Principles of Communication Systems - Part II

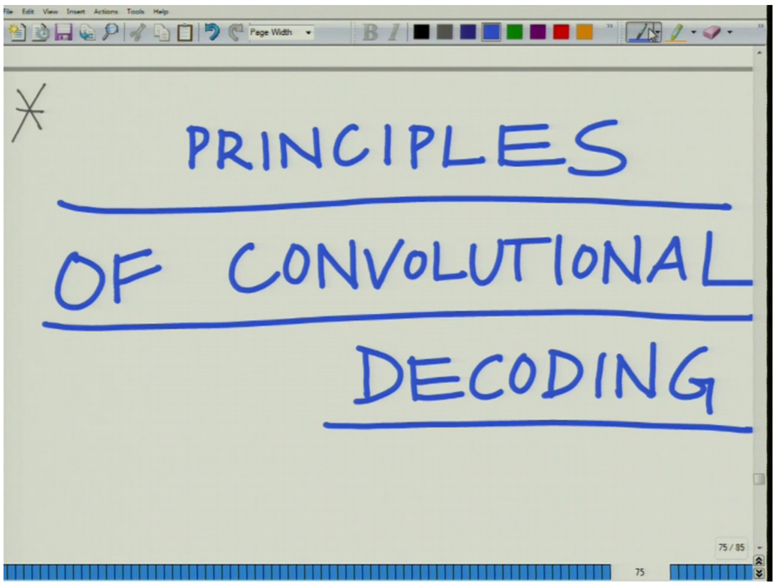
Prof. Aditya K. Jagannatham
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture – 55
Principles of Decoding of Convolutional Code

Hello, welcome to another module in this massive online course. So, we are looking at convolutional codes we are looked at the state diagram representation and the most important trellis representation. And we have also seen that decoding of convolutional code corresponds to finding a path through the convolutional, but through the trellis corresponding to the convolutional code, but what any path the path that corresponds to the path or the path that corresponds to the codeword that was it has the maximum likelihood, this is known as the maximum likelihood decoding principle which has the maximum likelihood of having been transmitter and of course, the corresponding information sequence.

So, towards that we would like to explore this decoding of convolutional codes. And this is going to be based on 2 central principles of fine. Remember we said the decoding of convolutional codes aims to find the codeword, if the minimum hamming distance or with the minimum distance metric path from the trellis curve minimum distance with respect to the received codeword and towards finding that codeword which has the minimum distance from the received codeword; we are going to be using 2 principles. And I would like to spend some time trying to explain these 2 principles to you in this module.

(Refer Slide Time: 01:33)

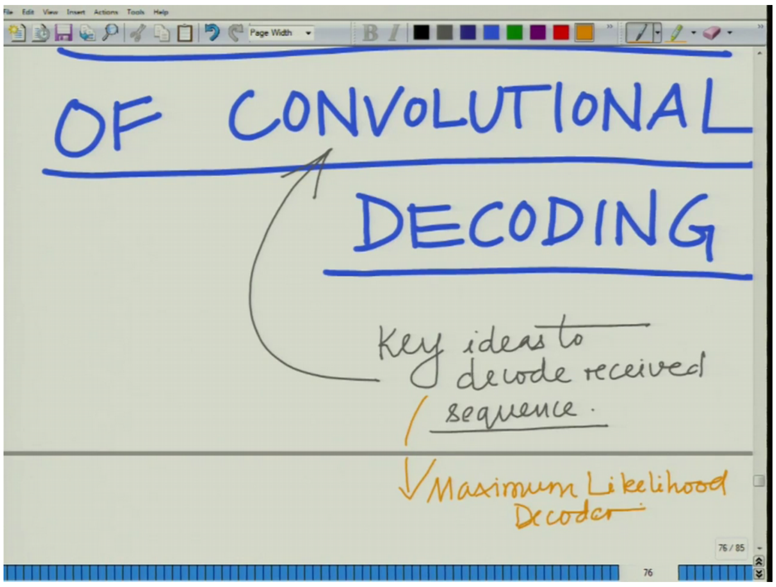

PRINCIPLES

OF CONVOLUTIONAL

DECODING

So, we have principles of so, these are principles of convolutional decoding. These are basically the principles

(Refer Slide Time: 02:24)


OF CONVOLUTIONAL

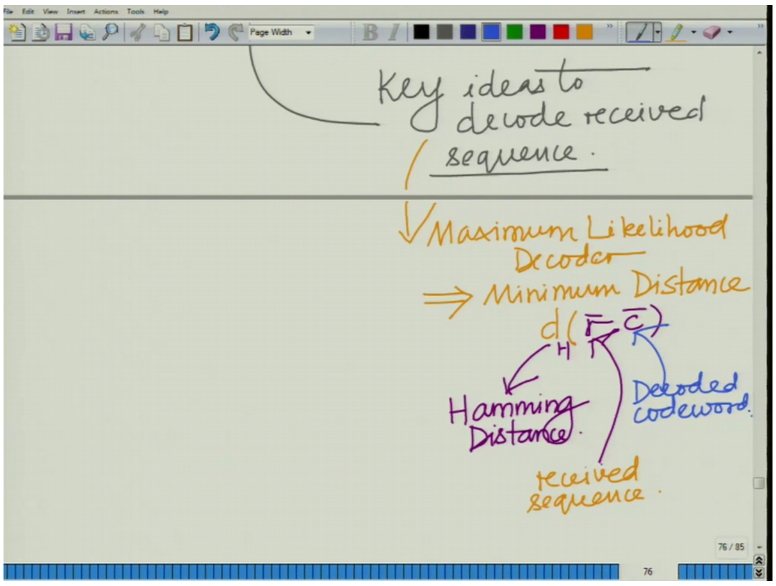
DECODING

Key ideas to decode received sequence.

✓Maximum Likelihood Decoder

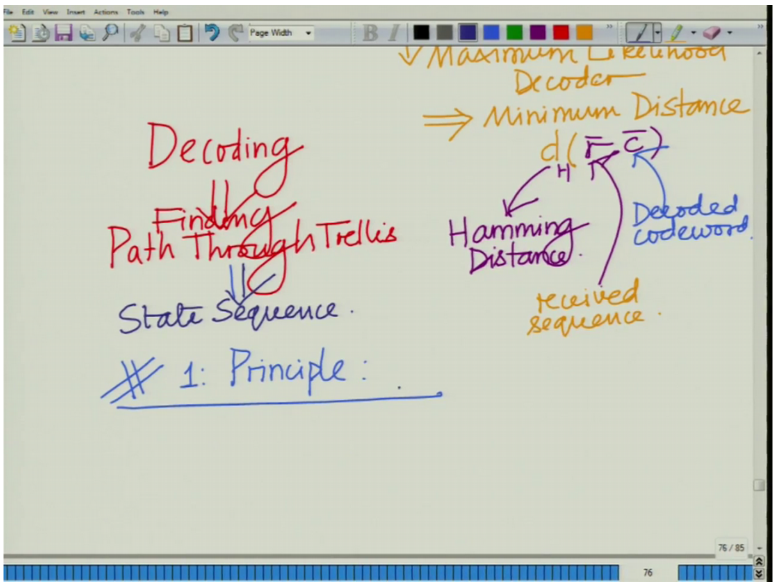
That we are going to, what are these? These are key ideas to decode the received, the received sequence or the received sequence. It might not be a codeword. So, we can simply say the received sequence. And in particular as we have already seen we are interested in the maximum, maximum likelihood decoder implies.

(Refer Slide Time: 03:19)



The minimum distance decoder. Minimum distance between r bar this is the hamming distance by distance r bar is the received word or the received sequence, you can say and c bar is the decoded code word, and c bar is the decoded keyword ok.

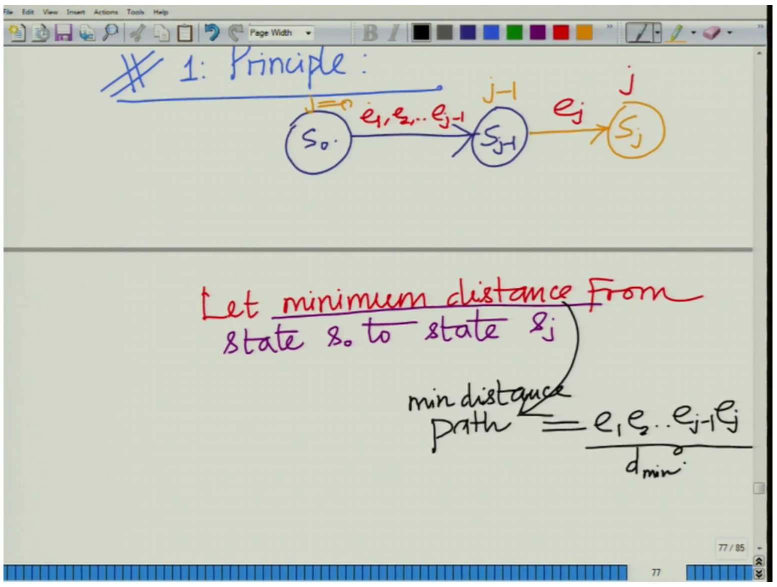
(Refer Slide Time: 04:15)



And the principle in the first principle that we are going to look at is the following; we can call it principle number 1. And the first principle is the follows now of course, what we have realized is basically the decoding, the decoding implies that since every code corresponds to decoding implies, finding a path through trellis path means, decoding means finding path through trellis in which implies basically, which this decoding implies finding a path through the trellis. So, that is what your decoder implies which means if I basically implies finding a state sequence and the transitions between the state the time evolution that is a transition this implies finding a state sequence that is what is the state sequence through which the decoder has progressed. Through which the encoder has processed right what is the state sequence in time that is what is the sequence of state what is the state at time 0 followed by state at time one state at time 2. So, we want to find the corresponding state sequence the codeword which is basically equivalent to finding the path which is nothing, but finding a state sequence through the trellis which you can be mapped into a information bit sequence.

So, all these are one and the same se we are interested in finding a certain state sequence corresponding to the received word. The codeword correspond to which has the minimum distance with respect to this we got ok.

(Refer Slide Time: 06:38)



1: Principle :

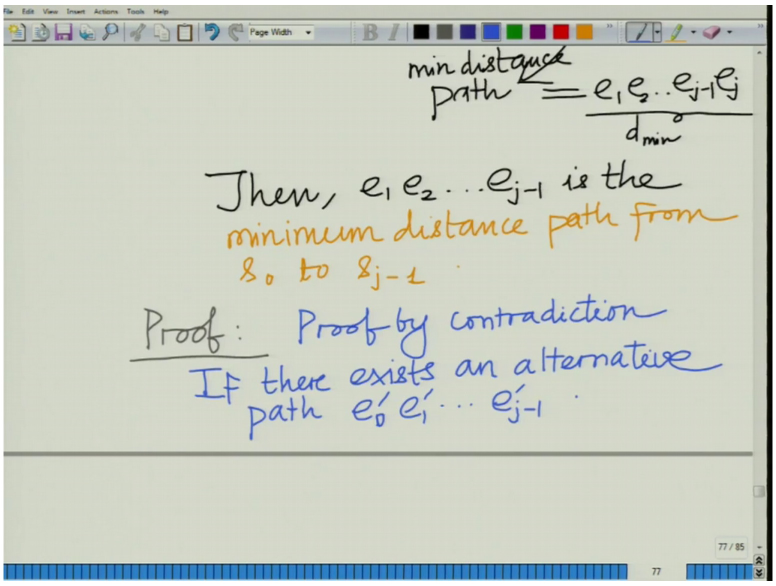
$$\begin{tikzcd} \overset{j=0}{S_0} \arrow[r, "e_1, e_2, .. e_{j-1}"] & \overset{j-1}{S_{j-1}} \arrow[r, "e_j"] & \overset{j}{S_j} \end{tikzcd}$$

Let minimum distance From state s_0 to state s_j

min distance path $= \underline{e_1 e_2 .. e_{j-1} e_j}$
d_{min}.

So, that is the basic idea if you try to understand it. Now if consider the state sequence let us say we start with state s 0 there can be several intermediate states and at some point we end up with state x j minus 1 at time j. So, this is at time 0 and this is at time j minus 1 and let us say there is a state s j at time j. Now the path the edges or the branches from s 0 to s minus 1 are given by e 1 e 2 e j minus 1 and the branch from s j minus 1 to s j corresponding to this minimum distance path is e j. So, what we say is let minimum distance, let minimum distance from state s 0 state s j. Let this corresponding path be minimum distance. So, the minimum distance, the minimum distance, minimum distance path is e 1 e 2 e j minus 1 e j. So, this is your minimum distance path it goes through a branches or edges e 1 e 2 e j minus 1 e j.

(Refer Slide Time: 08:31)



min distance path $= \underbrace{e_1 e_2 \ldots e_{j-1} e_j}_{d_{min}}$

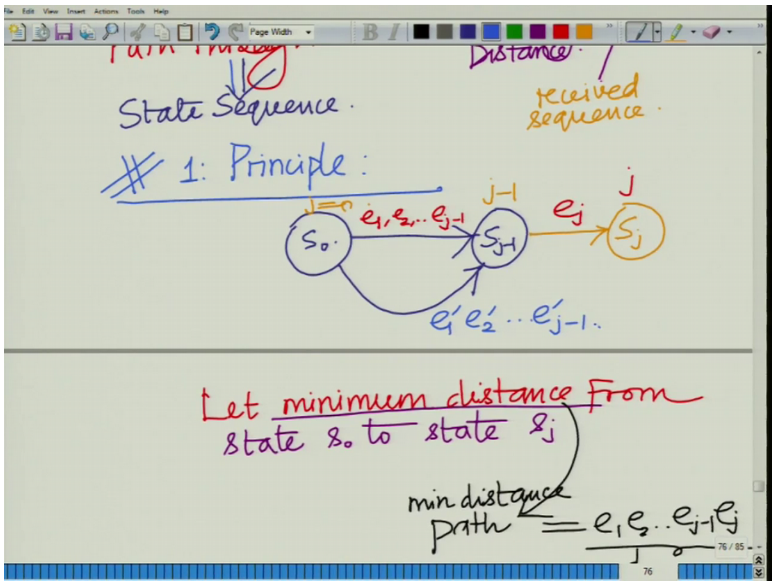
Then, $e_1 e_2 \ldots e_{j-1}$ is the minimum distance path from s_0 to s_{j-1}.

Proof: Proof by contradiction

If there exists an alternative path $e_0' e_1' \ldots e_{j-1}'$.

Then e 1 e 2 e j minus 1 is the minimum distance path from s naught to s j minus 1 this is an important point. It is very easy to see this that is if from s naught to s j right, when you go through state s j minus 1 e naught e 1 up to e j is the minimum distance path from s naught to s j. Then e the edges e naught e 1 up to e j minus 1 must in themselves can comprise of a minimum distance path from s naught to s j minus 1 as simple as that correct. Because and the proof is very simple proof is it is seems like a 1 line proof.

(Refer Slide Time: 09:55)


State Sequence.

Distance

received sequence.

1: Principle:

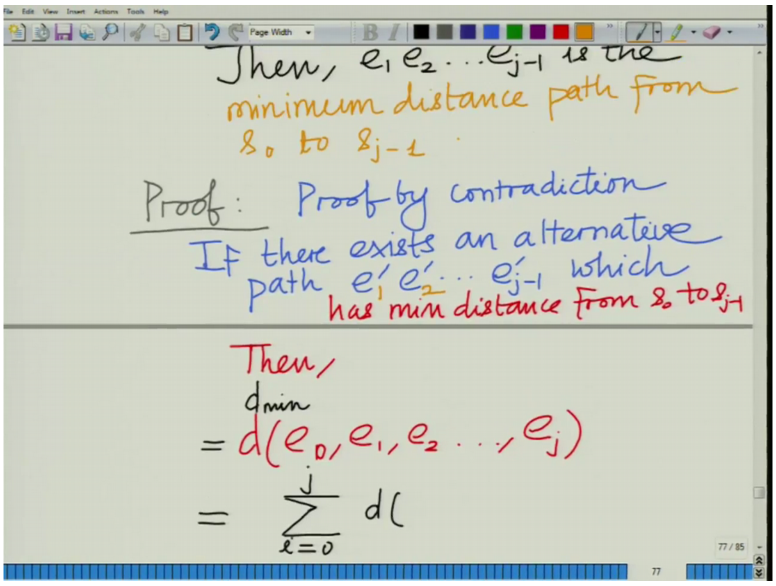
Let minimum distance from state s_0 to state s_j

min distance path $= e_1 e_2 .. e_{j-1} e_j$

The proof is very simple, if this is not a minimum distance path from s naught let there be another alternative minimum distance path that is, e 1 prime e 2 prime so on up to e prime j plus 1. So, we apply proof by contradiction.

The proof by contradiction is simply, if there exists an alternative path from alternative path e naught prime e prime 1 e prime j minus 1 which has minimum distance from s naught to s j minus 1, from s naught to s j minus 1.

(Refer Slide Time: 10:58)


Then, $e_1 e_2 ... e_{j-1}$ is the minimum distance path from s_0 to s_{j-1}.

Proof: Proof by contradiction

If there exists an alternative path $e_1' e_2' ... e_{j-1}'$ which has min distance from s_0 to s_{j-1}

Then,

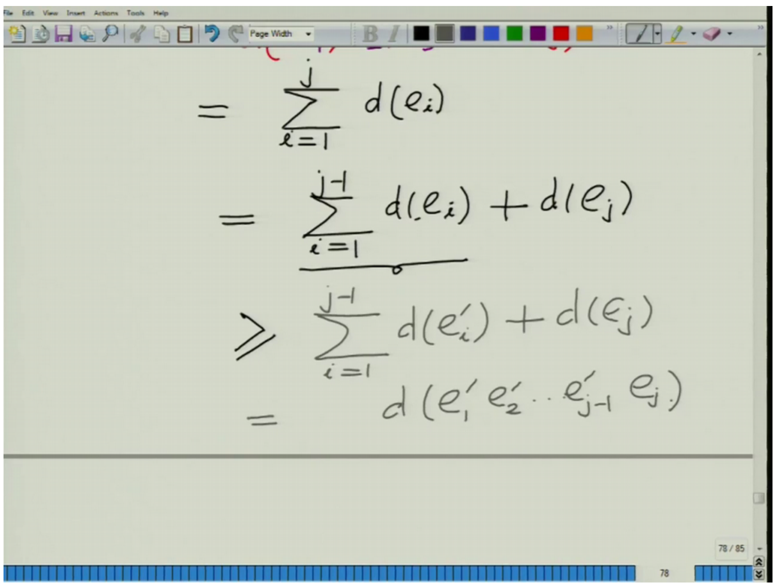
$$d_{min} = d(e_0, e_1, e_2 ..., e_j)$$

$$= \sum_{l=0}^{j} d($$

Then what will happen is if we look at the distance comprising of the path e 0 e 1 e 2 up to e j which is basically now the distance d min, correct? Which we have said is d min, d min equals the distance corresponding to, now because the distance matrix is additive I remember we have seen in the previous module the distance matrix is additive. So, I can write this as i equal to 0 to j d of well I can write this as i equal to 0 or basically let say we have e 1 e 2 I am sorry, we have e 1 e 2.

So, this is e 1 e 2 if there is another path e 1 e 2 e prime j minus 1, correct? Which has a minimum distance, correct? From s naught to s minus 1 then what is going to happen is, if you look at this is that the distance of e 1 e 2 e 3 is the distance matrix is additive. So, this is simply go into d distance of e i which I can write as summation i equal to 1 to j minus 1 d of e i plus the matrix the distance corresponding to the last branch e j.

(Refer Slide Time: 13:01)



$$= \sum_{i=1}^{j} d(e_i)$$

$$= \underline{\sum_{i=1}^{j-1} d(e_i)} + d(e_j)$$

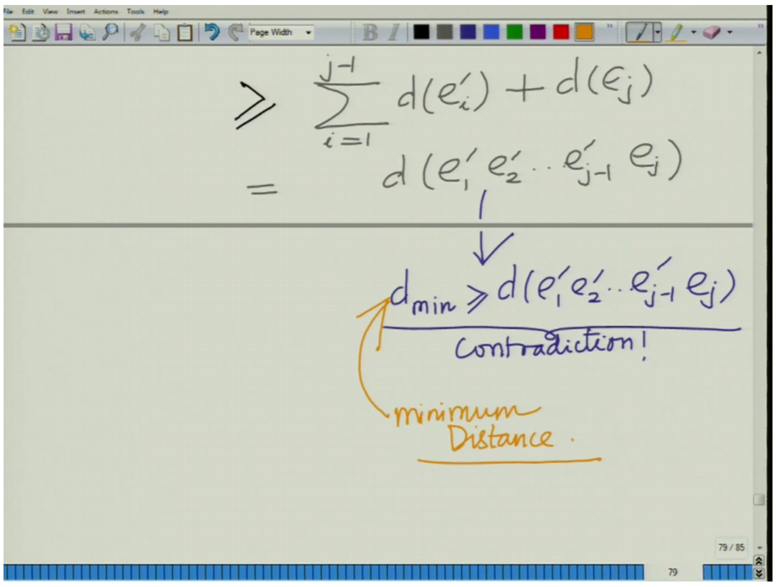
$$\geq \sum_{i=1}^{j-1} d(e_i') + d(e_j)$$

$$= d(e_1' e_2' \cdots e_{j-1}' e_j)$$

Now we know that this is the distance corresponding to branches e 1 e 2 up to e j minus 1 and by our assumption we have seen e 1 prime e 2 prime e j minus 1 prime is the minimum distance path. Therefore, this is greater than or equal to summation i equal to 1 2 j minus 1 d of e i prime plus d of e j which is nothing, but d of e 1 prime, e 2 prime, e prime j minus 1 e j.

So, d min is greater than or equal to d of e 1 prime e 2 prime which is a contradiction because that is a minimum distance path right.

(Refer Slide Time: 14:20)



$$\geq \sum_{i=1}^{j-1} d(e_i') + d(e_j)$$

$$= d(e_1' e_2' \ldots e_{j-1}' e_j)$$

$$d_{min} \geq d(e_1' e_2' \ldots e_{j-1}' e_j)$$

Contradiction!

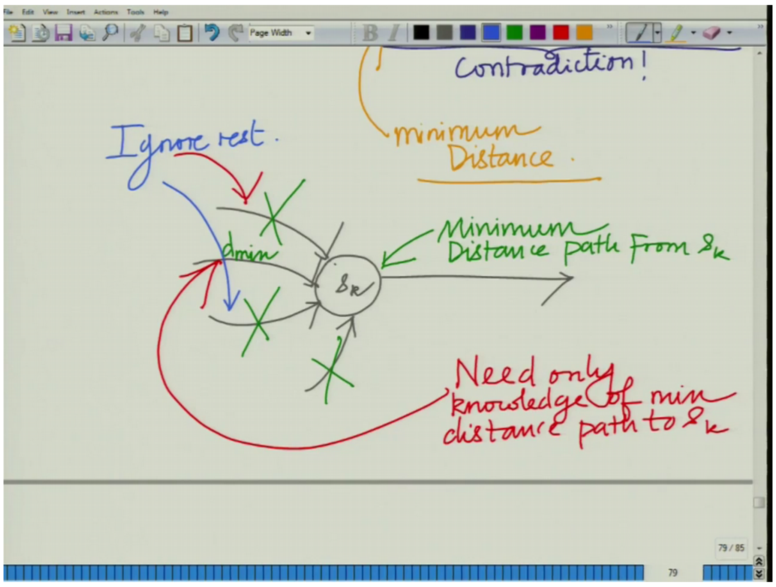
minimum Distance.

So, we are obtained a contradiction. So, this is a contradiction, this is a, this is a contradiction. Reason mean this is the minimum distance; this has to be the minimum distance. So, if there is another minimum distance path from s naught to s j minus 1 e 1 prime e 2 prime. So, on up to e prime j minus 1 then the minimum distance from s naught to s j through s j minus 1 will simply be that minimum distance path e 1 prime e 2 prime up to e prime j minus 1 and is followed by the branch e j all right. So, you cannot have a different sub path which is minimum distance from s naught to s j minus 1 it is very simple.

If there is if I have to go for instance I am from IIT Kanpur if I have to have to travel from Kanpur to Delhi and from Delhi to let say Bombay there is only one route then the path and I want to trans and I want to travel along the minimum distance path then the path I take from Kanpur to Delhi must be the minimum distance path as well. I cannot take a sub optimal path from Kanpur to Delhi and expect the overall path to remain minimum distance that is as simple as that. So, if there is an intermediate node the path to that intermediate node must be the minimum distance as well.

So, basically; that means, at every node I can only, I need to only store to find optimal minimum distance path at every node, I only need to store knowledge of the minimum distance path I can ignore. So, a node can have several paths leading up to it, but if I am interested in finding the minimum distance path. I only need to store knowledge of the minimum distance path. I do not need to store knowledge of any other path, correct? If I am interested in travelling from Delhi to other places and I am starting from Kanpur, I only need to know the knowledge of the minimum distance path from Kanpur to Delhi there might be several other paths from Kanpur to Delhi, but I do not need knowledge of any other those paths because I am interested only in the minimum distance path as simple as that. So, hence at every state.

(Refer Slide Time: 16:50)

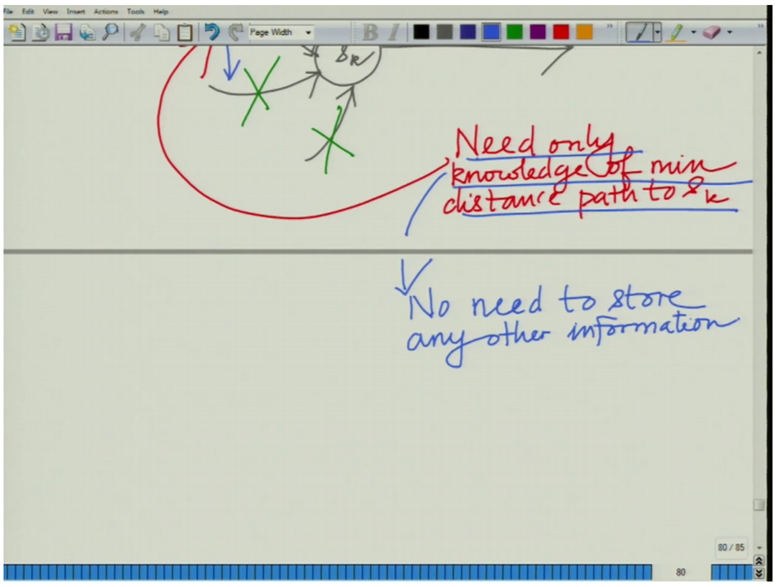


Contradiction!
minimum Distance.
Ignore rest.
dmin
s_k
Minimum Distance path From s_k
Need only knowledge of min distance path to s_k
79 / 85

So what is happening is you have x j minus 1 you have x j and there might be several paths coming into x j minus 1. Or they need not be e 1 and s j I just have s j minus 1 or I just have any node for that matter let us call it just avoid confusion let us call it s k.

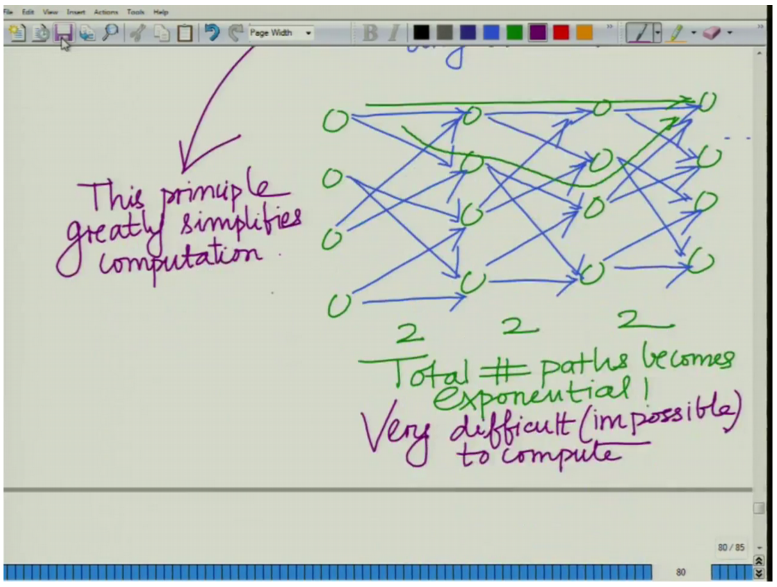
So, from s k I am interested in finding minimum distance path from s k. So, out of all these things, let say this is the minimum distance path to s k then I can ignore all these other paths. I only need knowledge, need only the knowledge of this minimum distance path, need only the knowledge of minimum distance path to s k. I can ignore, I can ignore the rest of the paths that is the whole, I can ignore the rest need only to. So, there is no need to. So, need only the knowledge of the minimum distance path no need to store any other information ok?

(Refer Slide Time: 18:54)


Need only knowledge of min distance path to s_k
No need to store any other information

So, no need to store at each. So, that significantly simplifies. Now you will not realize that this point why is important because if you go back to a trellis, if you look at your trellis correct remember the trellis has 4 states.

(Refer Slide Time: 19:18)


This principle greatly simplifies computation.
2
2
2
Total # paths becomes exponential!
Very difficult (impossible) to compute

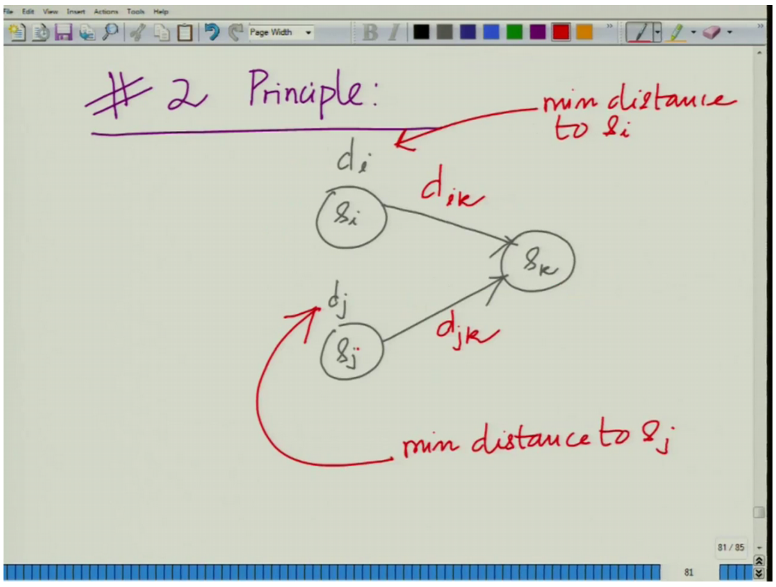
Let us go back to your trellis. So, I if I can illustrate why this principle is important if you look at this your trellis has 4 states, correct? And from each state there are 2 paths. So for instance, you have this for each state remember, you have 2 paths and this continuous, correct? Now what you will realize is, so to get to a state like this for instance, I can go through this path, I can go through for instance, for instance, this path I can go through for instance I can go through there are several other paths right.

So, the whole point is since each node. So, this split is into. So, each node split is into 2 paths, 2 paths, 2 paths. The total number of paths becomes exponential that is the whole point. So, you see first node first layer of node split is into 2 paths again at second time instance the node split into 2 paths third time inst. So, 2 into 2 into 2 into 2 after n stages the total number of possible paths through the trellis becomes exponential. So, you can come from a node you can take 2 paths, from the subsequent node you can take 2 paths from that node you can take 2 paths.

So, if you look at these all the possible paths through the trellis, the number of paths becomes exponential and therefore. If you have to keep track of the distances of all the paths it is impossible, that is not possible. And therefore, you need to find a way to optimize this efficiently, by reduce because exponential complexity no one can handle, it cannot be handled in a modern day computing system for instance if you have a 64 stage trellis the total number of possible stages will become 2 to the power of 64 if we have hundred stage trellis the total number of possible path will be 2 to the power of hundred. And that complexity cannot be you can have no computer or storage device that can store the information corresponding to 2 to the power of hundred path that is, that is a gigantic number. So, therefore, we this principle greatly simplifies the computation since at every stage or every node you only need to store one path which is the minimum distance path.

So, the number of paths becomes exponential this is very difficult. In fact, I will write impossible practically impossible, correct? For any practical number it is, for instance if you have 100 stage it is impossible very difficult or if not possible to compute. And for this principle this in principle gave greatly simplifies the computation that is computation require for decoding the received codeword.

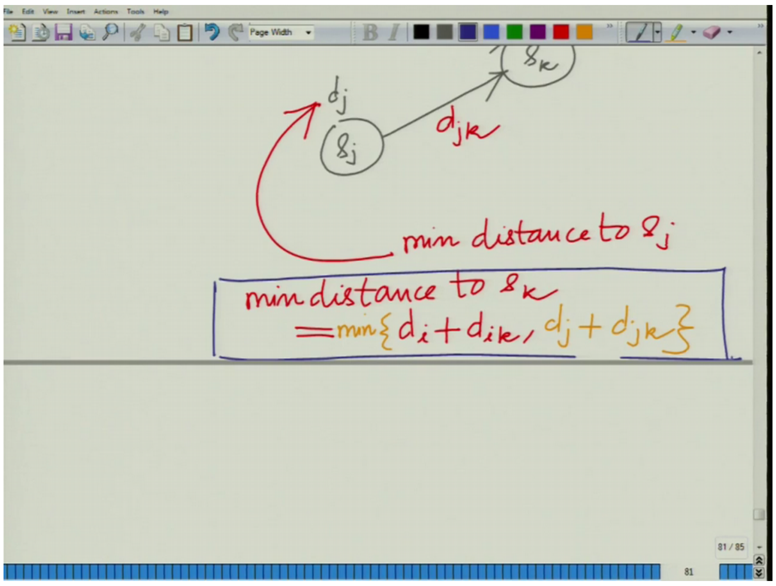
(Refer Slide Time: 23:48)



Now the second principle is very simple as follows if I look at the principle number 2. The second principle if I have 2 states s i, if I have 2 states and the corresponding distances to be is this d i and d j and you have to reach a third state s k and the distance. So, now, d i is the minimum distance remember we said we only need the minimum distance to s i. So, this is d ik this is d jk. So, d i this is the d i, this is the minimum distance to state i and d j is the minimum distance to state j.

Then the minimum distance to s j is k. So, minimum distance to s i is minimum distance to s i is d i minimum distance to s j is d j.

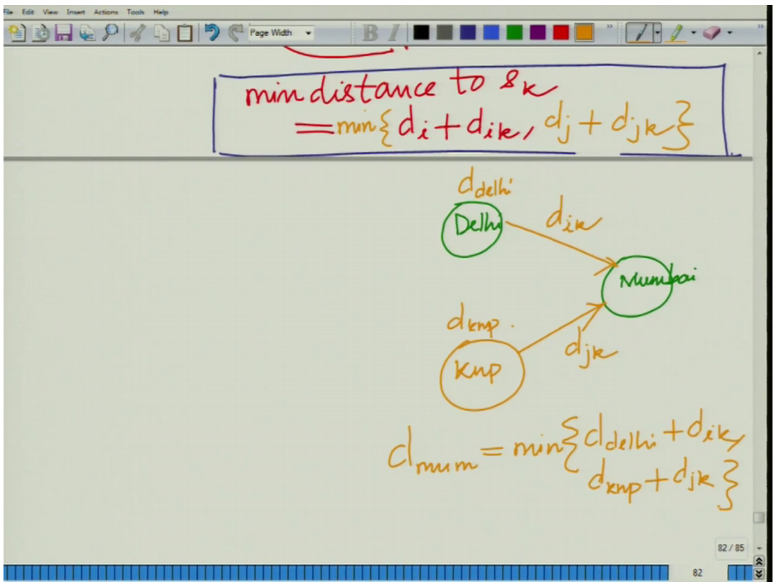
(Refer Slide Time: 25:18)



min distance to s_k
$= \min\{d_i + d_{ik}, d_j + d_{jk}\}$

Now, minimum distance to s k is obvious and s k you can either go through s i, but the minimum distance to s i is d i. So, the minimum distance to s i at minimum distance to s k via s i is basically d i plus d ik. So, minimum distance to s k via state I is d i plus d ik. And minimum distance to s k via d j will be minimum distance to j plus distance from the and therefore, the overall distance minimum distance to s k will be the minimum of these 2 and that is the important point. So, what we are saying is the minimum distance to s i. So, we can reach s k through either s i or s j.

(Refer Slide Time: 26:34)



min distance to s_k
$= \min\{d_i + d_{ik}, d_j + d_{jk}\}$

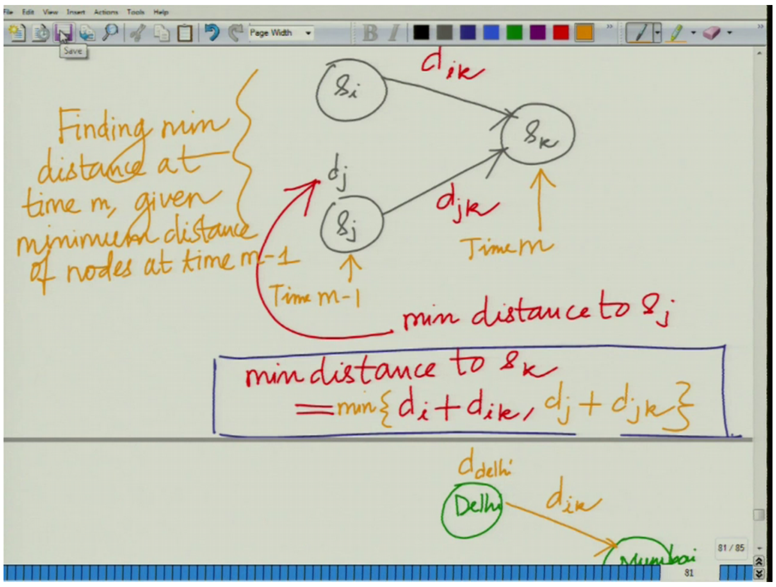
$$d_{mum} = \min\left\{\begin{matrix} d_{delhi} + d_{ik}, \\ d_{knp} + d_{jk} \end{matrix}\right\}$$

So, for instance let us again take a simple example. Let say I have, let say I need to travel to and let say I need to travel to Mumbai and this is Delhi or this is Kanpur. And I know the distance from Delhi to Mumbai I know the distance from. So, this is your d ik and this is your d jk and I know the distance to Delhi, this is the distance to Delhi minimum distance, this is the minimum distance to Kanpur and the minimum distance basically to Mumbai, if you can, if you have to go through either Delhi or Kanpur the minimum distance to Mumbai is the minimum of the distance to Delhi plus d ik comma the distance of Kanpur plus d jk.

That is all this says basically say something very simple that is, if I have to go through node k via either i or j. I know the minimum distance to i I know the minimum distance to j, I know the distance from i to k, I know the distance from j to k the minimum distance of course, to k via i has to be d ik at d i plus d ik if minimum distance to k via j has to be d j plus d jk and therefore, the overall minimum distance to k has to be the minimum of these 2 quantities as simple as that.

So, what I know. So, this helps me propagate the minimum distance to next layer. So, think of i and j as nodes at time at time, let say m minus 1 because we already used j.

(Refer Slide Time: 28:41)



Finding min distance at time m, given minimum distance of nodes at time m-1
s_i
d_{ik}
s_k
d_j
s_j
d_{jk}
Time m
Time m-1
min distance to s_j
min distance to s_k $= \min\{d_i + d_{ik}, d_j + d_{jk}\}$
d_{delhi}
Delhi
d_{ik}
Mumbai

So, let us say these are nodes at time m minus 1; this is the node at time. If I know the minimum distances to nodes at time instant m minus 1, then I can find the minimum distance to node at time instant m, as simple as that. Because to reach nodes at time instant m, I have to pass through nodes at time instant m minus 1 to reach a grater destination I have to pass through a hub. So, basically nodes at time instant m minus 1 are basically hub through which I have to hop to reach the node at time instant m.

So, if I know the minimum distance. So, if I know the minimum distances to each hop that is all of the nodes at time instant m minus 1. Knowing the distance from the hop to the destination that is the node at time instant m I can find the minimum distance therefore, to the destination that is all this is say. So, d min is to node s k is minimum d i plus d ik plus minimum d i plus d ik comma d j plus d jk this is an important and this we will use in basically, as I have said finding minimum at time m given minimum distance of nodes at time m minus 1. That is the important point. So, this will help finding the minimum distance at time m given minimum distance of node nodes at time m minus 1 that is the important point ok.

So, these are the 2 principles. The first principle is that if there is any minimum distance path that is passing through node s j minus 1, then the path up to node s j minus 1 must be the minimum distance path as well and 2, if there are 2 intermediate nodes and a final destination node and I know the minimum distances to each of the intermediate nodes and I know the distances from the intermediate nodes to the destination node then, the distance to the destination minimum distance to the destination node can be found using the second principle.

So, these are the 2 principles which might be little abstract at this point and you might find difficult to relate to the eventual viterbi decoder, but believe me. Firstly, you can see that these are very intuitive I mean all the way we have stated them formally these are very intuitive principles if you think about that. And 2 these are going to be, these simple principles are going to lay the foundation for the viterbi decoder. That is going to find the maximum likelihood, maximum likelihood estimate of the code word corresponding to the received code. So, we will stop here.

Thank you very much.