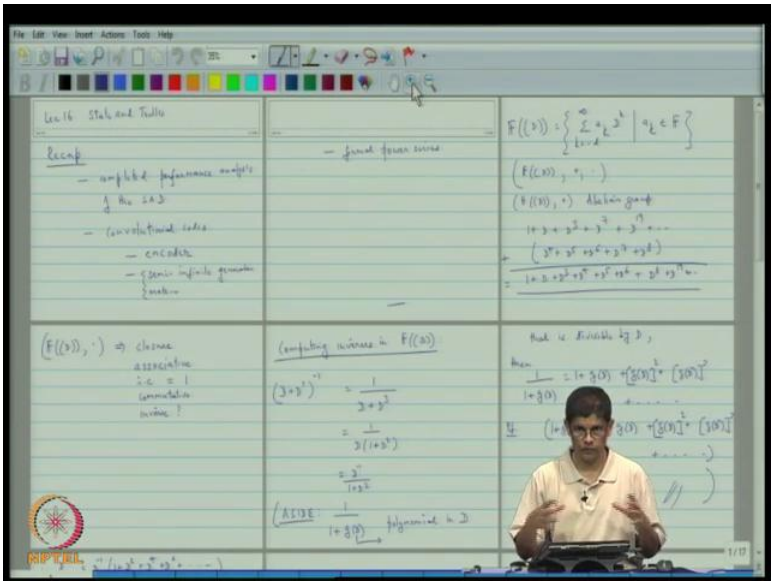
Error Correcting Codes
Dr. P. Vijay Kumar
Electrical Communication Engineering
Indian Institute of Science, Bangalore

Lecture No. # 17
The Viterbi Decoder

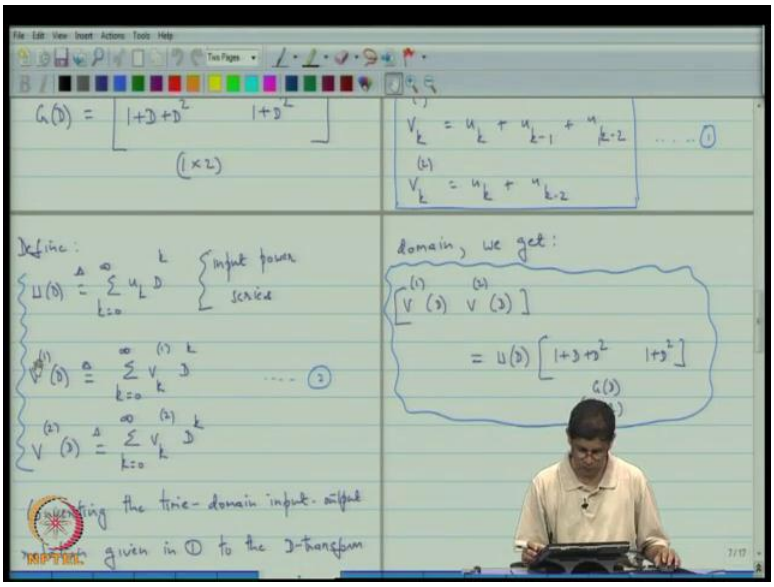
Good afternoon, now today is our lecture 17 nr series, and we will continue our discussion of convolutional codes. Let me just gets stated with so this will then be lecture 17, the Viterbi Decoder. And perhaps we should just recap our last lecture here, and let me zoom out and increase the zoom a little bit. So in the last lecture, we were concentrating on this state and trellis.

(Refer Slide Time: 01:27)



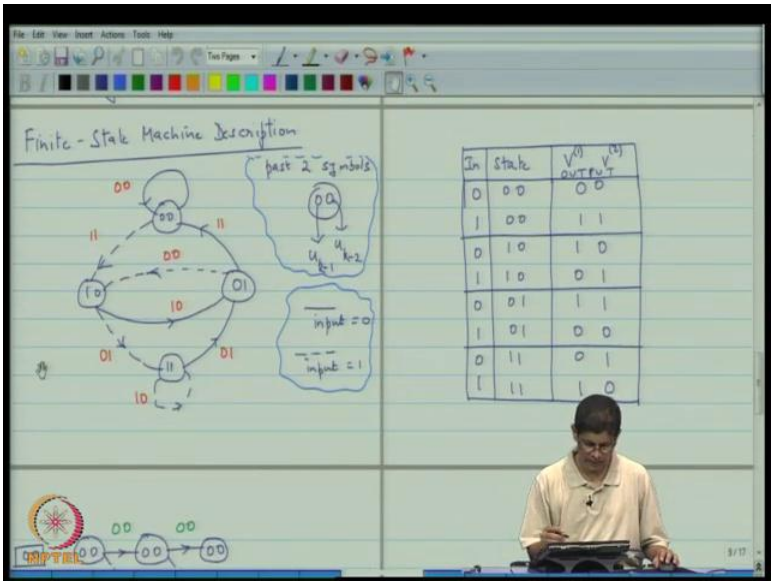
While discussing convolution codes, we are looking at various representations of the codes, and very useful representation is that of trellis; and origin of trellis lie in the state diagram, so that was theme of the last lecture. We started out by first completely, early of discussion of some series, and polynomials and if you will recall, we had given polynomial description for encoder here. If you look here, you will see that what we have on the left is the polynomial description let me just make sure that I can see two pages here.

(Refer Slide Time: 02:34)



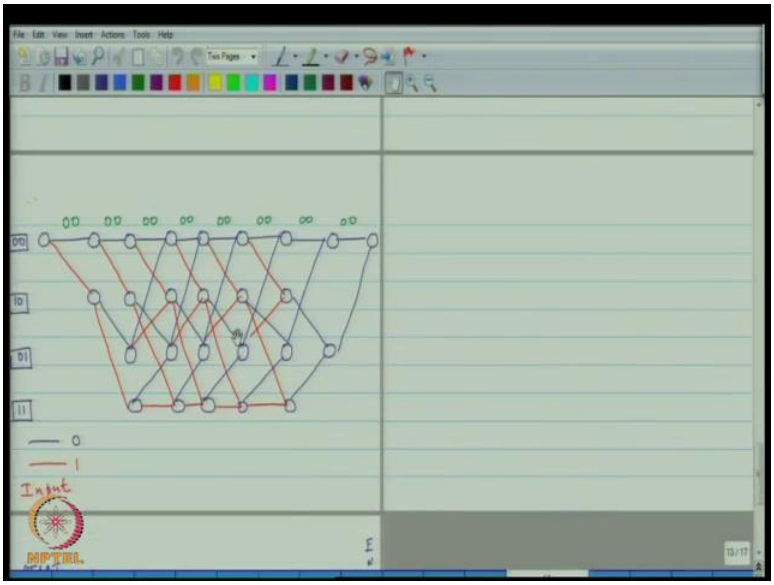
So, on the left what we have is the polynomial description of the encoder; and this is be called, polynomial generator matrix. And the input, output relationship is in terms of power series. So, there is the power series that represents the input, and there are two power series that represents the output, because in this case, the convolutional code has one input and two outputs.

(Refer Slide Time: 02:50)



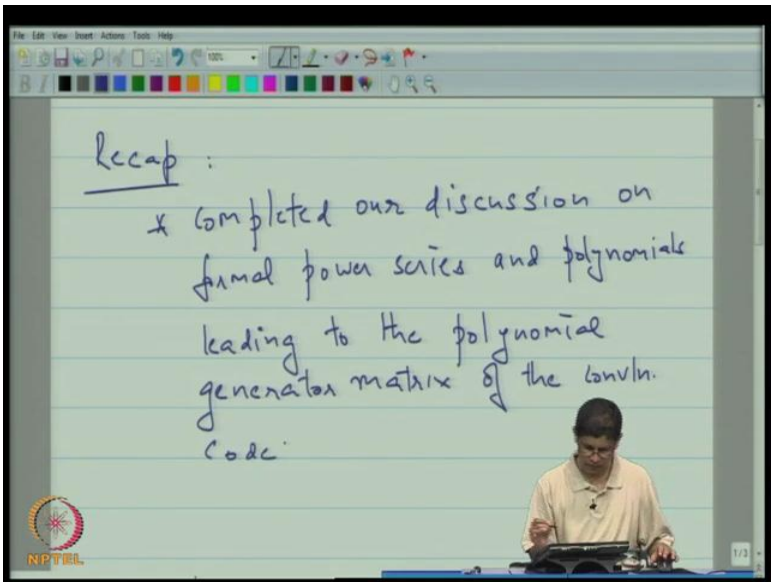
And then we drew the finite state machine diagram, for the convolution code, which directly relates to the state diagram that actually see you over here. We looked at the finite state machine, and this is the table that tells you that if you are in a given state, in have given input, what is the output? We have labeled the diagram according to that; they started drawing the trellis is to give the idea.

(Refer Slide Time: 03:07)



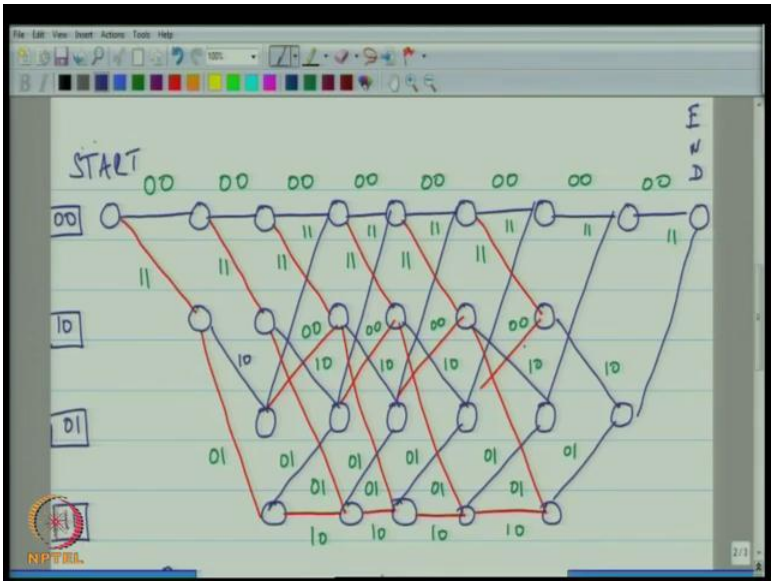
Here is the completed trellis and if actually have two version of trellis; one version kept the idea of the being that doing I wanted an unlabeled version, and then here is the label version, which has the output? So, let me just make this a truly unlabeled version, by removing these labels on top here. We have an unlabeled version, and we have labeled version, and we are going to actually use both. So let me the way will actually begin today is by copying, we will begin by copying the particular page, going to select the page and will copy it, and let us paste it in our new note. So it is not paste it.

(Refer Slide Time: 04:27)



So, I will write a very quick recap here; so the recap is that we completed our discussion on formal power series and polynomials, leading to the polynomial generator matrix of the convolutional code.

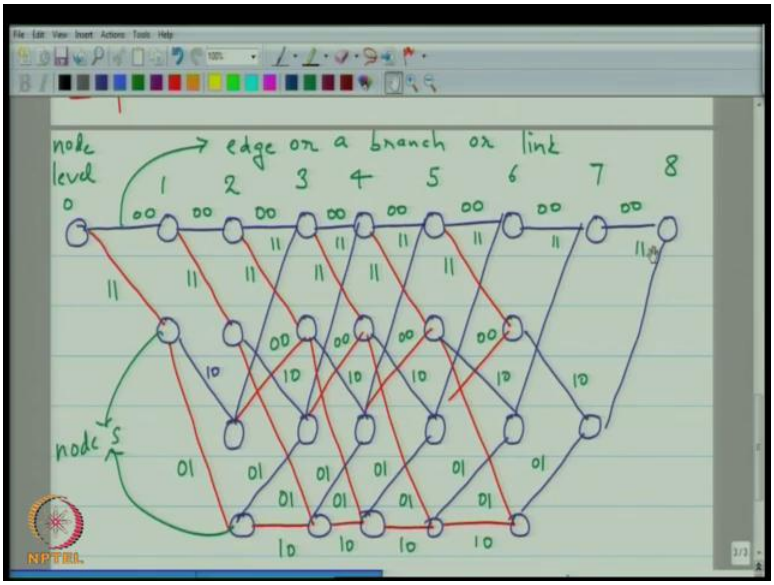
(Refer Slide Time: 06:27)



Then, we went on to talk about the state diagram of the encoder, viewed as the finite state machine, and then we started drawing the trellis diagram. And as I think I mentioned last time,

the trills is nothing but an unfolded state diagram. Basically, you are making the distinction between the same state at time t and time t plus 1. So, you keep track of time and state in this diagram, otherwise the connections are same. In this graph, there is some terminology associated with this; for instance, so let me do one thing here, just to introduce terminology, let me just pick out just the trills, if I could, if I copy the trellis over.

(Refer Slide Time: 07:08)



Here, I have the trellis and I am going to move it. Let us paste here labels and now, what I am going to do, thus the missing state is put that in. So, one terminology is that of nodes and so, every one of these is a node. So, let me use the different color for this, so these are nodes in the diagram. What you see vertically across time here, these are node level, so this is node level 0, 1, 2, 3, 4, 5, 6, 7, 8 and these corresponds to different instance of time.

So, this is node level 0, 1, 2, 3, 4, 5, 6, 7, 8. So, these are the different node levels. Now, then this is the graph, so these are nodes, it is very natural between two nodes are in edge. We will sometime also refer to an edge, has a branch. So, we call this either an edge or branch; and now sometimes, we may even speak of a link. Now, in other thing in keep in mind is that, the way leading this color. The edge in graph is that, whenever the input is the 0, we have an indigo color label colored line, whenever the input is 1, it is right.

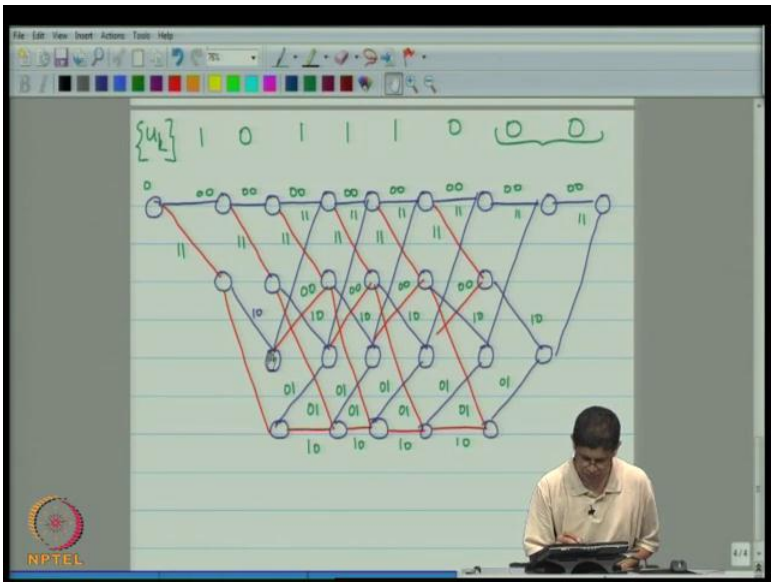
And I also pointed out that mean, so that means that if I feed the encoded the particular information sequences, for instance if I feed the encoded the all 0 sequence, then I will be tracing out this particular path in the trellis. On the other hand, if I feed the trellis all one sequence, message sequence, then I will be tracing this particular path in the trills. So, in this way suppose the alternative 0 and 1s, so that 0, that 1, that 0, and that 1, that 0, that 1, that 0, that 1. There is no one here, because there is the option one for at the end. I get idea. So, that means for every, because in the trills possibly first and last symbols to be 0. So, that was the reason, I could the get to an, but in general you get the idea that there is the 1 to 1 correspondence between message sequences and the task in the trellis.

In this layout now, you are understand that can associated with the path message sequence, but at the same time, and I can also associated with the every path, a code word, because in this trellis, I also label the branches of the trellis with the output of convolution encoder. And in this way, I can actually associate every path in the trellis, a code word as well. So let me just go back to this labeling that I had here let me just (( )) erase this so that 0 0 0 0.

Now, I have all my branches labeled with output as well, and again repeating the same argument, but this time, they focus on the output sequences. We will see that, if the input message sequence is all 0, is the output message sequence is all 0 as well, if the input message sequence is all 1, which means we follow all 1s except the last being 0s and corresponding output sequence we can trellis. So, this is 1 1 0 1 1 0 1 0 1 0 1 0 0 1 and 1 1.

So in this way every path of the trellis is connected with both message sequence, as well as code word. And path is basically sequence of edges or sequence of branches or sequences of links. There are same thing. You got the terminology out of the way. Now, what we would like to do? So I have entitled the lecture, the Viterbi Decoder, so I have focus on using this trellis to actually decode the code, when you send the certain message across the channel and certain and the other end of the channel you would seen certain trellis sequence. Now, what you are interested in doing, we trying to find and recover the underlying message sequence. So, we have to make choice here, so I thing by now and so let see how I want to do this. I thing the term, I am going to copy the trellis, one more time. Let us try that once more. So I have selected perhaps little more than the trellis, but that is ok. We paste it here; now let us get rid of some of the unwanted lines.

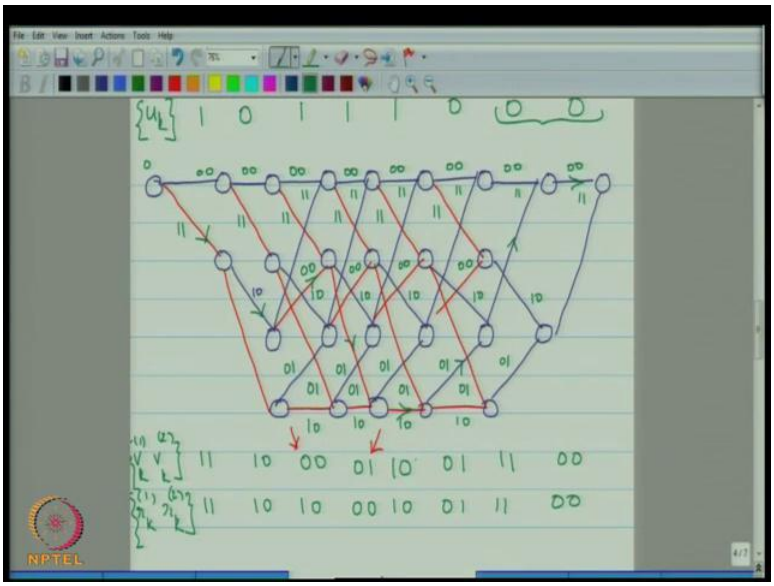
(Refer Slide Time: 13:38)



Let us say now here in a particular instance, well I want to illustrate how decoding works now, so let us assume, let me see if I can actually scaled trellis down in the little bit. So, it may that not require. The finite series, kill that, I thing that require. I want to put down, a particular input sequences here, so let say that, that input sequences and maybe I want to use and let me use green for the input sequences. So, let say that my sequence u k is 1 is 0 1 1 1 0 0 0. Now, at the last two 0 are first, because we want to bring the trellis back to the all 0 state; and reason for that is purely convenience factor. In practice, you do not need to do it; you can actually build the code, you not have to terminate the trellis, because it turns out that there is a certain loss, the small loss information rate will be terminated. On the plus side you do again a little bit performance, but the biggest reason for doing that is just for convenience in x position much easier to explain the convolutional code, when you actually terminated.

So we will assume that the it has been terminated by putting, is last two 0s over here trailing or terminating 0s. So, this is the message that you actually want to convey. Now, what path are you tracing along the trellis; your tracing, since you are going 1 0 1 1 1, 1 0 1 1 1 and then we have three 0 0 0. So, that is the path that you are actually tracing out. On the code, let me see, what I want to do. I think the wise thing for me to do right now is to make several copies of trellis because I want to represents different thing and different version of the trellis. So, let us do that.

(Refer Slide Time: 16:00)



Let us select the trellis and copy it of few times over. I got the trellis copy, the few times. I can form the trellis from 1, 2, 3, 4 and go back on this 1. Now, so then, but I want to do is I just illustrate the input sequences. Here, putting this, the rarer dark or just to dark. Let see how do we want to do it, maybe I will try, to put in arrow, along the paths. So, let us put this is 1, this is 0, this was 1 1; so let two 1s, three 1s, and have three 0s.

If you follow the arrows, let me zoom that little bit. If we follow the arrows now, so if you have 1 0 1 1 1 0 0 0, you actually, have the message sequence. But more than that, you also have the corresponding output sequences, which is given by again, let me use green hear. So, that will be v 1 k v 2 k. So, I have the output sequence here and I want to focus only on the particular message sequence that I have identified here, so the corresponding output will be therefore, the output that corresponds to this. So, I thing I have zoom out of there to right that; so if you look at top sequences and then, you look at the path then you will see that the output is here I have a 1 1 1 0 0 0 0 1 0 1 0 0 1 1 1 1 1. The last is the 0 0. Let me just check theoretically, (( )) on my nodes, so you see thus.

This message sequence, on top his associated with this code sequence here. Now, I want to deliberately introduce two errors; and I am going to deliberately introduce two errors, and then show you how will actually decode. Now, it turns out that this convolutional code can correct,

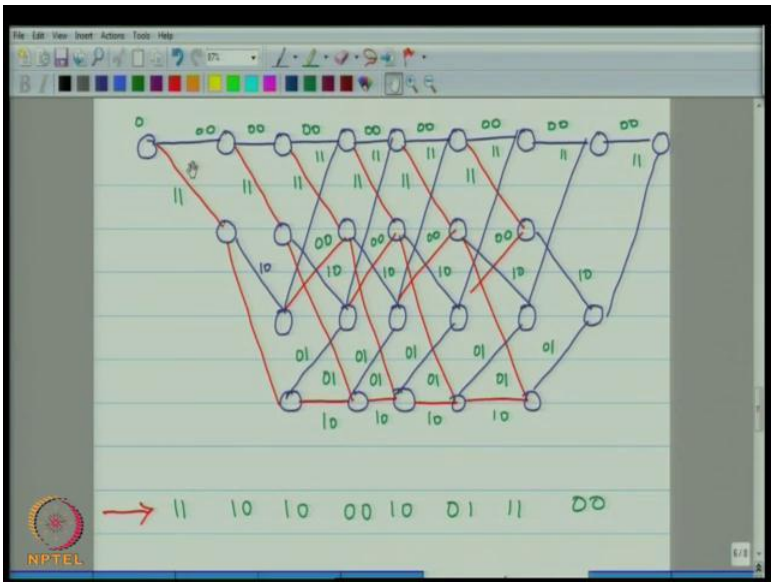
the error correction capabilities of code depends up the where the error occur. Since we are only dealing with the small segment of the code, is wise not to introduce too many errors.

So let us, let me introduce two errors, so normally if this is the class room setting, I would ask someone on the class to pick two of the symbols in (( )), since we cannot do that, let me do the picking. And I am just going to correct attributively, correct two of bits. So, then that will be r k 1, r k 2, these are the two. And I am going to the change, I will put a small arrow to indicate one that I am going to change; I am going to change this symbol and I am also going to change this symbol. It does not really matter, after pick something. So now (( )) sequence therefore, because of the error 1 1 1 0 1 0 0 0 1 0 0 1 1 1 0 0. Now, the question is how would actually decode the code?

So, for decoding, what you want do and this is the brute force, this decoding the code look an then, we talk about the decoding the black code, we said that the to decoding binary black code, if you were interesting in minimizing the probability of codeword error, what you would have to do maximum likelihood the decode error. If the channel was banal the symmetric channel then the maximum like decoder and find the code words equal likely, which is all most solve the case then, we choose minimum hamming distance decoder.

Here, that is the goal; we want to use the trills, and giving the particular receiving sequence for you want to do. We want identify the code word that, closest to the hamming distance to the particular receiving vector, so that is all goal and way it the easy, as way to do that to the trills it actually, going to every branch and write hamming distance between the output associated with the branch and the true output of the code word associated with branch and receive the distance. So, we can do that, then if we do that then, I thing smoothly copy this page over and we repeating this here.

(Refer Slide Time: 23:15)



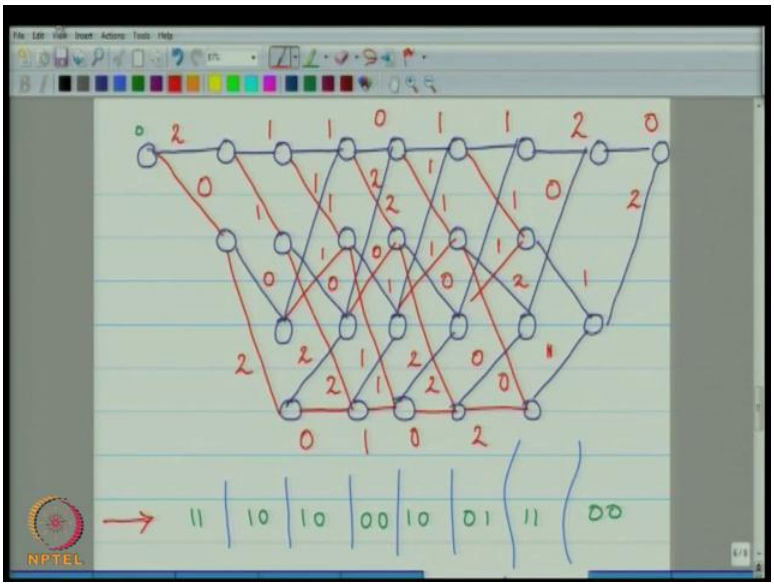
I also want select term let us what happen here, let me try that one more time, I want to copy this page here, this entire page, select page and now, new page and paste. The replace, that will check and time consuming, so let us want to that. So let me just paste the received sequence separately. We have that here. So in fact, let us move that little bit closer to the trellis? Here now, what I want to do, this I want to that trellis, I want to mark, I want focus on the receive sequence, I want focus on this, the mark on the trills in close the output sequence, I want to mark the hamming distance between the corresponding the received segment, and the code word. So I guess, I can delete this, the actual code word is not of interest now.

We are decoding; so like the receiver, the only thing that we have accessed to is the received vector. Now, will go ahead and put the branch, what are called the branch matrix, which is nothing, but the hamming distance? So, focus on the branch on here. The output is 0 0, but receiving that 1 1, which means the hamming distance between the output, the associated with the branch and receiving that the vector is true? So, likewise on this branch of 0 0, this is 1 0, and this is 1. So these differences are called branch matrix. Let us put down, for every branch in trellis, the corresponding branch matrix; and I feel at the easiest way today is to actually, go back. When denote the output and you can get the corresponding receive signal and then compute the hamming distance.

So, for that unlucky do that in red, so the branches basically, here is 2 1 1 0, so just little bit worried, but missing the elementary, just the symmetrical. I thing and this edge to the shift edges, little bit the right. So, let us move the trigger right impact actually draw. So, lines which help demark it. This here is this for the every session of the trellis and I am going to use the corresponding output. So, the basic node here so right know I am operating here. So, the branch matrix here is 1, this is 1 that is 2 that is 0.

Now, the freaking down that branch matrix, we can get these symbols. And similarly, we can do that the other branches, so let me do that with these branches here. So, the branch matrix here, because the distance between the all one branch, that is coming down and what is in near; that is 0 1 1 2 1 1. Let us delete this 1 1 symbols. Then the next is, let us focus on the branches, on there are going of output again 1 1. So, we have 1, 2, 3, 4, 5 of the branches, which take of symmetric start making, start up from third. So, this one could then the 1, this should be 2; this should be 1 1 and 0, interested in the branch, that is correct. That presents 0.

(Refer Slide Time: 31:00)



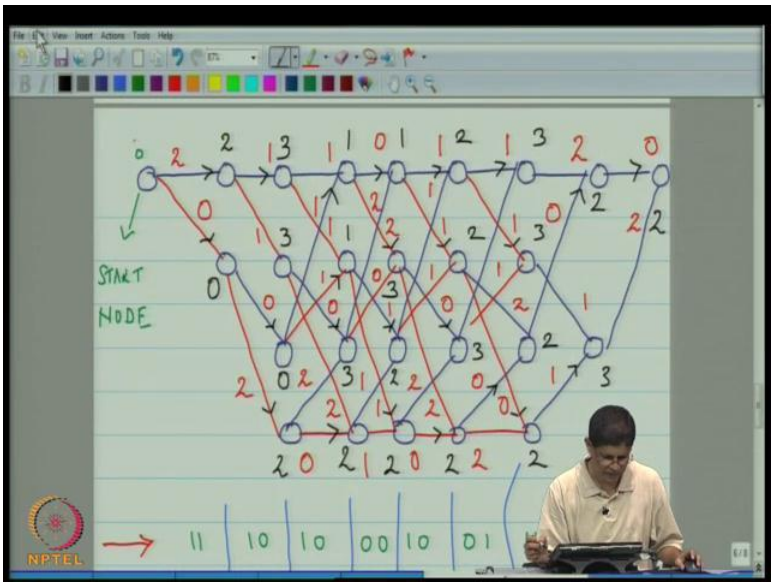
Let us remove the corresponding entries and this is little bit checking, giving that is source continue that and so, actually I got it right. So, now done it for the 1 2 3 classes of the branches. So, the 5 more to go, let me take care of the four at bottom, but the output is 1 0, but that looking at the true output receiving, the hamming distance is 0 1 0 and 2. Then we have this downward

branches here, and output is 0 1. So, let us put down the symmetric, for those branches. There output 0 1. So, the metric is 2 here, this is again 2, this is 1, this is 2 and this is 0. Now, take a look at for this branches here, which have label 0 1 1 2 3 4 5 of them and let put the branch metric.

This is the third segment 0 1, but it is actually 1 0, that hamming distance is 2; 2 here hamming distance is 1, that is 1 2 0 0. So hopefully we will (( )) to handle this; it is getting little bit crowded here. But the I guess, news the three the branches go, so, it is the branch, which is branch 1 0 1 0 1 0 1 0 1 0 1 2 3 4 5 6 fit. So, let us take care of it so here, that was 0 and here again it was 0. This form is 1, this form is 0 and this form is 2. So, now there is the one branch, which first I forget this is, 1 0, so let us put down hamming distance of 1, for that. We had only the left with the 0 0 branches. These are the branches that are going 1, 2, 3, 4, so the hamming distance is 1 0 1 1. Let us 1 0 11. I thing we have actually done it. So now manage to there is one, there I thing here, there is one output, hamming distance is 2, and here we did not get rid of these because I thing, 1 and 2 we are running to close together, let me it is fix that here.

So, we will get rid of that. Now label the entire trellis with the branch matrix and decoding. Now, is that matter actually go along path, and seeing, which path along less matrix. The groups for matters just look at the path in the trellis, and then say, what is the matrix each other. The matrix of the path is some of the matrix along branch, because after all we are computing the hamming distance between the code word and receiving that, so the symmetric along the branches. Now that involve computing, the matrix of every code word, which is too many code word and we know want to do that. The key and aspect of the Viterbi Decoder simply that you can actually, discards some parts as we go alone, and the illustrate.

(Refer Slide Time: 37:37)



So, here will go node level by node level. So this node level actually see that you have, there are two parts; one which has the branch matrix of 2 1, which branch matrix of 0. So at this point look like this, may have been, the more likely path taken by the code word. So, if you like, you can actually put the let choose the different color, to indicate branch matrix, and may be what should I choose let us choose yellow (( )), maybe I should use black. So this one is 2, and this one is 0. Now, in attaching branch matrix to node just so that I know that, the path looking that this node higher hamming distance than the path leading to this node.

We come over here, now 2 plus 1 is 3, so I will put down 3 here; 2 plus 1 is 3, 0 plus 0 is 0 and 0 plus 2 is 2. So this level see that the node matrix are 3 3 0 0 0 and 2, which mean that the path leading that node from start node, this is start node, so the path leading from here, from the start node is node has matrix 3, here it is 3, here it is 0, here it is 2 keep going. Now, here it is 4 and here it is 4 and 1 and 3 and 3. Now, we see the something interesting, that happen here, you say wait a minute, I cannot quite do that, because in reality to each node now, I have two, to each node now, I really have two incoming path, so which should I choose. So, how to make a choice? The answer is as you might expect you choose the more likely path. And this is where the rejection that I perceiving about contracting place.

So, now the two paths leading this node; one which has matrix has 3 plus 1 which is 4, and other which has 0 plus 1, which is actually 1. So, we will take the less of the 2 which happens to be 1. So, we will attach that to this; so this is 1. And let us just put an arrow, up here to indicate the that we made this selection. Now, this matrix, the branch matrix here, is 3 plus 1, which is 4 or 0 plus is 1, which is 1; so we will put down the lesser of the 2 is which is 1, and we will indicate that it came from here. Now, down here it see the 3 plus 0 or 2 plus 2 to 3 plus 3 0 ones and we will indicate that. And here it is 3 plus 2 or 2 plus 0, so the 2 plus 0 1 so, we will actually put down here and put down arrow here. So that means just be consistent, let us also put an arrow in the previous path, because although there was no selection to be meet there, that is only because there was only one incoming edge to each of the nodes.

Now, each node if you follow, if you trace back path along the path, you see that path that most likely given what you seen in thus far to be the code word that in terms the hamming distance. So, that these things; in the way decoding is about picking the winner; and winner is the code word that is closest in hamming distance to the received vector. What the Viterbi algorithm allows you do is it allows you to identify the looser that is code word, which can never be closest and through them away. So, we did the first time we actually did that was this node, because here we had a choice between this and a choice between this.

And since, in this case this one out this means that this lost, what is that mean? Because after all this is only a part of the code word, it means that any code word which begin in this manner, and continues with and however it continues, it always lost out to a path of this type because no matter what you can add as a suffix to this path you can also add it to this. So, this path has loss to this path as this node is lost forever, because beyond that, whatever happens is common to both of them and therefore, it cannot change the situation. So, relative comparison remains and so you have a looser here that since that can never be the closest code word. So, what remains re called the survival? So, each node has the survival, and the survival the matrix are 1 1 3 2 and also attach them to the node, but really there the matrix of the survival.

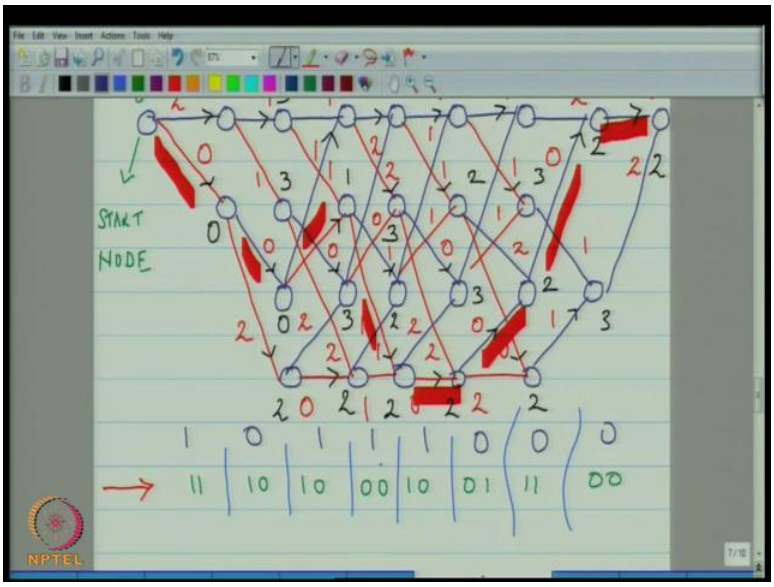
Let us continue here, the next node level we have a choice between 1 plus 0 or 3 plus 2. So, 1 plus 0 means, now we have the survival, you have rejected this path; 1 plus 2 or 3 plus 0. That is the attack. Now in case of (( )), you just arbitrarily flip a coin let us say that I prefer to take downward arrow always. So, it is 1 plus 2 and I will put down 3; and here it is 1 plus 1 or 2 plus

1, so that is 1 plus 1 and select this and here, it is 1 plus 1 or 2 plus 1, so prefer the 1 plus 1, so I pick this.

So again at this level, I have four survival, and closing back I can identify for instance if I am here the survival at that is this particular code word; if I am here it is this particular code word and so on. So, let us go about next and continue this. So, here it is 1 plus 1 or 2 plus 1; so that is 2 and we pick this. This is 1 plus 1or 2 plus 1, so that is 2 and we pick this; and this is 3 plus 0 or 2 plus 2. So that is 3 we pick this and here it is 3 plus 2 or 2 plus 0. So, we pick the 2 we pick this; again we have four survivals; and this keeps on going, so the keeps on going till the end, and we will see happen that.

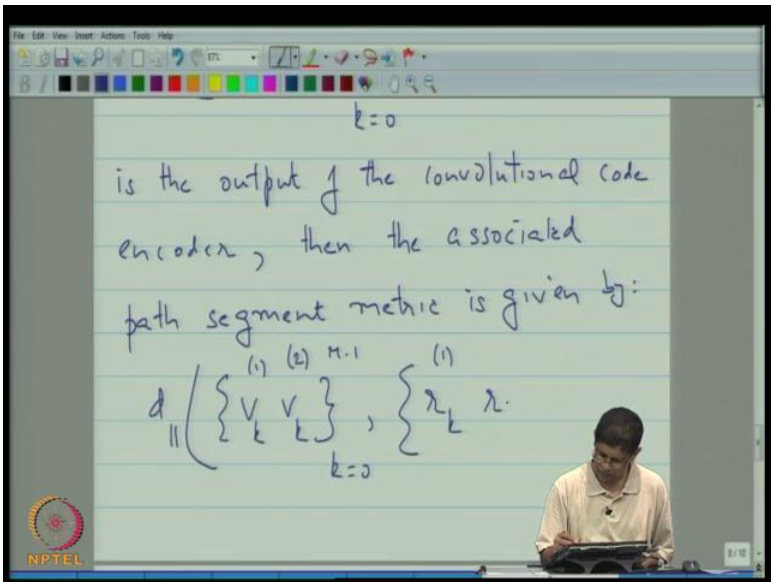
Now, continue here 2 plus 1 or 3 plus 1, so that is 3, we will pick this; this is 2 plus 1 or 3 plus 1 - 4. So, that is 3, we pick this, we see there 2 plus 2 or 2 plus 0. So that is this, and we pick this; and here you see that 2 plus 0 or 2 plus 2, so that is 2 we pick this. Over here you see the 3 plus 2 or 2 plus 0, so it is 2 plus 0. We take this 3 plus 1 - 4 or 2 plus, what was it, 2 plus 1, this is supposed to be 1. So, let us a 3 plus 1 or 2 plus 1. So, 3 plus 1 and we pick this. So now, it is 2 plus 0 or 3 plus 2, so the 2 plus 0 and your matrix is 2. So that means the path that is closest to the received vector that hamming distance is 2. Now, we finish, because you can trace back, you can trace back, so let me now copy this page again select page, copy page, insert page and paste page.

(Refer Slide Time: 46:39)



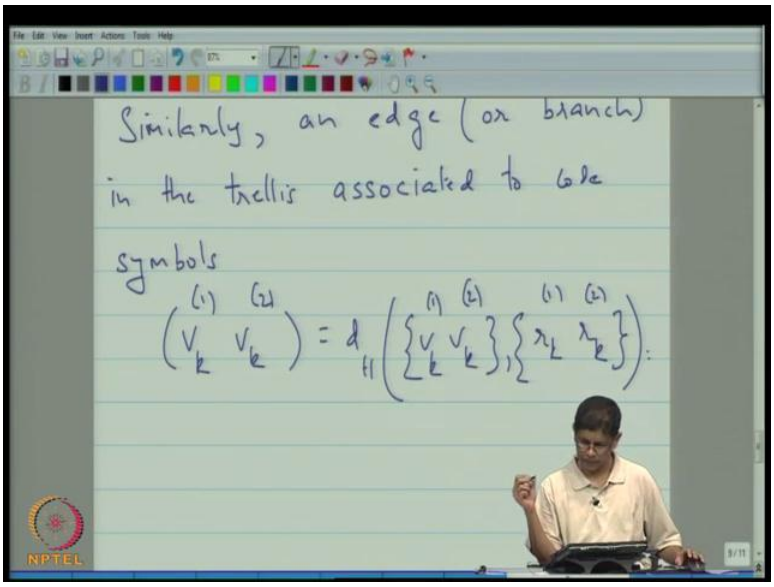
So, what that mean here is that now let me use this thick brush here identify, what exactly we decode it. We decode it, this path this one, because we were following back in here. So, because this survival one not so that is 1, that is 2, that is 3, so you want trace the arrows. That is 3, so that is 4, that is 5, that is 6, that is 7. So, what is you saying that, the winning path, do not like this and from back can actually, underline the message sequence and significant do that, so that underline the message sequence, we done the as 1 0 1 1 1 and 0 0 0. So, which 1 0 3 1s, three 0s, you go back look here, you find that is same 1 0 and 3 1s and 3 0s. So, we were decode the code and that much prizing if you know little bit, more ever the property of the code, two time create to error we can also able to decode in actual fact. We can actually decode node, we can decode large number of errors, but then, worry about the impulse errors. So, that is have decoding after 6.

(Refer Slide Time: 49:36)



So, maybe I should to this put down few notes, corresponding to some of the terminology and introduce here, if v k 1, v k 2, k goes to 0 to m minus 1, where m is less than or equal to n, is the output of the convolution encoder, then the associated metric. The associated, the path segment metric is given by the hamming weight between and the corresponding receiving sequence.

(Refer Slide Time: 51:09)



Similarly, an edge or branch in the trellis associated to code symbol is the hamming distance of… Then, so that these are just some notes to company discussion of the decoding. In the Viterbi Decoding process with each node at each node level, we associate a survivor, where the survivor, we mean, the path from start node, so I should just qualify these and say there by survivor at a node. So, let me do this, they by the survivor at a node, at an node we mean, the path from the start node to that node, having least path metric. In that should have the survivor itself.

So in closing, I guess the technology, had limitation, if you had the board, sometime discarding the encoding. The decoding process go little bit faster, so base this technology is little bit slower, but I guess there are something get faster using technology. So, I think it (( )). So, today we basically discuss, Viterbi Decoding algorithm, and I actually can we through example use a (( )) example trellis. So I showed you how the Viterbi Decoder reduces the complexity by being able to identify in some sense if you think of decoding as choosing winning path, it allows you to identify the user early enough, so that you do not spend lot of time loosing path in the quickly identify winning of the code word that closes to the receiving factor. The next time, we will continue on discuss a convolution codes.