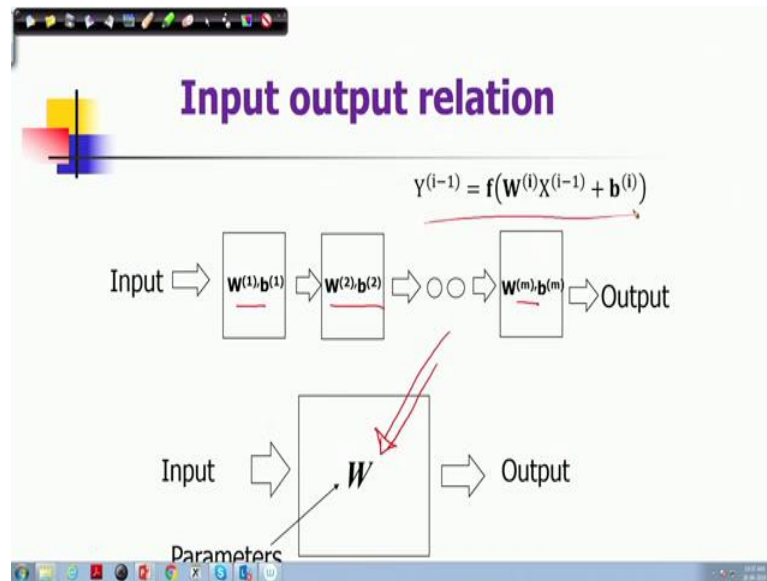
$$Z^{(i)} = \begin{bmatrix} W_1^{(i)T} \\ W_2^{(i)T} \\ \vdots \\ W_{n_j}^{(i)T} \end{bmatrix} X^{(i-1)} + \begin{bmatrix} w_{10}^{(i)} \\ w_{20}^{(i)} \\ \vdots \\ w_{n_j 0}^{(i)} \end{bmatrix}$$

Input \rightarrow $W^{(1)}, b^{(1)}$ \rightarrow $W^{(2)}, b^{(2)}$ \rightarrow \dots \rightarrow $W^{(m)}, b^{(m)}$ \rightarrow Output

So, the input output relation can be described to in this form as we can see that in a particular block diagram we have explained here. So, given an input then from this input layer use this parameters $W_1 b_1$ as described, you generate the output for the next layer. And, it in this way you propagates the output to the end of this output layer to the input of the output layer and finally, you get this output. And, at each layer the description of the parameters is in this form, there is a mistake here. It should be n_i as per make our description should be align, similarly it should be.

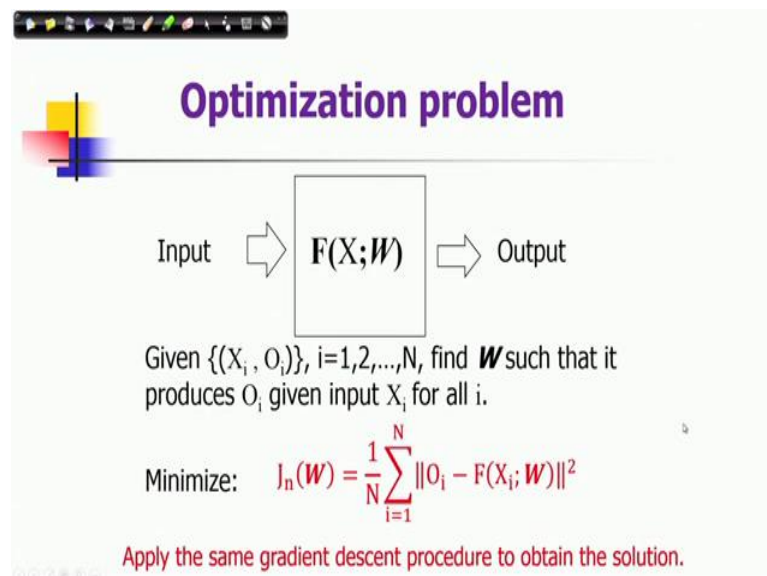
$$Y^{(i-1)} = f(W^{(i)} X^{(i-1)} + b^{(i)})$$

(Refer Slide Time: 04:47)



So, in short the model can be considered the whole set of parameters, see all the set of parameters they define, the collection there are the symbols parameters as it is represent the whole collection of parameters. So, in a model you have the parameters are defined in this model and it has its corresponding input output relation following the functions of the neurons as we discussed before. And, at each functional block this is how it is processing the input.

(Refer Slide Time: 05:31)



So, when you considered an optimization problem, we are trying to model a neural network in such a way that it satisfies the input output specifications which means that if I give you an input output specifications in that forms of X_i , O_i and these are the samples of n samples. So, you have to find out W such that it produces O_i given input X_i for all i . So, this is the optimization problem and we can use the same gradient descent approach a method to solve this. So, we define an error function which is defined in this form here.

You can see that it is considered the sum of mean square error, it is basically mean square error given the output and given the predicted output of from the model which is expressed as $F(X;W)$. So, this is a predicted output and this is the actual output. So, the square of this deviation is a single observation and then average over all these observations will give the mean square error. So, this error has to be minimized. So, you have to find out that W which minimizes this error. So, once again we can apply the same gradient descent technique.

$$\text{Minimize: } J_n(W) = \frac{1}{N} \sum_{i=1}^N \|O_i - F(X_i; W)\|^2$$

(Refer Slide Time: 07:04)

Optimization problem

Input \Rightarrow $F(X;W)$ \Rightarrow Output

Training samples: $\{(X_i, O_i), i=1,2,\dots,N\}$

Minimize: $J_n(W) = \frac{1}{N} \sum_{i=1}^N \|O_i - F(X_i; W)\|^2$

Apply the same gradient descent procedure to obtain the solution.

1. Start with an initial W_0 .
2. Update W iteratively.

$$W_i = W_{i-1} + \eta(i) \left(\sum_k (O_k - F(X_k; W_{i-1})) \nabla F(X_k; W_{i-1}) \right)$$

Stochastic gradient descent:

$$W_i = W_{i-1} + \eta(i) (O_k - F(X_k; W_{i-1})) \nabla F(X_k; W_{i-1})$$

So, that is what we can do here. So, the procedure is similar, we can start with an initial weight $W^{(0)}$, then we update W iteratively. So, it is weight means collection of weights as we have described the collection parameters, it considers weights of weights and biases of

every layer of the artificial neural network. So, the corresponding update scheme can be shown in this form.

$$W_i = W_{i-1} + \eta(i)(O_i - F(X_i; W_{i-1}))\nabla F(X_K; W_{i-1})$$

So, $\eta(i)$ takes care of all the scaling factors here and as we can see that when you are doing derivation of this particular function with respect to W . And then of course, you have to compute the derivation of F gradient of F with respect to W and you can apply once again stochastic gradient descent instead of instead of considering the O all the samples at it together.

That means, instead of considering the sum you can simply use a single sample like the previously what we have done and we can immediately update this weights and continue doing for every sample. And finally, when weights are getting you know converged; that means, there are a little changes in the values of weights after updation then we can stop.

(Refer Slide Time: 08:30)

Chain rule of computing gradient of a single neuron

Target response: t

Error: $E = (t - o)^2$

$\frac{\partial E}{\partial w_i} = 2(t - o) f'(z) \cdot x_i$

$\nabla(W) = \left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right)$

So, computation of gradient itself is a task that we need to do it here and here let me first discuss with respect to a single neuron or single perceptron. So, we can apply the chain rule very effectively. So, we know what is a chain rule, let me explain that say; this is the corresponding functional description that. So, this is an input here and you have corresponding weights.

So, this is the net input finally, after considering weighted combination then adding a bias term and then next process next part is that we have to this becomes an input to a non-linear function f . And, then it produces output O and your error function, if your target response is t then error function in this case we considered as $(t-O)^2$. So, this is a square error that is that we are considering.

So now, if you would like to perform you are interested here, you need to compute the derivation derivative of E with respect to individual weights because, we are trying to update the weights. So, how this particular response varies that we need to find out, that gradient and you should move towards or how it is affecting this error.

So, you should move towards that direction which will be minimizing the error; so, let us take us gradient directions. So, how do you compute it? Now, by applying chain rule what we can do at this part we can compute

$$\nabla(W) = \left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right)$$

$$\frac{\partial E}{\partial w_i} = -2(t - o)f'(z)z$$

So, that is how the chain rule is considered, let me give you the corresponding you know summary here also. So, we need to compute as we mentioned, we need to compute all this gradients.

(Refer Slide Time: 11:44)

Chain rule of computing gradient of a single neuron

Target response: t
 Error: $E = (t - o)^2$

Diagram: A neuron with inputs x_1, x_2, \dots, x_n and weights w_1, w_2, \dots, w_n . The neuron's output is o . The target response is t . The error is $E = (t - o)^2$.

Mathematical derivations:

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial w_i}$$

$$\frac{\partial E}{\partial o} = -2(t - o)$$

$$\frac{\partial o}{\partial z} = f'(z)$$

$$\frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial E}{\partial w_i} = -2(t - o) f'(z) x_i$$

Analytical method!
 Computed given the functional values.

$$\frac{\partial E}{\partial w_i} = -2(t - o) f'(z) w_i$$

$$f'(z) = \frac{1}{1 + e^{-z}} \left(1 - \frac{1}{1 + e^{-z}}\right) = f(z)(1 - f(z))$$

And as I was telling you that all these components we have to compute. So, you just multiply all this three things so, you get the corresponding

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial w_i}$$

Now, let us consider a particular form of $f(z)$, this is a sigmoid function. So, if I take the derivative with respect to z , it will look like this.

$$f'(z) = \frac{1}{1 + e^{-z}} \left(1 - \frac{1}{1 + e^{-z}}\right)$$

Now, incidentally this can be expressed in this form and which is nothing, but $f(z)(1 - f(z))$.

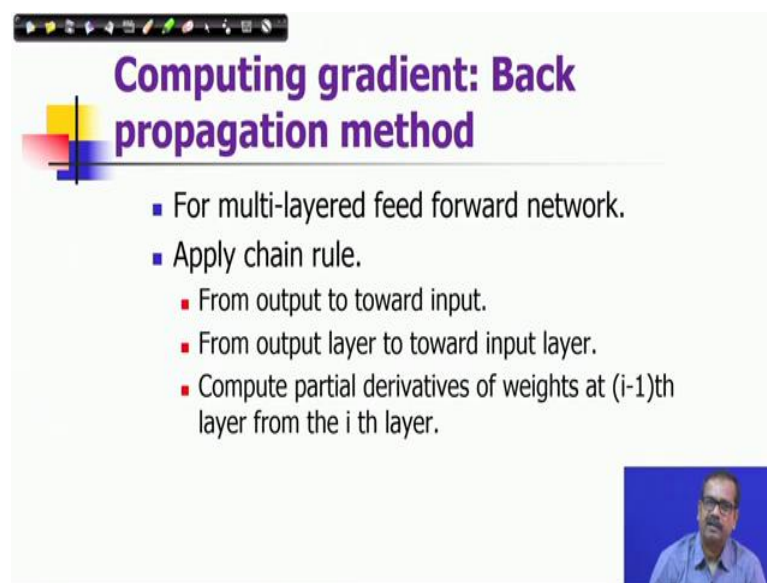
So, this is one interesting simplification, if you consider only sigmoid function. So, sigmoid function has this property and in many cases in neural networks sigmoid function is used and this particular property is effectively used in computation. Yes, if I want to compute $\frac{\partial E}{\partial x_i}$ because, we will see later on also that in some cases we need to compute that and then also we can apply chain rule.

So, you will find that the expressions would be considered here. The interesting part here is at the whole computation can be done using analytical methods. You do not have to do any numerical since you do not have to use the numerical simulation to compute. So,

sometimes for a complicated function what we observe that we give a change to the input and observe the change in the output and then use the ratio of those changes as a derivative.

But, in this case we can directly compute by giving the functional by giving the values at that point. So, only values of x w or weight centre weights and inputs they will be only required to compute this particular function. So, once you compute the functional value, then you can compute also the derivative using those values only. So, you can compute at a point that is what is advantage of this particular method.

(Refer Slide Time: 14:22)



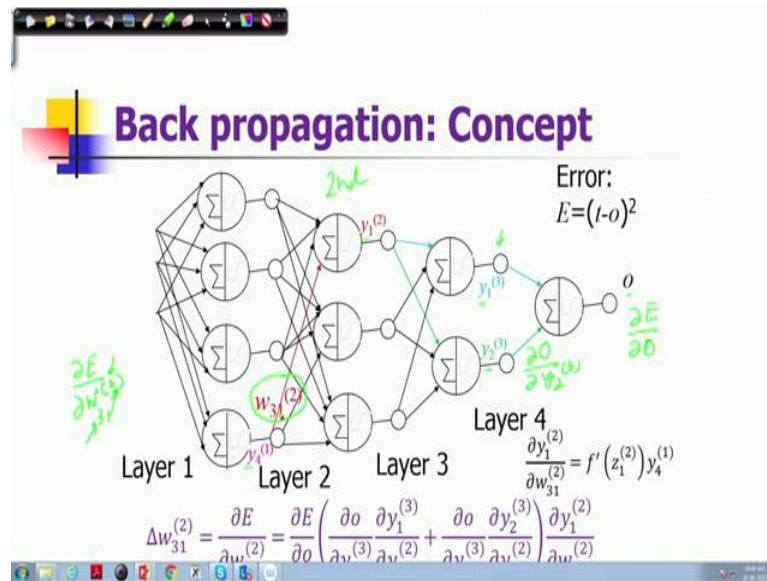
Computing gradient: Back propagation method

- For multi-layered feed forward network.
- Apply chain rule.
 - From output to toward input.
 - From output layer to toward input layer.
 - Compute partial derivatives of weights at $(i-1)$ th layer from the i th layer.

So, when you would like to compute the gradient of a feed forward network, multilayered feed forward network; we apply the same chain rule. But, in this case since there is a layered computation now, the chain rule has to be also applied following that layered architecture which means that we should compute from output towards the input. So, you should compute all the derivatives from the output end and then you should proceed towards input and compute to successive derivatives.

So, which means from output layer to input player and we can compute the partial derivatives of weights at $(i-1)$ th layer from the corresponding derivatives of the i th layer. We will see how we can do it.

(Refer Slide Time: 15:18)

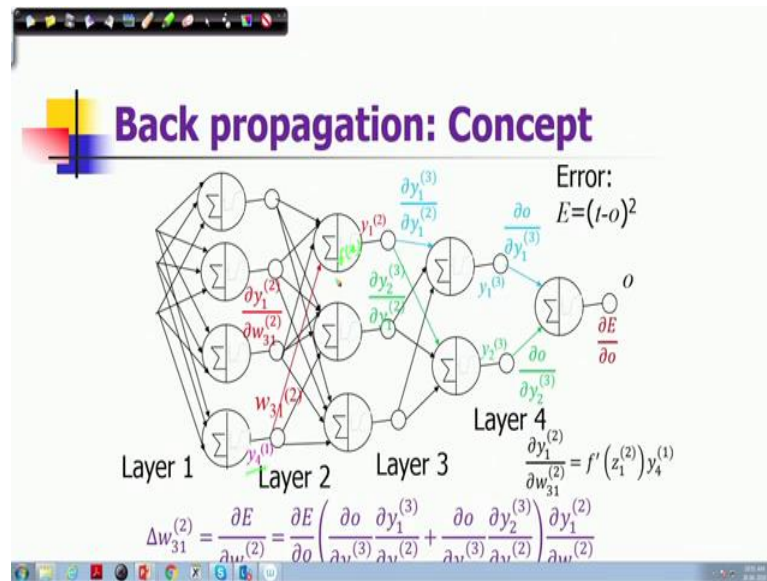


So, let us consider this particular concept here, we would like to compute the derivative with respect to this weight; you note that here in my notation 2 denotes the layer. So, this is a second layer and 3 denotes the jth neuron of the previous layer. So, this is a third neuron of the previous layer and here are the weight 3 1, it is a connectivity of the third neuron of the previous layer to the first neuron of the second layer. So, previous layer is a here so, that is a notation we are using here.

Similarly, these are y's are shown as outputs here. So, actually I have shown all these outputs which are affected by the change of this weights. So, this response is affected by the change of these weights. So, this is generating, this y_4^1 becomes an input to this particular neurons. So, it is acting like an input, but any change of this weight will affect the changes here. So, which means I need to compute here for example, $\Delta E \Delta o$ then any change here will affect outputs. So, I need to compute $\Delta o \Delta y_{23}$; that means, with respect to this.

Similarly, I need to compute the derivative with respect to this, then I need to compute this derivative of these with respect to this. So, in between there is a functional block here so, we will have to consider that. So, in this way we have to compute. So, only these derivatives are to be computed and then you can find out you know this one.

(Refer Slide Time: 17:24)

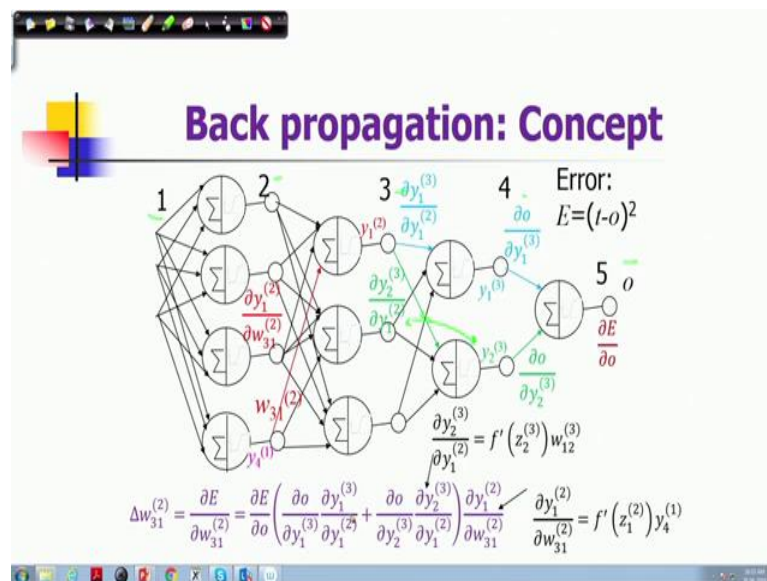


So, this is a relationship with respect to this and you note that the computation of

$$\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}} = f'(z_1^{(2)}) y_4^{(1)}$$

So, this is single neuron, this is the output.

(Refer Slide Time: 18:46)



So, we will continue this computation once again. So, it shows at the top these are the layers 1 2 3 4 5, these are the layers and then it is this computation is carried out in this form. So, $\Delta w_{31}^{(2)}$

(Refer Slide Time: 19:49)

The slide is titled "Back propagation: Delta rule". It features a diagram of a neural network with 5 layers, labeled 1 to 5 from left to right. Layer 1 has 3 neurons, layer 2 has 3 neurons, layer 3 has 3 neurons, layer 4 has 3 neurons, and layer 5 has 1 neuron labeled 'o'. Weights are shown between neurons in adjacent layers, with $w_{31}^{(2)}$ specifically highlighted. The error function is given as $E = (t - o)^2$. Below the diagram, the following equation is derived:

$$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o} \left(\frac{\partial o}{\partial y_1^{(3)}} \frac{\partial y_1^{(3)}}{\partial y_1^{(2)}} + \frac{\partial o}{\partial y_2^{(3)}} \frac{\partial y_2^{(3)}}{\partial y_1^{(2)}} \right) \frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$$

We will be discussing about a particular simple rule by which the gradients can be propagated from output towards the input direction and that rule is called delta rule. So, let me explain this particular rule by which can organize the computations very nicely. So, we measure the error as we discussed that as a square of deviation from the target t to the at response given by the network for a particular input and particular values of weights parameters of that network which is o .

So $(t - o)^2$ is a error and we would like to find out compute the gradient to of this error with respect to a derivative of error E with respect to some parameters. For example, in this diagram we are considering the parameter w_{31} . So, you note here in my notation this is a weight and you can see this 3 is a third neuron in the previous layer and 1 is the first neuron of the current layers. So, the previous layer of neuron this is the first layer and this is a second layer.

So, our convention is to denote the weight connectivity from the third neuron, output of the third neuron of the previous layer to the current layer is in this form. Similarly, the output of the third neuron of the previous layer which is the layer 1 is denoted here y_3 1 and also you can see output of the first neuron of layer 2 is denoted in this form. Similarly,

output of layer three first neuron of layer 3 is denoted here of layer 3, it is also denoted that is the second neuron in this form.

And finally, this is of course, the fourth layer and only one neuron is there and fifth layer is just output response which is redundant and we are not denoting with any other symbol other than o . So, this is a convention that we are following and we are showing all the necessary responses which are affected by the change of weights w_{31} . So, when we compute the gradient with gradient of error E with respect to w_{31} this weight then these are the variables which will play a role in determining this gradient.

So, let us find out how this gradient value is dependent. So, you can see here we would be like to measure this value and which will give me the corresponding updates of the weight w_{31} parameter. And so, you applying the chain rule. So, first we compute the gradient of E with respect to output o , then gradient of output o with respect to this gradient y_{12} and then subsequently gradient y_{12} .

So, this is that value, similarly these two added here. So, because of the linear operations this is other value and finally, when you are computing the gradient with respect to w_{31} , it is $t_{y_{12}}$ with respect to w_{31} . So, these are the components which we need to compute or which these expressions which we need to find out given this particular responses at this instance. So, in this case we have already discussed how to compute the gradient of y_{13} with respect to y_{12} .

Suppose we know the gradient of output o with respect to y_{13} and that is we are calling that is a value δ_{13} . That means, this is the accumulated gradient till this point, till the output of the first neuron of third layer we denote in this fashion. Similarly, the gradient at this point will be denoted as $\delta_1 \delta_2$. So, it is δ_{23} so, that is a gradient at this point, that is my notation and that is the definition of delta.

So, delta is the accumulated gradient till a output till the output of certain layer and the corresponding convention of writing delta in this fashion. Similarly, you can write the gradient changes along a edges also particular a edges, we will see how we can do it. So, let us expand this particular quantity; that means, gradient of y_{13} with respect to y_{12} . So, let us explain this particular quantity which is the gradient of y_{13} with respect to y_{12} .

(Refer Slide Time: 26:02)

Back propagation: Delta rule

Error:
 $E=(t-o)^2$

$\frac{\partial \delta_1^{(3)}}{\partial y_1^{(2)}} = f' \left(\frac{\delta_1^{(3)}}{w_{11}^{(3)}} \right) w_{11}^{(3)}$

$$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o} \left(\frac{\partial o}{\partial y_1^{(3)}} \frac{\partial y_1^{(3)}}{\partial y_1^{(2)}} + \frac{\partial o}{\partial y_2^{(3)}} \frac{\partial y_2^{(3)}}{\partial y_1^{(2)}} \right) \frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$$

So, these are how they are related, since in between you have the corresponding non-linearity which is $f(z)$ and that would play into role. So, we can write it as the we can write this fact as say $f'(z)$ and then we know that there is a weight which is also connected here. So, it is since it is a linear combination of this. So, if this weight as per our convention it is w this is a first neuron. So, this is 1 and this is also 1 because this is the first neuron of third layer and this is 3.

So, according to our convention this is weight so, this is linearly related. So, this is a scale $y_1^{(2)}$ is scaled by w_{12} and then it is contributing to the net input which is a again determining the $y_1^{(3)}$. So, we can write $\delta y_1^{(3)} \delta y_1^{(2)}$ as $f'(z)$ that is also at that point. So, we will see whether we have taken any conventions. So, we can write it also see $f'(z)$ 1 and it is layered 3 $f'(z)$ 1 3 into $w_{11}^{(3)}$ then 3. So, this is the expansion, similarly you can do this expansion and later on we will see how the chain rule is defined with respect to this.

(Refer Slide Time: 27:59)

Back propagation: Delta rule

Error:
 $E = (t-o)^2$

$\frac{\partial E}{\partial y_2^{(3)}} = f'(z_2^{(3)})w_{12}^{(3)}$
 $\frac{\partial E}{\partial y_1^{(3)}} = f'(z_1^{(3)})w_{11}^{(3)}$

$\Delta w_{31}^{(2)} = \delta_1^{(2)} f'(z_1^{(2)}) y_3^{(1)}$

$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o} \left(\frac{\partial o}{\partial y_1^{(3)}} \frac{\partial y_1^{(3)}}{\partial y_2^{(2)}} + \frac{\partial o}{\partial y_2^{(3)}} \frac{\partial y_2^{(3)}}{\partial y_2^{(2)}} \right) \frac{\partial y_2^{(2)}}{\partial w_{31}^{(2)}}$

$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o} \left(\frac{\partial o}{\partial y_1^{(3)}} f'(z_1^{(3)}) w_{11}^{(3)} + \frac{\partial o}{\partial y_2^{(3)}} f'(z_2^{(3)}) w_{12}^{(3)} \right) \frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$

Delta rule:

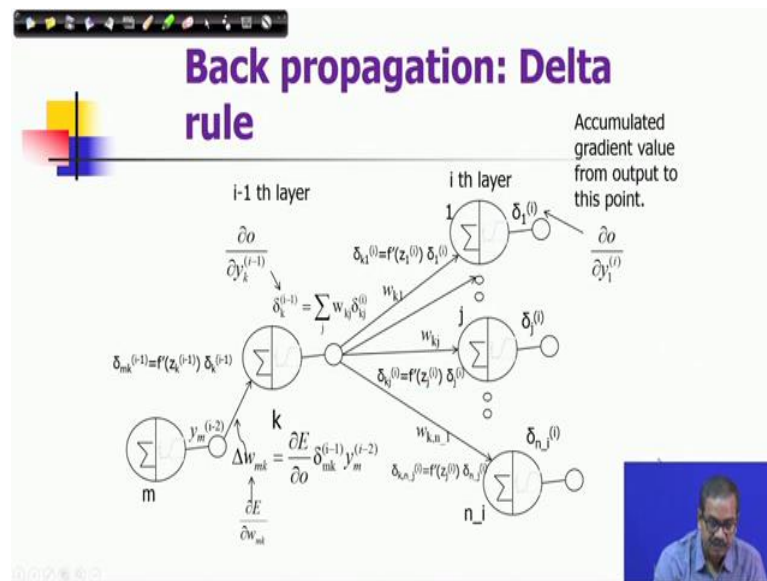
So, let me proceed. So, this is what as I mentioned we will compute this part and similarly we will compute the other one and this will expand. So, as I mentioned this is expanded into this form and this is expanded into this form, that is coming from here. This one is coming from here and this one is coming from here. Now, we define the delta 1 1 as I was mentioning delta 1 1, like I have defined earlier the delta this is delta 1 3 and at this edge this is delta 1 1 3 and that gets multiplied with w 1 1 3.

Similarly, at this stage this is delta 1 2 3 and that gets multiplied with w 1 2 3. So, we will find out how this is happening. So, this is what; this is what is delta 1 1 3 as I have mentioned, this is getting multiplied what with w 1 1 3 and this is w delta 1 2 3 and this is getting multiplied with w 1 2 3. And, that would give me the corresponding here that would give me delta 1 2 so, this is the delta rule. So, if I write it so, we can define the delta rule in this fashion as I was mentioning that delta 1 2.

So, this is what this is delta 1 2 is equal to then. So, this is added this is an this is equal to the weighted addition of delta 1 1 3 and delta 1 2 3 where the weights are the weights of the corresponding edges. So, that will be more clear when I show it here; so, this is a delta rule. So, once you get delta 1 2, similarly in the same way you can also get the delta value here. So, it means it has to be multiplied with f dash. So, from here it should be multiplied with f dash it should be z 1 2 and then you are computing actually the response of this point. So, it is delta y 1 2 delta 3 1 2.

So, it is this multiplied by $f'(z_{12})$ into the y_{31} because, when you are computing the gradient at this point in the multiplication factor with respect to this parameter will be this response. So, that is what we will be writing here. So, you can see that it is δ_{12} multiplied with $f'(z_{12})$. So, this is δ_{12} this is $f'(z_{12})$ in that path and then you are multiplying it y_{31} , that would give you this quantity and which is used for updating this weight.

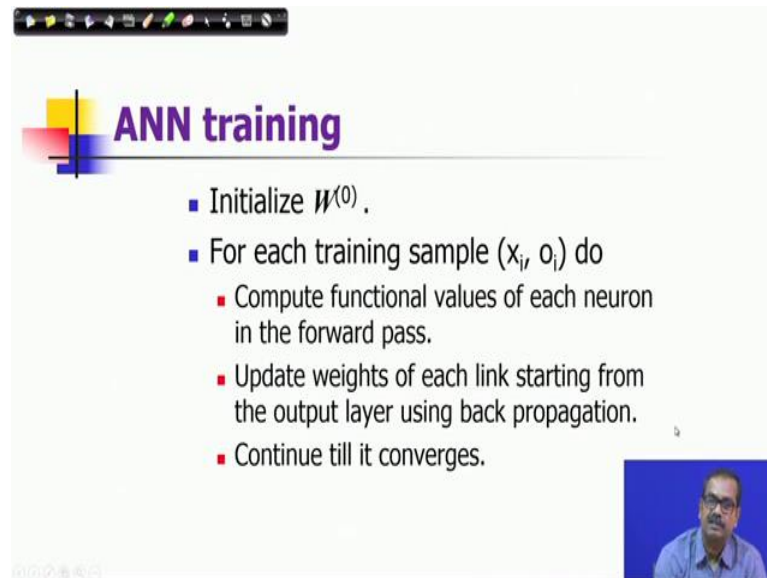
(Refer Slide Time: 31:27)



I think this is explaining this diagram explaining in a bit in a clearer way. So, what it is doing let me see, let me explain; see you compute as I mentioned δ_{1i} at this point. So, which is the gradient accumulated gradient value from output to this point, see this is δ_{ji} . So, in that you are computing all these accumulated gradient value and then it is first it is propagated. So, δ_{ki} is multiplied with respect to $f'(z)$ so, this is δ_{ki} similarly δ_{kj} is this.

So, you take the weighted sum of all this. So, you take the weighted sum that gives you there the corresponding accumulated gradient value from the output to this point; so, at this point. So, and next you will propagate in this fashion; so, it will again propagate in this fashion. And so, the great way the update of weight should be this particular know the $\delta E \delta o$ into this because it is from the weight output response. So, this is our update, in this way weights are you know updates are computed at every branch. So, this is what is at its semi computing in this gradient.

(Refer Slide Time: 32:47)

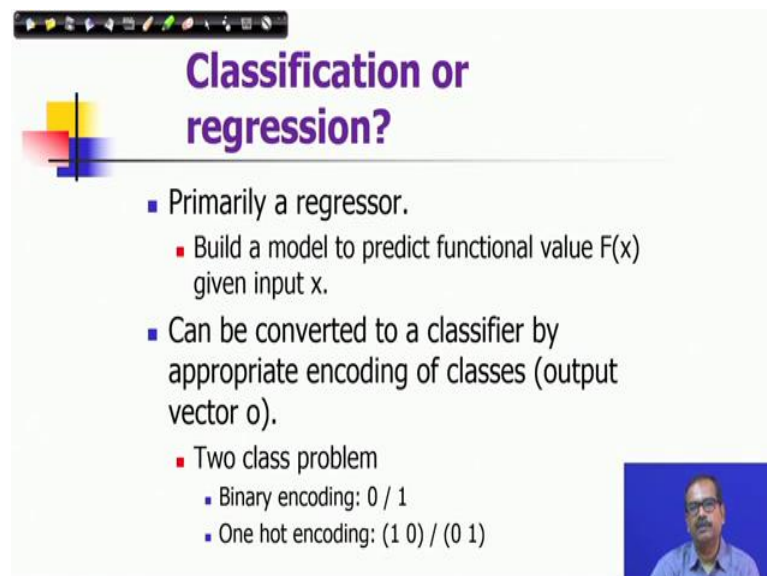


ANN training

- Initialize $W^{(0)}$.
- For each training sample (x_i, o_i) do
 - Compute functional values of each neuron in the forward pass.
 - Update weights of each link starting from the output layer using back propagation.
 - Continue till it converges.

So, the in this way you are computing the gradient of with respect to the corresponding weight, with respect from the gradient of the error and that is a how we get the gradient vector. And, then the algorithm follows in the same fashion that for each training sample you compute functional values of each neuron in the forward pass. And, then update weights of each link starting from the output layer using back propagation and then it should continue till it converges.

(Refer Slide Time: 33:21)



Classification or regression?

- Primarily a regressor.
 - Build a model to predict functional value $F(x)$ given input x .
- Can be converted to a classifier by appropriate encoding of classes (output vector o).
- Two class problem
 - Binary encoding: 0 / 1
 - One hot encoding: (1 0) / (0 1)

One of the things we would like you to know mentioned here is that when you are doing artificial neural networks using this model whether it is a classification or regression model. Now, you can see primarily it is a regressor, it builds a model to predict a functional value $F(x)$ given input x . But, you can convert this model, you can use this model also as a classifier by appropriate encoding of classes; that means, your output vector would be those encoding of classes.

For example, if you have a two class problem, you can consider a binary encoding. So, you can consider either 0 or 1 or you can also consider one hot encoding; that means, there are two neurons output neurons one of them will be 1, other will be 0. For the other class other will be 0 another is 1. In one hot encoding it is idea is that if there say n classes that could be represented by m such you know binary variables n bits and only one of them would be 1 for a particular class, rest will be 0.

(Refer Slide Time: 34:36)

Evaluation of a classifier

- Two class problems.
 - Positive class and Negative class
 - TP: Set of +ve samples predicted +ve.
 - FP: Set of -ve samples predicted +ve.
 - TN: Set of -ve samples predicted -ve.
 - FN: Set of -ve samples predicted +ve.

	AP	AN
PP	TP	FP
PN	FN	TN

Accuracy: $(TP+TN)/Total$

Precision: TP/PP
 Recall: TP/AP

Sensitivity / Recall: $TPR = TP/AP$
 Specificity: $TNR = TN/AN$

F-Score: Harmonic mean of precision and recall

$$F = \frac{2}{\frac{1}{Prec} + \frac{1}{Recall}} = \frac{2 \times Prec \times Recall}{Prec + Recall}$$

It is necessary that when you design a classifier you should evaluate a classifier, particularly for a supervised supervised classification problem. So, we will discuss about you know some methods of evaluation, some measures of evaluation. Consider you have two class problems and there are positive and negative classes. Now, there are several possibilities after classification like here in this diagram in this table I am showing the classification what has been predicted by the model.

So, you are seeing where it is these are the predictive predicted positive and predicted negative. So, it can predict other positive or negative. So, it could happen that the sample is actually positive and it is also predicting positive that would give you two positive or it is negative, but it is predicting positive. So, it is the class of false positive. When it is positive actually positive, but it is predicting negative then it is false negative and in the other way when it is negative actually, but it is predicting also negative.

So, this is an desirable situation, this is true negative. So, this true positive and true negatives are desirable outcomes those numbers should be high whereas, these two numbers should be less. So, in a measure we use this numbers there are different measures, like if I consider accuracy of a classification we consider what is the total number of predictions which are true either positive or negative. So, it is true positive plus true negative by total number of you know samples which have been tested, there is there are other measures like precision and recall.

So, in precision you see that what is a fraction of predicted positives are true and recall is what is a fraction of actual positives are true. There are other measures like sensitivity which is the same as recall, these are used particularly for the medical world. So, or we can considered it also true positive rate. So, it is true positive by actual positive and specificity is true negative rate which is true negative by actual negative.

And, you can combine these scores two scores into one F score which is called harmonic mean of precision and recall so, which is has been defined here. So, higher the F score better is the classification, when you consider it in combination.

(Refer Slide Time: 37:14)

Evaluation of a classifier

- Multi class problems.
- Confusion matrix

True classes →

	ω_1	ω_2	ω_3
Predicted classes ↑	ω_1		
		ω_2	
			ω_3

Accuracy: (Sum of diagonal) / Total

For a multi class problem we can use a confusion matrix where, once again you have you can I have shown here that they these are the true classes and this is the predicted classes. So, in the diagonal term if this numbers are high, those are the desirable outcomes. All other terms there are some kind of errors are there because, it belongs to actual class omega 1, but your prediction is omega 2 or omega 3. So, the accuracy measure which means which will be sum of diagonal by total.

(Refer Slide Time: 37:43)

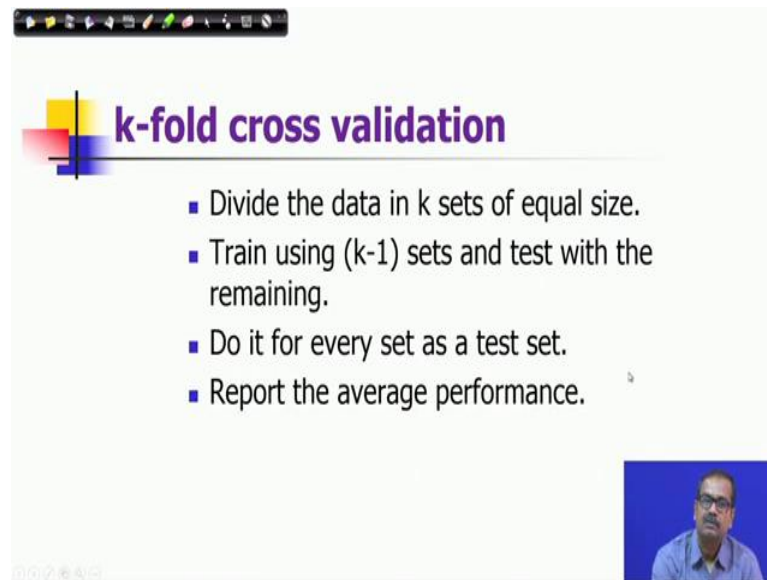
Cross validation

- For supervised classification.
- Separate training and test data.
- Train network using training data.
- Evaluate using test data.

© 2018

There are methods of you know testing the; testing the performance of a classifier. One method is called cross validation and once again it is applicable for supervised classification. What we do that we separate training and test data. Then we train network using training data and then evaluate using test data.

(Refer Slide Time: 38:08)

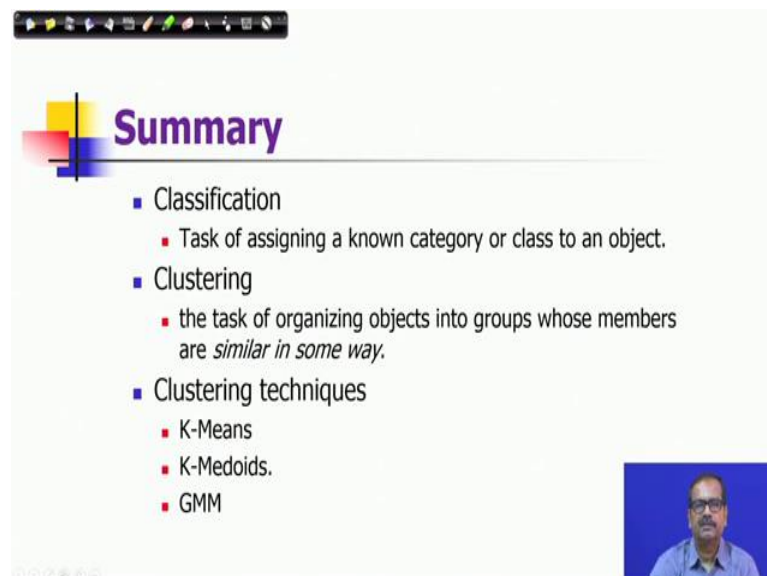


k-fold cross validation

- Divide the data in k sets of equal size.
- Train using $(k-1)$ sets and test with the remaining.
- Do it for every set as a test set.
- Report the average performance.

So, for k fold cross validation, we divide the data in k sets of equal size and we train using k minus 1 sets and test it with the remaining. And we do it for every set as a test set and take the average. So, report the average performance.

(Refer Slide Time: 38:27)

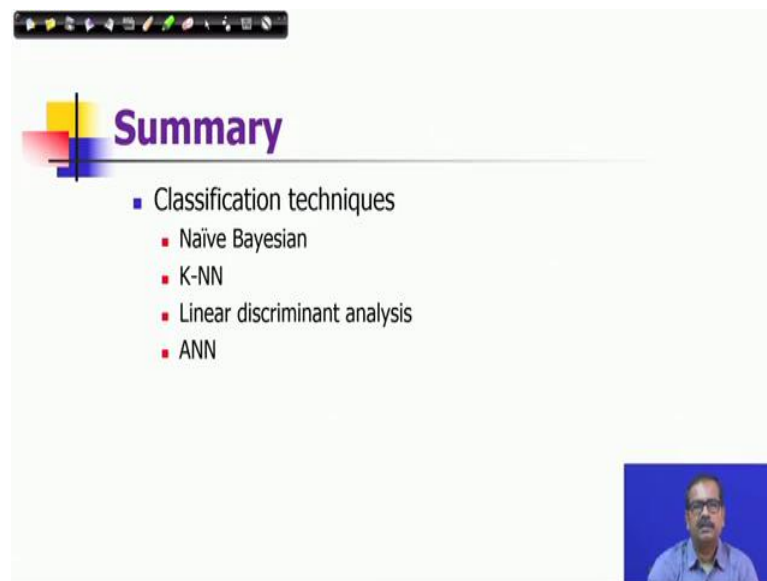


Summary

- Classification
 - Task of assigning a known category or class to an object.
- Clustering
 - the task of organizing objects into groups whose members are *similar in some way*.
- Clustering techniques
 - K-Means
 - K-Medoids.
 - GMM

So, here we come to the end of my talk for particular topic. So, just let me summarize whatever you have discussed under the topic of classification and clustering. So, as we know that classification is the task of assigning a known category or class to an object whereas, clustering is a task of organizing objects into groups whose members are similar in some way. So, clustering techniques there are several clustering techniques we discussed like K means, K medoids or Gaussian mixture model.

(Refer Slide Time: 39:04)



Summary

- Classification techniques
 - Naïve Bayesian
 - K-NN
 - Linear discriminant analysis
 - ANN

And, for classification techniques we considered Naive Bayesian classification scheme, then K nearest neighbor classification scheme, linear discriminant analysis and finally, artificial neural network models. With this let me stop here and we will continue our discussion for our lectures of the next topic.

Thank you very much for listening to this talk.

Keywords: feed forward network, chain rule, back propagation, cross validation.