

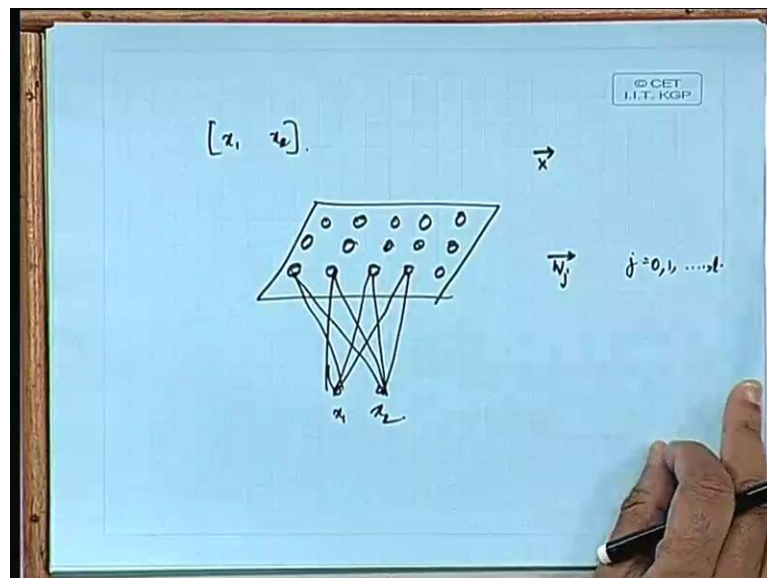
Neural Network and Applications
Prof. S Sengupta
Department of Electronic and Electrical Communication Engineering
Indian institute of Technology, Kharagpur

Lecture - 37
Vector-Quantization using SOM

The case, Vector Quantization using Self Organizing Map this very important and this is based on very important property of the self organizing map. And that is, about the approximation that it makes from a continuous space to the discrete output space. So, making use of that property, what is known as a vector quantization, that can be carried out.

So, we will be taking mostly about that in this lecture. But, before that I would like to clear some of the doubts, which might have cropped in to the students mind in the last class. That is about the way the topological order is taking place. I think you need to get a feel of how the topological ordering actually happens. So, we will try to clarify the issue today. Now, as we were taking the kind of example in the last class. Let us say that we have got an input, which is consisting of just two elements.

(Refer Slide Time: 02:14)



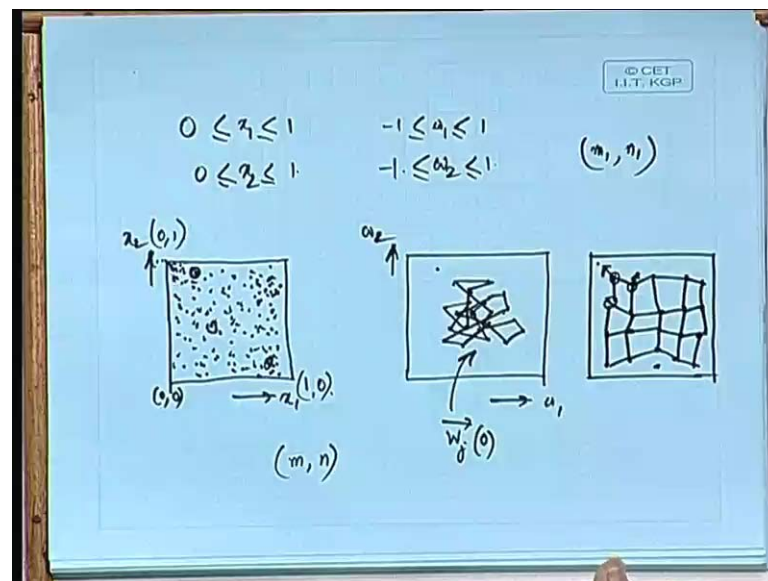
So, it is a two element vector let us say x_1 and x_2 these are the elements of the input. And supposing in the, output we have got a lattice of neurons. So, let us say these are the

output neurons, organized in some lattice. Let us, say two dimensional lattice, we can have one dimensional lattice also. And let us say that, here we have got the two inputs x_1 and x_2 .

So, what happens is that x_1 and x_2 will be connected to all these neurons at the output. So, x_1 and x_2 will be connected to all this. So, with each of these output neurons there will be a two element vector W_j . So, a two element W_j vector will be associated. Where j is equal to 0 1 upto l , where there are l number of neurons in this output layer correct.

So, we have got two element input. That is, the x vector is two element and W_j vectors individually are all two elements. So, let us try to visualize this picture, so we will be having the input variables. The input vectors should be picked up from a two dimensional space.

(Refer Slide Time: 03:48)



And let us say that, we pick up the x_1 and x_2 from a space, which is defined as follows. Say we take x_1 lying within 0 and 1. And, we take x_2 lying within 0 and 1. So, we actually create a space like this let us say that, this is our point of origin that is 0, 0. And in this direction we plot x_1 , and in this direction we plot x_2 . This is say x_1 is equal to 1. So, this is the 0.10 and this is at x_2 is equal to 1 which corresponds to 0 1, so this is the complete space.

So, whatever input vector we pick up, we have to pick up from this space. And similarly, we can create a weight space. Let us, say that the weights are restricted that for all the W_j vectors, we choose that W_1 is lying within minus 1 to plus 1, W_2 is also lying within minus 1 to plus 1, let us say, it is like that. So, what happens is that if we plot the weight space let us say that, here we have got W_1 and here we have got W_2 .

So, may be that for neuron 1 the W_1 , W_2 initial vector will lie over here, W_2 it is here, W_3 it is here, like that different vectors will be lying in different places. Now, what we are doing initially is that, we are only trying to; if you have any difficulty in understanding the concept of the special location of the neurons.

Let us forget about that. Let us say, that we only do some indexing of this neuron, just some indexing. And let us say that because there is a, because we want to organize ultimately that in the form of a two dimensional lattice. So, let us say that we follow a two dimensional indexing system that means, to say that the neurons will be denoted by let say m comma n , where n is the index in the x_1 direction and m and n is the index in the x_2 direction, or in x and y direction.

So, in two dimensions their indexes are called as m and n . So, we index all these neurons. So when we index all these neurons that means, to say that we are in a position to determine for any neuron. If we happen to pick up any neuron having an index m comma n then, surrounding that neurons there will be all other neurons in a neighborhood. Like, $m-1$ comma n , m , $n-1$, $m-1$, $n-1$ all these things will be the neighbor to the m , n neuron.

So, let us forget about the connectivity's for the moment. Let us say that, now what we do is that we simply try to indicate the positions of the initial weight vectors of the neuron. Let us say that, we pick up the neuron $0, 0$ it's initial weight vector supposing is here at this point. Supposing we pick up the next one that is $0, 1$ it's initial vector, it is initial weight vector is here in the W_1 , W_2 space.

Supposing the next ones initial vector is here, like that the initial vectors will be organized in different places. Because, initially it is not according to any arrangement, we are just randomly initializing it. Now, what we can do is that, just to depicted properly in our diagram. See normally what we do is that, we try to connect the neighbors by lines. So, those lines are our imagination, just to indicate the connectivity's.

So, supposing if this is the 0 0 neurons, neurons having index 0 0 the vector corresponding to that. Now, this is connected to the neuron at 0 1, this is connected to the neurons at 1 0. So, 0 1 supposing is here and supposing 1 0 is lying here. Now, 1 0 is connected to 1 1, may be that 1 1 vector is lying here. Now, again 1, 1 is connected to 2 1 and supposing 2 1 is lying here. And supposing, that 2 1 is connected to 2 2 it is lying here, and supposing 2 2 is connected to 1 2 and 1 2 is lying here.

So, like this we will be tracing out some zigzag some arbitrary zigzag paths. If we try to connect all these neurons, whose weights are very oppositely arranged so, these lines that we have drawn are only to show our imaginary connections, between all the neighboring m 's. Now, what happens is that, given this type of an initial weight vector we just place some, we just happen to pick up some input vectors.

Supposing the input vector, that we pick up 1st is this one, the 2nd input vector that we pick up is this, the 3rd that if we pick is here, 4th that we pick up is here, 5th is here, 6 is here, 7th is here, 8th is here. Like this, the input vectors are we large number of input vectors. And supposing this is a kind of an input vector arrangement, that we have got that we pick up the input vectors.

So, all the positions of the different input vectors are indicated over here. Now, look at this here I tried to make the x 1 the distribution of the input vectors to be uniform in this x 1 x 2 space. But, I cannot make it exactly uniform like you can very clearly see that in some regions the concentration of the input vectors is more. So, the probability of picking up an vector from this space is much more than, the probability of picking up an input vector from this space.

So, naturally the statistical distribution of the input vectors is not uniform. Actually it is varying now, what will happen is that after feeding these inputs, we try to train this network. And here, training means what? That supposing, we have fade the first input vector. Supposing this is the ((Refer Time: 10:45)) arrangement initially, so these are all the arrangements of $W_j 0$, so $W_j 0$'s arrangement is shown out here.

Now, what I will do is that supposing, I feed the very first patterns. Supposing I pick up the first input vector and I feed it. Now, all these l neurons, which are there here they will compute with each other one of them will be the winner. Supposing some index let say m 1 comma n 1 that is the winning neuron index.

Now, it will not only the winning neuron, but also its surrounding neurons which will get excited. So, $m-1$ minus n minus $m-1$ minus $n-1$, all these neurons will get excited. And all the neighbors will get excited in accordance with their neighborhood functions, which I tried to describe yesterday. So, what happens that in the process all these $m-1$ $n-1$ neurons all of that one of them, will be the winner and the winner will be taken closest to the x vector correct the x vector, which I have picked up.

So, even though initially it was here it will try to move towards here. Whether it will actually move or not will be dependent upon the η the learning rate. But, it will try to move closer to this and not only that, it will try to drag along with all the other neighboring neurons. Now, you feed this vector someone else will be the winner and someone else, will try to drive its neighbor out there.

Pick up here someone else, will be the winner it rest to pick up there. So, ultimately when you feed up feed large number of patterns. You will see that ultimately it will follow the topological distribution. So, that topological arrangement, that we are having mind you is in the weight space. So, the ultimate weight space arrangement will be like this, had it been absolutely uniform perhaps. We would have expected a uniform, the grid would have looked absolutely uniform, but now the grid may be looking something like this.

So, the grid will look not exactly the two dimensional that is, a regular two dimensional lattice it would look like a regular two dimensional lattice, when the distribution is really is picking uniform. But, it is not it there is some non uniformity here. So, once this kind of a topological arrangement is there that means, to say the first phase of adaptation is over. Then what will happen is that we will go through the convergence phase.

Now, what happens that it has picked up the rough locations, but there are some may be that we have placed the neuron over here in the weight space. But actually, the input vectors are sometimes distributed here also. It is close to this, but not exactly, so what it will try to do is that in the convergence phase, there will be some internal, some minor, or some small movements of all these vertices will take place.

So, this vertex may try to push up to this place, this vertex may try to push up to this place like that, during the convergence phase the same topological arrangement will be preserved. But, the vertices of these will try to adjust to each other in order to fill up the

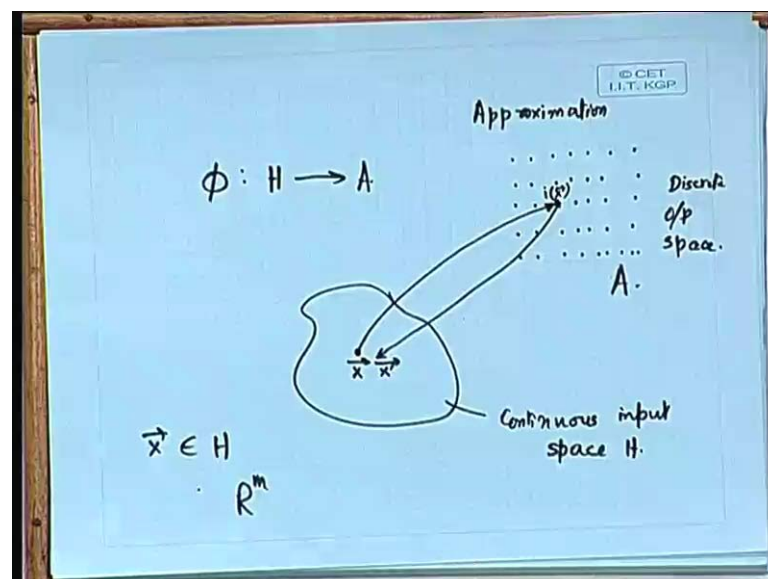
whole space, the space in which, this $x \ 1 \times 2$ vector is distributed. So, this is the kind of clarification that, I wanted to give and now, I think that the topological ordering concept should have gone clear in your mind.

In fact, I was telling in the last class that, when I had first talked about the topological ordering and try to give this initial arrangement, I was looking at the face of the students and I found that something has definitely gone above the head. And now, when I see the same peoples face once more, I find that now they are convinced that, yes there is a definite topological ordering that is followed.

So, now that we have followed one of the very important properties of the self organizing map. We can move over to the actual topic of the day, which is about the vector quantization. Now, vector quantization is something, which is followed in the data compression. In fact, just to tell you that neural networks are being increasingly used in applications, which involve data compression.

And we have seen , we had got a feel about it when, we were talking about the principle component analysis, because there we saw that there was a essentially a dimensionality reduction that, we did by picking up the largest of the Eigen values. And then, projecting the input vector in to those eigenvector, which are corresponding to those largest Eigen values. So, that was very useful and likewise even the self organizing map also, can be used for the data compression. In fact, we can present the concept in this.

(Refer Slide Time: 16:55)



Let us say, that we have got a continuous input space. So, here what I draw is a continuous input space, input space H . So, this H is actually an m dimensional space, but I have to somehow draw it. So, this is ((Refer Time: 17:17)) some space that I have created in m dimension. And out of this, I can pick up my input vector. Supposing I happen to pick up an input vector x vector from this continuous space mind you, this is continuous.

Now, what I have got in the self organizing map is a, discrete output space, because I have got only limited number of neurons. I have got either, 10 by 10, or 20 by 20, or 30 by 30 whatever be the number, but ultimately it is a limited space, ultimately, it is a discrete output space, so we have got a discrete output space. Let us say, that this is the discrete output space that we have, I am unable to draw it in a regular way anyway.

So, supposing this is the discrete output space, and we call this discrete output space. So, this is the discrete output space, we call this space as the space A . So, now what we are having is that this x vector, the input vector definitely belongs to the continuous m dimensional input space H . So, H is actually \mathbb{R}^m , H is in the \mathbb{R}^m space. So, this is, so x vector is belonging to H and the discrete output space is created here.

So, essentially what we are having as a feature map is a mapping essentially, the what the feature map or the self organizing map, is trying to do is to map this x vector in to this discrete output space is not it. So, essentially it is a non-linear mapping definitely. So, we call this mapping function as the ϕ function.

So, what ϕ does is to map. from the input the continuous input that is H space in to the discrete output space, that is the A space. So, now this is the way in, which we can imagine and let us say that, this x vector is actually mapped in to some point. Let us say here, supposing after feeding this x vector this is, the winner this neuron happens to be the winner, so we get the i of x .

That means, to say in the competition this is supposing the i of x the winner. So, this is the winner position, which will be taken very close to this x vectors position, but it is mind you an approximation, because it is in the, this one is in the discrete space. So, that is why it is an approximation of the continuous space. So, what the discrete output space is doing is an approximation.

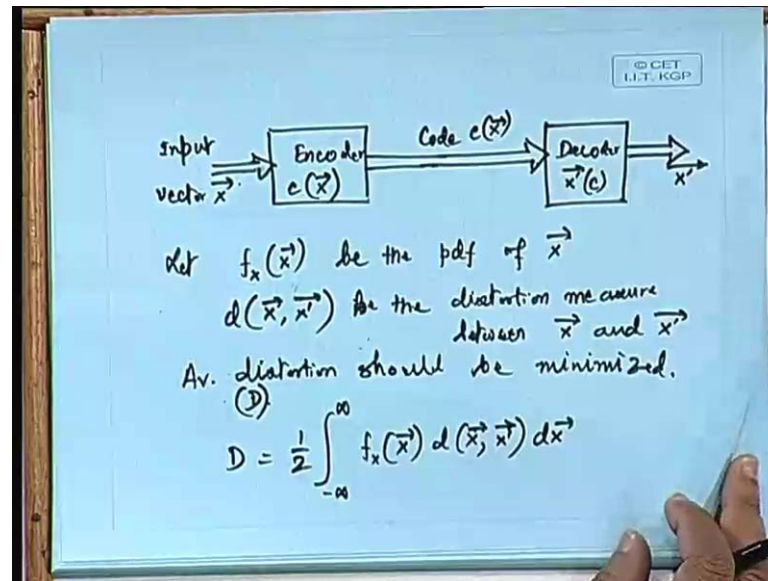
So, now what happens is that now, that $i \cdot x$ is the winner and $i \cdot x$'s position is arranged as close to possible as that of the x vector. Now, if using this $i \cdot x$ I try to do a reverse mapping, reverse mapping that means, to say that from the discrete output space, if I try to map back again to the input space, given this point $i \cdot x$ will I come back exactly to this x vector no, I will be coming in to some other point x prime vector let us say, I will be coming to x prime vector, which will be different from that of the x vector.

So, the distance that exists between this x vector and the x prime vector will be the error that, I incorporate and that is the error in the process of mapping it in to the discrete space. So, what we are doing is effectively the continuous space, where there can be a large set of values is A is being mapped in to the, discrete space having a limited number of neurons. And my codeword will be consisting of, how many code words as many neurons as I have.

If I have hundred neurons 10 by 10 then, I can only encode all these input vectors using one out of this 10 by 10. So, essentially here, I have got a limited resource from, which I try to encode a large continuous space. I may be having let say, 25 dimensional input vector. Supposing m that we are choosing is of dimension 25, but I may be mapping it to a 10 by 10 space. So, essentially what I am doing is that from a large space, I am going over to a small space. And I will be able to do a data reduction, or data compression in this process.

So, let us now go more in to this approximation theory. In fact, this you can look at it this mapping from x vector to $i \cdot x$ and then, the inverse mapping from $i \cdot x$ to x vector. This whole process you can think in terms of, as if to say that this, x to $i \cdot x$ mapping is a process of encoding it is being encoded in to the discrete output space. And the reverse mapping from, $i \cdot x$ to x prime vector is the process of decoding and x prime vector is the reconstructed vector for x .

(Refer Slide Time: 23:17)



So, essentially what we are imagining or modeling, is that we have got an input vector x here. And then, we are having an encoder which, we are calling as the let say c of x , so c is the encoder function. And we are getting a code, which is c of x vector and then, this will be passed through a decoder. Now, I am not modeling the any channel noise now, actually speaking in the communication what we are having is that after the encoder, it is put in to the communication channel and, the communication channel can have some noise in it.

So, that we are for the time being assuming a noise free model. Now, what we are doing here is, that this code c of x will be pass through the decoder. And the decoder function, we are writing as x prime vector c . So, what the decoder will try to do is to get back this vector x , but it will not get x exactly it will get x prime. So, this is the model that is followed by this type of a mapping. And supposing, what we are having is that this distribution, we let us now assume some distribution of this x vector in the continuous space H .

So, let f_x of x be the probability density function, let f_x of x be the PDF of the x vector. Now, the optimum encoder, decoder scheme we can call this thing as optimum, such that supposing we choose a distortion measure, that the distortion measure between x and x vector. And what we want to do, we want to minimize this distortion is not it. So, let d of x

and $\|x - x'\|$ is the distortion measure, or let this be the distortion measure between x vector and x' vector.

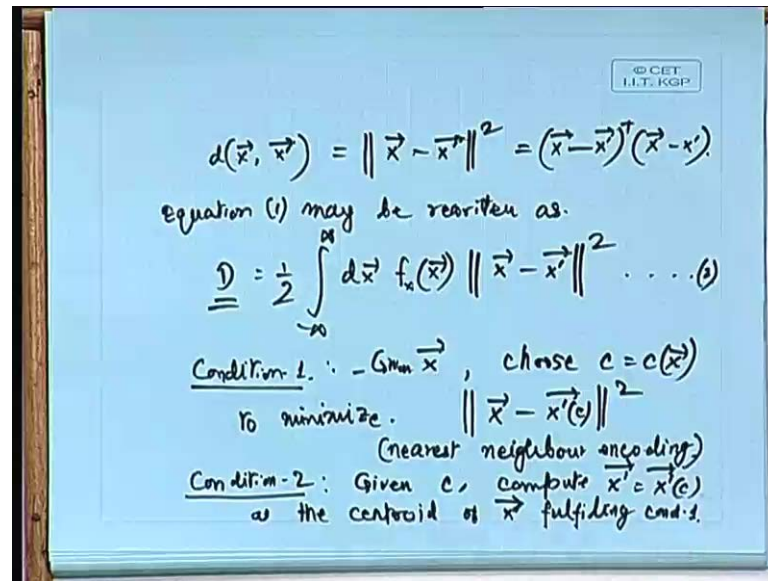
So, we want to minimize this distortion of course, we do not want to minimize just the distance between these two x and x' vector, but we want to minimize the averaged distortion. Because, actually speaking we will be having, ((Refer Time: 26:07)) we will be picking up a large collection of these x vectors. So, on an average we would like to see that this distortion is minimized. So, our c function and the decoder function, so the encoder and the decoder functions should be chosen in such a manner, that our average distortion should be minimized, so that is our motive, so average distortion should be minimized.

So, let us first express the average distortion in terms of, what we have already defined in quantities. Now, average distortion let us call it as D . So, we are indicating this average distortion to be equal to integral of, I am again introducing this half, you know the reason that naturally when, I talk about half I must be doing some differentiation and there must be some squared quantity, and I just want to nullify this half with the two that I will get from the differential.

So, always that is why we will be writing it as half anyway, within the integral side what we have to do? is to take $f(x)$ of x , which is the probability, probability density function of x vector, this you multiply by $D(x, x')$ the distance between this and x' vector. Now, x vector is an m dimensional vector, so what we have to do is to integrate in the m dimensional space.

So, the integration that, I am considering is over the entire space. So, if this H is the continuous space, we have to integrate it over the entire space. So, that is why in the m dimension I am showing the limits of these to be minus infinity to plus infinity, because it is a continuous input space assume, so this will be the expression of the average distortion. So now, what is the form of this $d(x, x')$, now the best measure that or one of the popular measures, that we take for $d(x, x')$ is the Euclidean distance between the two.

(Refer Slide Time: 28:36)



© CET
I.I.T. KGP

$$d(\vec{x}, \vec{x}') = \|\vec{x} - \vec{x}'\|^2 = (\vec{x} - \vec{x}')^T (\vec{x} - \vec{x}')$$

Equation (1) may be rewritten as.

$$D = \frac{1}{2} \int_{-\infty}^{\infty} d\vec{x} f_x(\vec{x}) \|\vec{x} - \vec{x}'\|^2 \dots (6)$$

Condition-1: Given \vec{x} , choose $c = c(\vec{x})$ to minimize $\|\vec{x} - \vec{x}'(c)\|^2$
(nearest neighbour encoding)

Condition-2: Given c , compute $\vec{x}' = \vec{x}'(c)$ as the centroid of \vec{x} fulfilling cond.1.

So, d of x comma x vector, d of x comma x prime is equal to the Euclidean norm x minus x prime square. So, this is equal to just the matrix vector form of, it is to be written as x minus x prime a transpose x minus x prime. So, this is this much, so now if we put our Euclidean distance definition ((Refer Time: 29:06)) in to the integral of equation 1. Supposing we call this as equation 1, so in light of this the equation 1 may be rewritten as, the D equal to half of integral minus infinity to plus infinity $d x$ vector $f x x$, I just dot this $d x$ earlier that there is no problem, x minus x prime whole square. So, this is our equation number 2.

Now, what we have to do is that we have to fulfill to minimize this D we have to fulfill two conditions. The first condition is that, given the input vector x vector, given the input of x vector, given x vector choose the encoder that is, c is equal to $c x$ to minimize simply, the Euclidean distance between x and x prime c . So, that is the first requirement, in fact, this is called as the nearest neighbor encoding, in fact, what we are essentially doing you can see that by mapping from, the input space continuous input space to the discrete space, what we are doing is that we are only making the nearest neighbor as the winner.

So, it is there that it is moving, so it is a nearest neighbor encoding, so this is the condition 1. And now, the condition 2 that one has to fulfill is the decoder condition, so which means, to say that given c compute, the compute x prime is equal to x prime c all

in vector notations. As the centroid of the input vectors that consists that satisfies this condition 1, as the centroid of x vector centroid of x vector fulfilling condition 1.

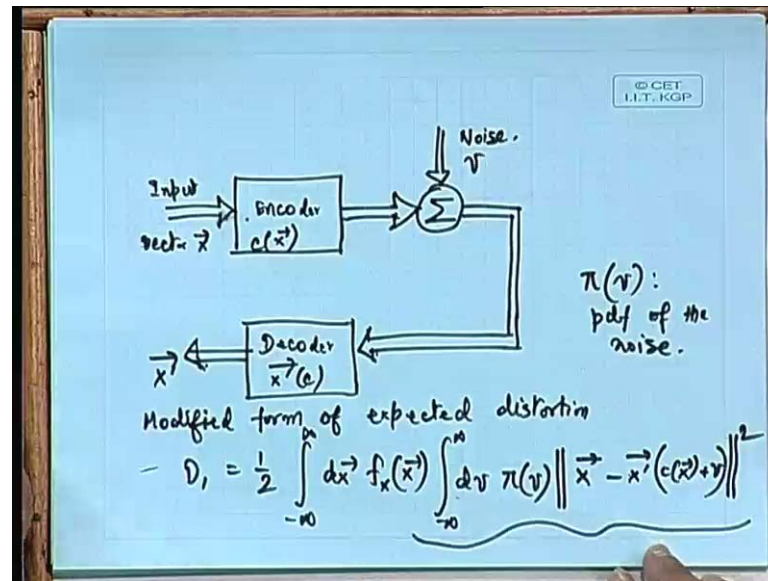
I will just explain what by now, what by this line effectively you see that from the x we are encoding in to the discrete space the diagram that, we had shown here let us go back to this, ((Refer Time: 32:01)) and from here, it gets mapped back to x prime. Now, what happens is that we can pick up another vector, which is close to this x vector and that, may also get mapped by the same another vector very close, in the very close neighborhood this also get mapped by this.

So, what we are trying to say is that whatever, we feed ultimately it will indicate that the this vector will be positioned. This lattice position will be the centroid of all this x vectors, which are causing this to be the winner. So, that is like the condition 2, if you try to imagine in the way of the self organizing map. Now, this aspect actually is called as a vector quantization, this whole thing that is from the continuous space to the discrete space by dimensionality reduction, is the type of the concept, which is followed in vector quantization.

Where, we can have a large collection of regions, but those regions every region will be encoded by one code given in the code book. So, that is the vector quantization concept and, the self organizing map is very much I am unable to this. In fact, just to show the capability of the self organizing map and is and more. So, because we have introduce not only the winning neuron, but also the neuron surrounding to this for the topological ordering.

So, that is why this noise free model of the encoder decoder will not be of much use to us. What we have to do is to, introduce the noise component also. Just to model the self organizing map in to this vector quantization. So, what we do there is that we slightly change the in the encoder decoder model and try to introduce a noise.

(Refer Slide Time: 34:10)



So, let us say that we have the input vector \vec{x} vector and then, we have the encoder, so this is the encoder c of \vec{x} . And then, we have the code coming out of this and then, we have got an additive noise. So, supposing this is the noise let say, \vec{v} and then we are just this noise term add it together will come to the decoder, so this is the channel effect. And then, comes the decoder although this is an analogy to the communication mind you and, we will be talking specifically about the analogies little later on, the encoder decoder model analogy with self organizing map.

And here, we will be having the reconstructed vector that is \vec{x}' vector. So, the modified form of the expected distortion, so the expected distortion that, I had already written down now gets modified. So, how it gets modified simply, that we were earlier writing it you see here, ((Refer Time: 35:32)) we were writing it as \vec{x} minus \vec{x}' and, what was \vec{x}' was nothing but, c of \vec{x} .

So, it was a \vec{x} minus c of \vec{x} vector the coded and the here, what we are going to do is that we have going to just put it is, no, it was actually, \vec{x}' of c of \vec{x} was there. Now, instead of writing it as \vec{x}' of c of \vec{x} we are going to write it as, \vec{x}' of c of \vec{x} plus \vec{v} the noise talk. So, the modified form of expected distortion will be D_1 equal to half of integral minus infinity to plus infinity $d\vec{x} f_x(\vec{x})$. And then, we have the star coming in you will be able to follow it soon.

D is the distortion, ν is the noise, and $p(\nu)$ is the probability density function of the noise. So, what we are doing is that we are first integrating this in to the space, so this whole thing will be the distortion measure. The distortion measure for x vector will be, what this entire integral expression and, what is that expression this term, we have written. And then, it is x minus x' of the argument of x' is not just $c \cdot x$ now, the argument of x' is $c \cdot x$ plus ν .

So, this is $c \cdot x$ plus ν and we have to take the Euclidean norm of this, so this is square of this and $p(\nu)$ this is the PDF of the noise. So, this integral expression is the is now, the error that we are is the distortion that, we are having for the mapping of x vector in to x' . And we have to just multiplied with $f(x)$ of x that is, the PDF of x with the x vector and integrate it over the entire input space, so this is the modified form of the PDF.

And then, what we have to do is that in order to that given this model, we have to find out the optimum encoder and the optimum decoder. Now, to find out the optimum encoder what we have to do is to, differentiate this D_1 expression with respect to c . Now, that we have got a term involving c , we can find the optimum decoder by finding.

(Refer Slide Time: 38:36)

The image shows a blueboard with handwritten mathematical derivations. At the top right, there is a small logo for 'CET I.I.T. KGP'. The text 'Optimum encoder' is written above the first equation. The equation is:
$$\frac{\partial D_1}{\partial c} = \frac{1}{2} f_x(\vec{x}) \int_{-\infty}^{\infty} d\nu \, p(\nu) \frac{\partial}{\partial c} \left\| \vec{x} - \vec{x}'(c) \right\|^2$$
Below this, the text 'Optimum decoder' is written. The equation for the decoder is:
$$\frac{\partial D_1}{\partial \vec{x}'(c)} = - \int_{-\infty}^{\infty} d\vec{x}' \, f_x(\vec{x}) \, p(c - c(\vec{x})) (\vec{x} - \vec{x}'(c))$$
Below this equation, it says '= 0 and then solve for $\vec{x}'(c)$ '. To the right of the equations, there is a definition: $c = c(\vec{x}) + \nu$.

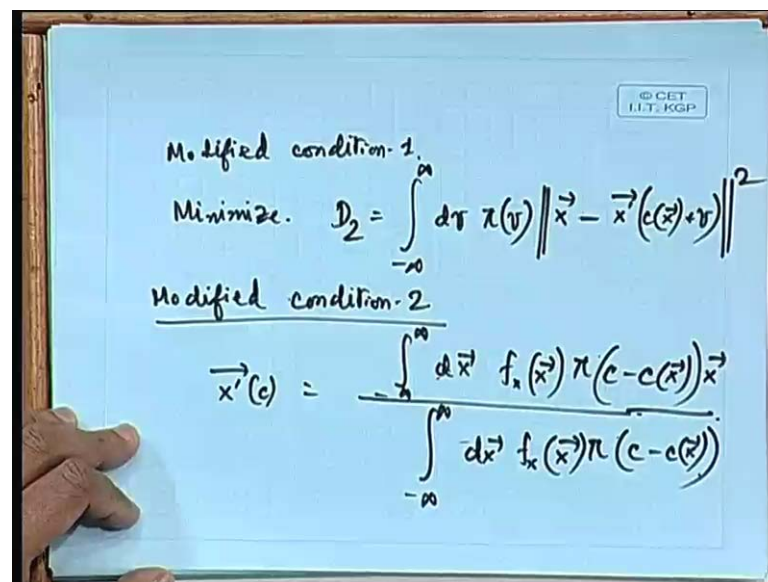
Optimum encoder by computing D_1 w.r.t c , c vector. And this, if you evaluate this it works out as, this is equal to half of $f_x(\vec{x})$ integral minus infinity to plus infinity $d\nu \, p(\nu) \, \| \vec{x} - \vec{x}'(c) \|^2$ whole

square, this is at the point c is equal to $c \cdot x + \nu$. So, this is for the optimum encoder we have to encode this thing to 0 and solve, and for the optimum decoder what we have to solve is optimum decoder, what we have to solve is by equating D_1 to 0.

So, D_1 expression is, again evaluated from this integral expression itself, is equal to minus of integral minus infinity to plus infinity $d x f(x)$ of x pi of c minus $c \cdot x$ vector. This, multiplied by x minus x prime c . So, what we have to do is to set this thing to 0 and then, solve for x c . So, we have to first solve for the encoder and then, we have to solve for the decoder.

Now, the condition that we had put forward earlier that is to say, in the noise free encoder decoder model, we had given the condition 1 and condition 2. Condition 1 was the nearest neighbor condition and condition 2 was the reconstruction vector condition, where we had said that it will be map to the centroid. Now, in this case with the noise model the condition 1 and condition 2 slightly gets modified.

(Refer Slide Time: 41:06)



Handwritten notes on a blue sheet of paper showing the modified conditions for a noisy encoder-decoder model. The text is written in black ink. In the top right corner, there is a small logo that reads "© CET I.I.T. KGP".

Modified condition-1.
Minimize. $D_2 = \int_{-\infty}^{\infty} d\mathbf{r} \pi(\mathbf{r}) \left\| \mathbf{x} - \mathbf{x}'(c(\mathbf{r}) + \mathbf{r}) \right\|^2$

Modified condition-2
$$\mathbf{x}'(c) = \frac{\int_{-\infty}^{\infty} d\mathbf{x} f_x(\mathbf{x}) \pi(c - c(\mathbf{x})) \mathbf{x}}{\int_{-\infty}^{\infty} d\mathbf{x} f_x(\mathbf{x}) \pi(c - c(\mathbf{x}))}$$

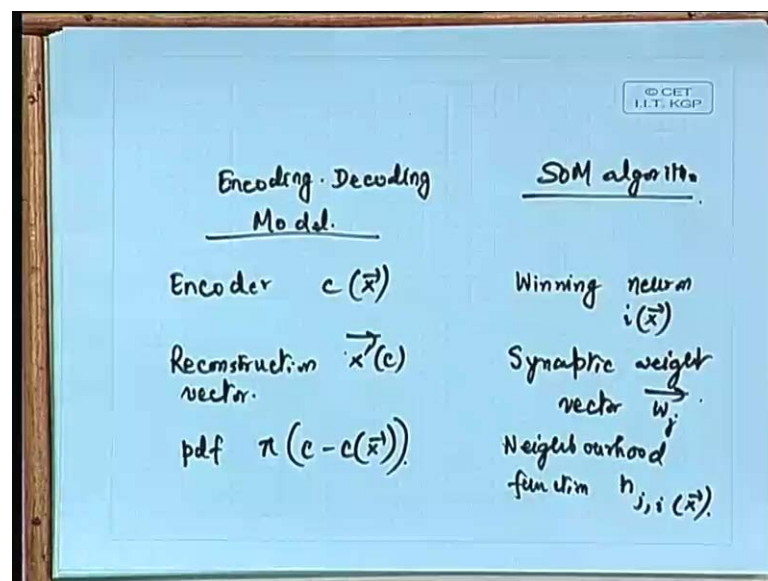
And, what are they the modified condition 1, that works out as is to minimize, again we define some distortion measure D_2 and what, is the distortion measure D_2 that is nothing but, you can see it is this ((Refer Time: 41:26)). This we have to modify, so this minimization of D_2 is given by, that D_2 is defined as minus infinity to plus infinity $d \nu$

π of \mathbf{x} and the Euclidean distance $\|\mathbf{x} - \mathbf{x}'\|$, whose argument will be \mathbf{x} plus \mathbf{u} . So, $\mathbf{x} - \mathbf{x}'$ with this argument squared norm of this, so that becomes our D^2 .

And the modified condition 2 will be, obtained by solving this term that is to say the optimum decoder term, ((Refer Time: 42:14)) that is setting this D^2 to 0. And then, solving for \mathbf{x}' and there, the solution of this \mathbf{x}' is given by, integral minus infinity to plus infinity $d\mathbf{x} f(\mathbf{x}) \pi(\mathbf{x} - \mathbf{x}')$ and in the denominator, it will be integral minus infinity to plus infinity $d\mathbf{x} f(\mathbf{x}) \pi(\mathbf{x} - \mathbf{x}')$, not this \mathbf{x} term.

So, this is the \mathbf{x}' of \mathbf{x} , which is essentially the equivalent form of the centroid this is basically the centroid that, you will be getting only thing is that this is under the noise term. So, this model actually is very much analogous to what we have defined as the self organizing map algorithm.

(Refer Slide Time: 43:33)



Let us see, that let us try to present this analogy, so we have the encoding decoding module listed on this side and the SOM algorithm listed on this side. Now, if we talk about the encoder part that is, c of \mathbf{x} in the encoding decoding model, what will be its equivalent analogy in the self organizing map, what will be the analogy of the encoder, do not have to look at the equations. Just again go back to our concept of discrete space approximation ((Refer Time: 44:19)) yes, that is i of \mathbf{x} , i of \mathbf{x} is the equivalent of the encoding process.

So, what is encoder c of x for the encoding decoding model, it is winning neuron i of x for the self organizing map. Now, what is known as the reconstruction vector, in the encoding decoding model, we have shown the reconstruction vector x' of c . And what, is it's equivalent analogy in self organizing map, the answer should come from you. Yes,

Student: ((Refer Time: Refer Time: 44:54))

Weight vector, that is correct the synaptic weight vector, because the synaptic weight vector is ultimately approximating the input vector very correct, thank you. So, this is the synaptic weight vector W_j and, you see that I have also got in the encoding decoding model, where I had shown the noise term also I have got the probability density function p_i of c minus $c \cdot x$ vector I had shown that. And what, is it's equivalent in the SOM algorithm.

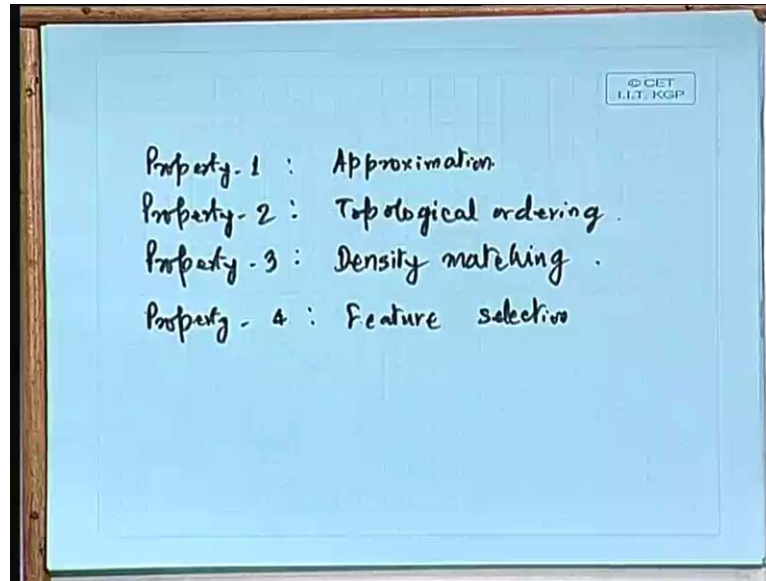
Student: Neighborhood function.

Neighborhood function, that is correct, because you see this c minus $c \cdot x$ is something, where we have got the noise built in. Noise means, equivalent to that there is a neighborhood function, that not only the winning neuron, but we are also exciting the neighborhood of this. So, the equivalent in self organizing map, is the neighborhood function h of j $i \cdot x$ vector. Now, actually this self organism, so these two these, three analogies are highly interesting, in fact that is what makes the self organizing map as a very good candidate for vector quantization, which one can use in the data compression applications.

In fact, we could talk about some of the practical applications of SOM details, but because of, the time constraint such discussions we could not have. Now, we are going to also, now I am also going to tell you about a few important properties of the self organizing map. Now, two important properties, which we have already seen in is number 1 is topological ordering, which we have already seen and understood also and the 2nd property is the approximation property.

So, these two properties can be accepted, in fact the approximation property itself leads to the vector quantization concept.

(Refer Slide Time: 47:41)



And there are two more properties, for the self organizing map. So, if we call the let us list out the four properties, if we call the property 1 as the approximation, which we have just now describe. And if we call as property 2, the topological ordering which also, we discussed. And if as property 3 and then, as property 3 we can mention about the density matching. Now, just we explain what density matching really means, let us go back again to this diagram.

((Refer Time: 48:33)) You see here, what I intended to do initially was to make this $x_1 \times x_2$ vector distributed in a uniform way, we wanted to have an uniform distribution of $x_1 \times x_2$ vector in this $x_1 \times x_2$ space but we could not make it uniform. Perhaps what might happen, is that supposing I get more concentration of vectors over here and, I get less concentration of vector over here. So, what can you expect out of the topological mapping.

Student: ((Refer Time: 49:17))

It will shift towards that and not only that, because there are large number of inputs over here, it will try to pull in more number of neurons. So, there will be more number of winning neurons there. So, more of the neurons will try to accumulate there that means, to say that even for the neurons also, when we find out the topological arrangement we will be finding that, the ultimate neuron positions will be with more resolution in this area as compare to the neurons, which are located in the parts regions.

Perhaps in this region only one neuron will be sufficient to resolve, but whereas here may be 4 neurons will required to do it. So, essentially again the topological arrangement, that we are having there is not much of qualitative difference between the property 1 and property 3, but the property 3 tells that the arrangement of neurons that we will be having at the end, at the end of the convergence phase.

Will be a such that the density of the neurons in this space, ultimately in the weight space ultimately, we will be having the density of that matched to the density of the input vector itself. Whatever, characteristics the input vector space is showing, will be ultimately shown by the weight vector that is, the concept. If you have understood that you have understood the self organizing map properly, so that is the density matching part of it.

And then, the last property of it is the features selection. That basically is also something not very new, now we have got an input space with non-linear distribution is not essentially it is a non-linear distribution. And using, an even from that we are able to pick up the best feature, out of a non-linear distribution of input space, so this is also a very important property.

In fact, it logically follows from the other properties itself. So, this is what we have for the self organizing map a very useful thing and we have studied actually, in the light of Kohonen self-organizing model, where we have presented in a more general way. So, that is all for this class.

Thank you very much.