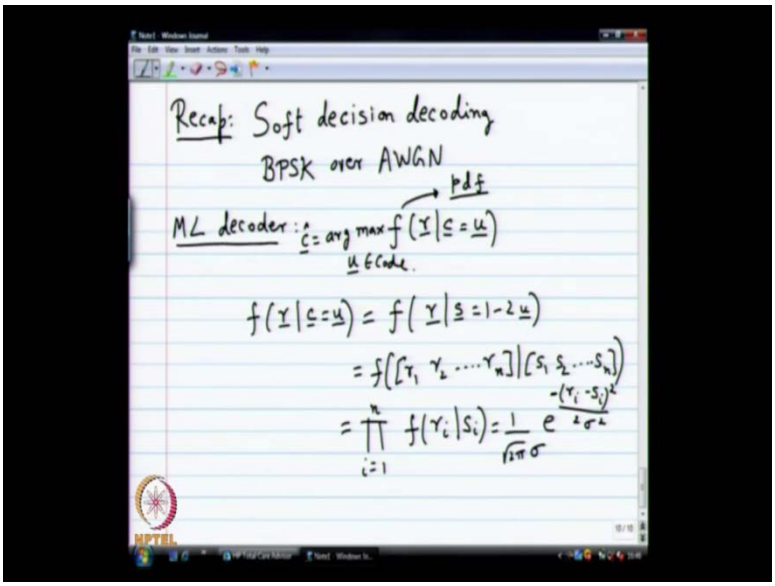
Coding Theory
Prof. Dr. Andrew Thangaraj
Department of Electronics and Communication Engineering
Indian Institute of Technology, Madras

Lecture - 18
Optimal Decoders for BPSK over AWGN

(Refer Slide Time: 00:19)



So let us begin. So, we are basically, so let me do a quick, very quick recap of where we were, we were basically looking at soft decision decoding. The setting is B P S K over a w g n, so this is a kind of technical way of describing the setting B P S K over A W G N and we want to do soft decision decoding. So the first kind of decoder that we are looking at, is the M L decoder which tries to compute f of r, given code word equals u maxima over all u in the code and take the argument as your c hat. This f is basically stands for p d f, can think of probability if you like, but this is the, this the expression.

So, let us try and simplify this, so this f of r given c equals u, so all this is quite standard, but I am doing it little bit more slowly, so that we can see what is happening, this is going to be f of r given s equals 1 minus 2 u, right?

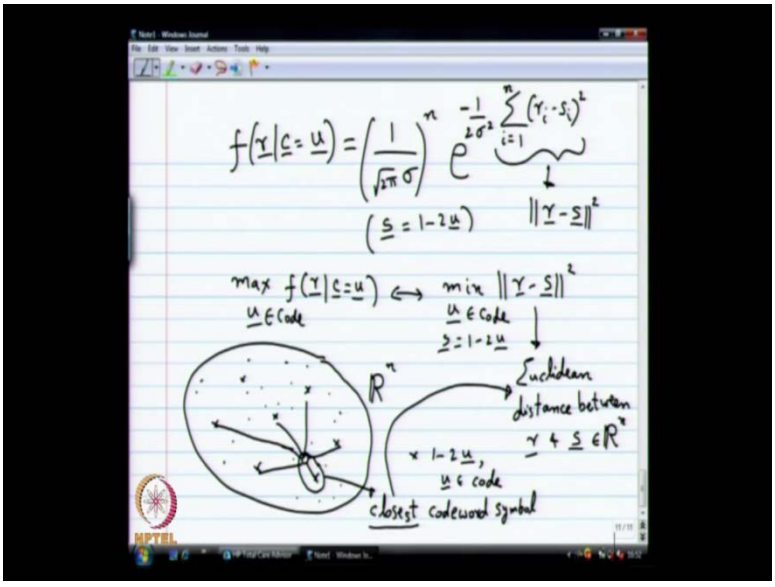
So, this guy, so what is this? Is a little bit more maybe it is a little bit more confusing. So, I will do this in a little bit more detail, this one is f of r 1 r 2 so on till r n given. Let us say some s 1 s 2 so on till s n. So, you can write, so remember s 1 to s n is given, so at some known symbol vector once the symbol vector is given the received values become

conditionally independent. So, you can write, if you want to write this conditional p d f, it is the same as the product of f of r i given s i equals 1 to n.

So, if I do not have the conditioning, I cannot write the joint p d f of r 1 through r n as the products of the p d f s of the r i, that is definitely not possible. Once, I have conditioned it then the s s f becomes fixed given that s is thus s is fixed the conditional p d f becomes product of the individual p d f s. It is more than one way to see this hopefully it is clear to you, if not try and convince yourself that this is true, ok?

So, once you that this is an easy expression to write down what is f of r i given as s i 1 by root 2 pi sigma e power minus r i minus s i square divided by 2 sigma square is that s i is something fixed. So, when you multiply, so hopefully this is clear, so 1 general thing I should warn you about this courses. I mean if you do an error control coding course today we started with linear algebra, linear block codes and then we moved on to abstract algebra, finite fields algebraic codes and then now we are moving on to probability, ok?

(Refer Slide Time: 04:23)



So, probability will play a big role for the rest of, the rest of the course. So, you have you need multiple frills and you have to be little bit comfortable, it is not that we are going deep into any one area, but still you need to be a little bit comfortable with the way how these things interact with each other, ok?

So, let us see now, if I do the multiplication, I am going to get remember what am I computing f of r given c equals u. Let us say s equals, let us say c equals u till the product. I wrote down it, down as a product, so when I multiply it out, I will get 1 over root 2 pi sigma occurring n times. Then, I have e power minus 1 by 2 sigma square sum from i equals 1 to r i minus s i square. So, right I am multiplying the e power minus something, so in the exponent it will add, I will get this expression.

So, this is exactly same as before. Now, remember what is s the only conversion, I should remember here is s is 1 minus 2 u. So, that is the conversion, I should remember now, I am trying to maximise the left hand side that is the same as maximising the right hand obviously now that is the same as minimising the exponent, right? This e power minus something.

So, if I want to maximise the whole thing it is the same as the minimising the exponent, so maximising f of r given c equals u over u is the same as minimising summation of this guy, this guy you can write it in compact as r minus s square. So, this is the Euclidean distance between the vector r and the vector s, so minimising r minus s square over u and s being 1 minus 2 u.

What is u? U is in the code, so I should write that down alright, so this is the Euclidean distance between the vector r and the vector s, so you have to imagine that these 2 guys are in some in the real vector space with n dimensions, is that ok?

So, once again you have a very nice geometric picture. So, what is this geometric picture which is very similar to, before if you have 0 1 n, it is not 0 1. Now, so that is the difference my circle, now is r n and then I have stars what are the stars symbol vectors, so codes word symbols, so to speak, so you convert the code word into symbols. So, these are the s, so these guys are the stars are the 1 minus 2 u, when you use the u belongs to the code.

So, these are you can call them various ways, I mean symbols vectors corresponding to the code word or coded symbols something like that code symbols or something like that. So, what is transmitted is one of these stars, one of these things is transmitted then what gets added to, it is some random vector z which will take a away to any one of these points, so all of these points in fact there as an uncountable infinite number of these things.

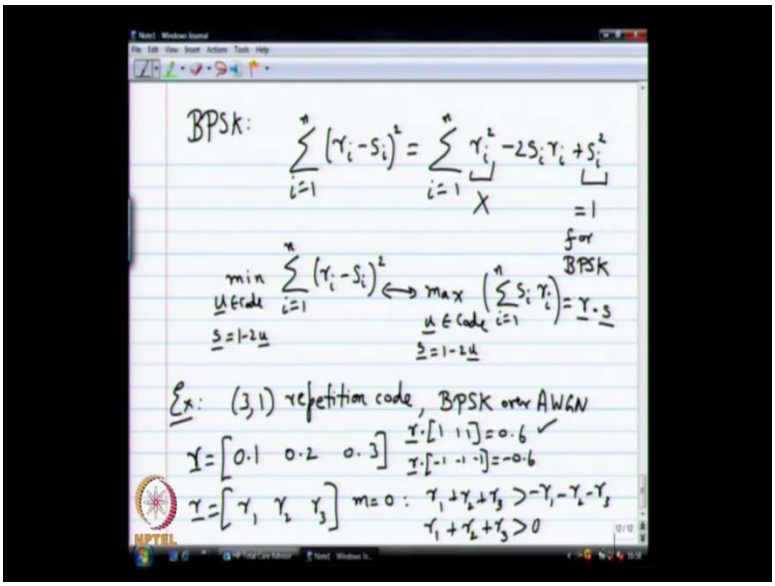
I am just drawing a few dots, do not get, do not be under the impression that those are the only possibilities. So, we have a huge number of possibilities, but they will be around the stars right, so they will be around the stars. Suppose, I am here that is my received value r what should my best possible decoder do the maximum likelihood decoder.

You should look at the distance from the stars all of them you have to look at all of them. Well, all of them and then pick that 1 which has the smallest possible distance, so the M L decoder is finding closest codes, codeword symbol the closest. Now, is defined with respect to Euclidean distance, so it is nice to have this view no, I mean you have to view this is in fact it is very reminiscent of the binary symmetric channel view right you. Remember, when we were doing the hard decisions no, not even thinking of r n just 0 1 n.

Once again, we had stars which were the codeword's themselves and then errors got added to it and then the M L decoder simply looks at closest codeword in closest in what sense in hamming distance. Now, you have to come to r n and once again you have an M L decoder, but the distance now is not hamming, but it is the Euclidean distance. The regular distance that you have in the n dimensions, ok?

So, in fact I mean, I have been talking about B P S K a lot here, but B P S K played very little role in this, in this entire derivation. If you see for any other consolation that you might have also a similar rule will come finally, if you list all your symbols that are possible clearly the best possible symbol vector is the closest 1 in Euclidean distance. There is no nothing wrong with that, so B P S K is just some playing a very small role here, hopefully that is clear to you is that alright, ok? So, for B P S K you can do a further simplification. So far B P S K did not play a big role like I said if you assume B P S K ok?

(Refer Slide Time: 10:36)



So, let us look at this summation i equals 1 to n r i minus s i square, this is summation i equals 1 to n r i square minus 2 s i r i plus s i square, so far i have still not used any B P S K knowledge. Now, I am trying to maximise over all possible vectors s, so the first term here is not going to be affected by changing s, so this I can drop from my calculation. Now, what about s i square?

This is always equal to 1 for B P S K again, if i change the symbol vector s because it is B P S K and I have picked that is minus 1, it is always going to be a constant and it is 1. So, I do not have to bother with that term also, if I want to minimise this expression over u n code s 1 minus 2 u. This is important B P S K is very important, this is the same as maximising over the u n code s is 1 minus 2 u.

What s i r i summation i equals 1 to n s i r i why? What do we know, this expression more familiarly, it is the dot product between r and s or the correlation between the 2 vectors r and s. So, of course this used B P S K, if you do not have B P S K this will not be true you will have to subtract this s i square from the correlation. You have to normalise the energies in the correlation once you do that. This is still useful, but this is also a standard notion in communications and detections that you can do a correlation here.

So, the way you think of dot product is you start look from the origin you have a received vector r you project it on to every symbol vector whichever gives you the

largest projection. You pick that one that is the same as minimising the distance we know that from classic results, so that is the same idea here, is that ok?

Now, it is time for an example. Let us do a very simple example and see how this works. The example we will do will be a three one repetition code, so three one repetition code. I am going to give you some received vectors and you have to tell me the message is that three one repetition code B P S K over A W G N, of course it is always going to be true.

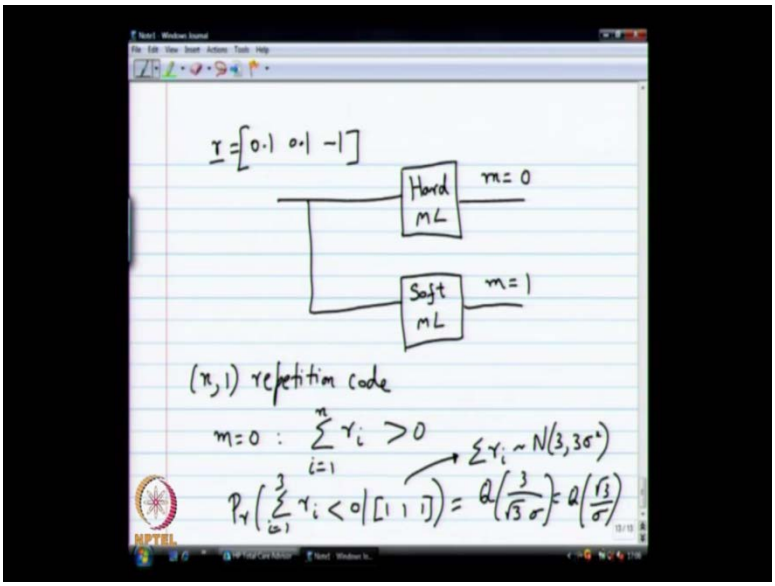
The way we will do this example is, I will give you some received vectors you have to do. Let us say M L soft decoding and tell me, what the transmitted bit was in the message bit? The message bit or the code word, anyone you can do is that, so the first one, I am going to give you is, let us say 0.1 0.2 0.3.

Quick to point out, so I mean, so you can do the correlation or anything else, but remember you have to only correlate with 2 different symbol vectors, one symbol vector is plus 1 plus 1 plus 1 the other symbol vector is minus 1 minus 1 minus 1 plus 1 plus 1 plus 1 corresponds to 0 minus 1 minus 1 minus 1 corresponds to 1. So, if you do the first correlation r dot o1 1 1. Simply, gives you .6 and r dot minus 1 minus 1 minus .1

This is clearly going to give you something negative which is in fact minus .6, so clearly positive number is greater than the negative number, so we pick this. So, let us let us try and write down a general expression. Suppose, I say suppose I do not really give you the r, say the r 1 r 2 r 3, what are you going to do? You are going to do, a dot product with 1 1 1 and compare it with the dot product with minus 1 minus 1 minus 1, so what is the condition, that M is 0.

M is 0, if r 1 plus r 2 plus r 3 is greater than minus r 1 minus r 2 minus r 3. Now, you can bring this to this side, we will get two times r 1 plus r 2 plus r 3 and 2 is clearly positive. So, you can throw it out, so this is this becomes equivalent to simply r 1 plus r 2 plus r 3 greater than 0. So, when will hopefully this is clear, this is no, so this is when M is 0 when will you say M is 1 r 1 plus r 2 plus r 2 is less than 0, so that is all it is very easy to look at these two, these two conditions. So any other r 1 r 2 r 3, I give you can very quickly find out. Next thing, I am going to ask you, is little bit more devious. So, what I want you to do is I want you to come up with an r, wich will give me different answers under optimal hard decision decoding and optimal soft M L decoding.

(Refer Slide Time: 16:49)



So, optimal hard M L decoding and optimal soft M L decoding, same r do you understand the question? Hopefully the question is clear, so it is an interesting question to consider, I want to come up with an r which when I decode with hard M L basically I make a hard decision and then I run my syndrome decoder, syndrome decoder is very easy for the repetition code right just majority you see which one is larger, I should get, let us say m equals 0. Then, if I run the same thing with soft M L, I should get m equals 1.

Something big right, so if you make it .1 .1. Anything greater than minus .2, so you make it minus 1, so what happens now, when you take hard decision is you are treating a value .1 the same as a value one ten times larger, but you are not distinguishing between the 2. You are simply slicing 1 as 0, slicing the other as 1, so when you do hard decision, you are going to get 0 0 1 and that when you do a syndrome decoding, will go to 0 0 0. On the other hand when you do a soft decoder you are taking into account all these things.

So, you are looking at r 1 plus r 2 plus r 3, which clearly goes to minus .8 and that will give you m equals 1. So, that is soft M L is that, so this is a simple way in which a three one repetition code works any questions on this. This is the simplest possible code, so it is very easy, so you can also generalise. Suppose I have an n 1 repetition code and I do same B P S K over A W G N, what will be the decoding rule? m will be 0, if yes, so summation over r i i equals 1 to n is greater than 0.

So, I have a very simple M L decoding rule for repetition codes, so you add up i equals 1 to n r i. If it is greater than 0, you say message was 0, if it is less than 0, is that is there a question something is disturbing you twice. If you have substantially high noise samples even that is allowed, but 1 high value is not allowed. What is? What do you mean by not allowed? Something like, that is the only question that I have, so the observation he is making, it seems like hard M L and soft M L agree. When there are 2 noise values which are large, but if there is 1 noise value which is large?

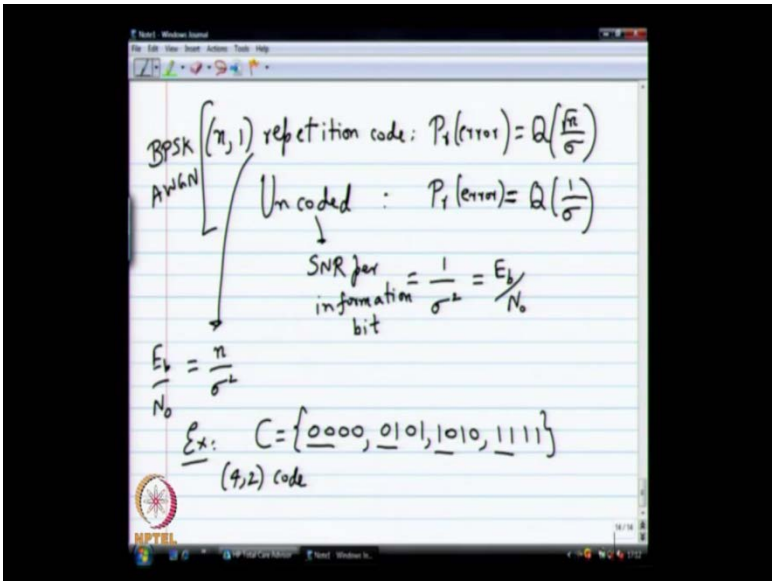
Then, they may not agree is that what you are saying that is, that is correct. That is why it will happen, so the question of when the hard M L decoder and soft M L decoder will agree and all are at least I do not know very simple descriptions bit complicated it is very hard to answer these things. So, you have to, so that is the modern view of things right, you do not have any deterministic clean answer it just works the way it works very hard to analyse this, but for the repetition codes things are very simpler I mean, we will see as we go along repetition is very easy the other 1s are tough even the simplest possible code after this we will see, will be much more difficult you cannot get a handle on what is happening, so this is the, this is the thing M L is equal to 0.

Now, what I want you to do is little bit more challenging is try and find out probability of error for the soft M L decoder. So, it is a little bit more twisted, so I will lead you a little bit, I would not just leave you a blank like this, so let us say m equals 0 as transmitted what is the probability that you will decide that m equal to 1. Let us compute that, so let us try to find probability that summation i equals 1 to n r i is greater than 0 given given 1 1 1 right. This is the probability that you will make an error, if no less than 0 right, I am sorry less than 0, I am sorry, but remember conditional on the fact that I sent 1 1 1. So, if you sent 1 1 1, what does summation r i become, so if you sent 1 1 1 summation, r i is simply normal with mean 3 and variance 3 sigma square is that.

So, it is basically 1 plus sigma square from 1 plus z z 1 1 plus z 2 1 plus z 3, so you will get 3 plus z 1 plus z 2 plus z 3 z y as basically i i d normal 0 sigma square. You add three of them, you get normal 0 mean 3 sigma square and 3 is 3 plus, that simply gives you normal with mean 3 and variance 3 sigma square, is that fine. So, when will a normally distributed random variable with mean 3 and variance 3 sigma square be less than 0, you can do that with q. This is simply q of 3 by root 3 sigma which is q of root 3 by sigma, so I have done it for n equal to 3, so this is for n equal to 3.

Now, there is no, there is no reason why you cannot generalise it to an arbitrary n, so if you put n here this will become normal with n and n sigma square. This will become n by root n sigma, so essentially you will get q of root n by sigma for the general case also, it is a very quick argument for the probability of error. You also have to argue for the opposite case m equals 1, but it will be symmetric, so you do not have to worry about it. The channel is a very nice channel, so you will get symmetric, so that you can fix that m equals 0 and you can do the problem is that, so you get q of root n by sigma, ok? So, I want you to point, i want you to point out something very interesting here.

(Refer Slide Time: 24:10)



So, if you use a n comma 1 repetition code repetition, code probability of error under M L decoding which is easy to do right probability of error. Soft M L decoding for repetition code is very easy will you need is an adder, you can implement it. This closes q of root n by sigma, so what is the encoded probability of error. If you do not code all, this is for B P S K over A W G N, right. So, remember that all this is for B P S K A W G N. If I do not do any coding at all probability of error is q of 1 by sigma, 1 by sigma.

So, this is standard B P S K simply q of 1 by sigma is that which will be lower will definitely be lower. So, it is look like the repetition coding is buying you a big advantage, but there is 1 minor cheating here, what is the cheating that i am doing yes, so you are sending n symbols to send only 1 information bit, so the amount of power you are spending to send 1 information bit.

In the coded case is n times as large as the power you are spending in the uncoded case, in the uncoded case every symbol carries 1 information bit, so you are you are sending spending n times as much power and you are not accounting for it. In your, in your S N R computation. So, remember for the uncoded case the S N R or S N R per information bit, what is S N R? By the way signal to noise ratio. I did not say it out loud hopefully it is known to most people.

Here, this would be 1 by sigma square, so the symbol power is just 1 right, so either minus 1 or plus 1 variance is just 1 what is the S N R per information bit here. So, S N R per information bit is also called E b by n nought for instance, so the S N R per information bit for the repetition code is what N by sigma square? N by sigma square.

So, this is E b by N nought it turns out if you express probability of error in terms of E b by N nought you will get the same answer for both right instead of expressing in terms of sigma. If you express it in terms of E b by N nought you get simply q of square root of E b by N nought. So, if you are constraint by S N R per information bit which is what you will be constrained by in most cases.

Repetition code does not buy you any advantage in probability of error for the same S N R per information bit, you get the same probability of error whether or not you do repetition coding. If that is the constraint then there is no problem, but if the energy per bit does not matter what matters is only energy per symbol then you will get a big benefit by repetition code.

So, that is something to remember if only 1 by sigma matters then you get a big advantage, if amount of power you spent per information bit matters then you do not get any advantage is that, so this is something that you have to account for usually when you do coding and I will talk more about this in the general case later on, but it is good to know, this even for the repetition code, ok?

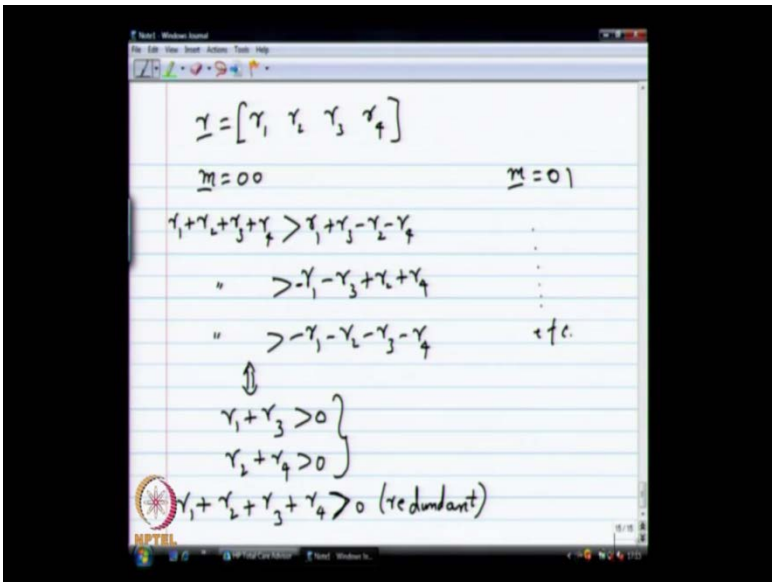
So, the way in which you phrase, it is to say that repetition code does not give you any coding gain in E b by N nought, you do not get any coding gain. So, to speak come back and see this things in more detail later, but for now this is a nice thing to look at. Ok, so let us do 1 more example just to get a little bit more confused simple enough example. Let us say my code now is a 4 2 code. He is not getting anything in terms of the probability of error . So, what is this question of maximum distance separable?

Well, it is more of an m it is, but so the question was does the maximum distance separable mean a lot of coding gain. That is your ultimate question, well obviously it does not right, so as I have shown, so n d s by itself does not mean much, so the rate matters. Of course, minimum distance matters in coding gain also you cannot say no, ultimately it will play a role particularly at very low error probabilities it will play a big role, but you will see there are lots of other things other than minimum distance also, ok?

So, this is a 4 comma 2 code. So, if you do a B P S K A W G N and if I give you an r which is some r 1 r 2 r 3 r 4, it is, it is immediately a little bit more complicated, it is not as easy as the repetition code right, so I have to decide between 4 different situations message being 0 0 0 1 1 0 and 1 1. Let us say the first two are the systematic locations, so you have to decide between the 4 possibilities and you have to do all the correlations, right.

So, you have to do 4 different correlations, so there might be smart ways to minimise your computation, so there are a lots of additions that you might repeatedly do for instance, I mean you can see some patterns. Here, there are smart ways to reduce your computation, but essentially you have to do those correlations and you cannot avoid it. It is very hard to compute for instance probability of error, ok?

(Refer Slide Time: 30:57)



So, let us try and do this, let us try and write it down in general and see what happens suppose I write down r being r 1 r 2 r 3 r 4. When will i say my message was 0 0? The

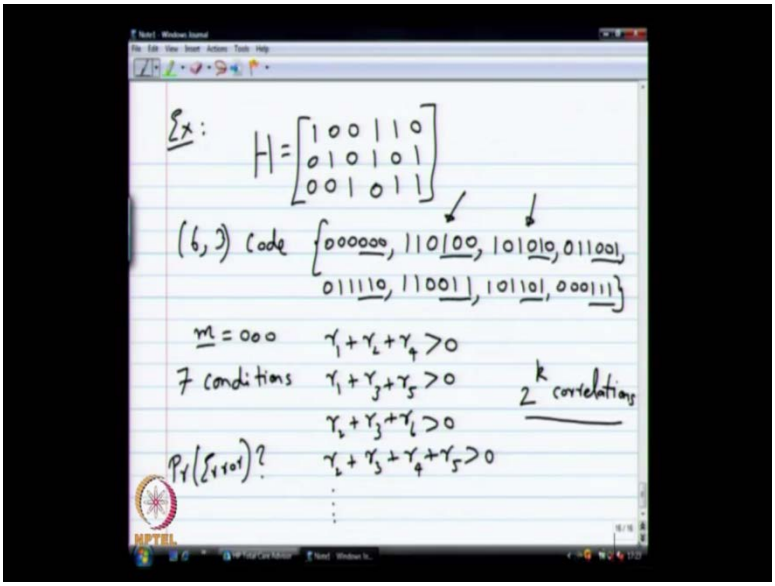
question is can we do something very smart for this specific case right. So, let me write it down then we will see, so maybe for this specific case there is something, but when I change the code you will see that it is a little bit more complicated.

So, let us start with this when do you say m is 0 0, you want so many things, right you want 3 things to be true what are the 3 things you want to be true r 1 plus r 2 plus r 3 plus r 4. which is your the correlation that you want should be greater than r 1 plus r 3 minus r 2 minus r 4.

The same guy should be greater than minus r 1 minus r 3 plus r 2 plus r 4 and then the last thing is minus r 1 minus r 2 minus r 3 minus r 4. So, like he was pointing out you can maybe play around with these equation, so simplify some of them alright. So, the condition you are getting here basically is this will become equivalent to r 1 plus r 3 being greater than 0 r 2 plus r 4 being greater than 0. The last one you do not have to check because once these two are greater than 0 then the last one will also be greater than 0. The last one is basically what r 1 plus r 2 plus r 3 plus r 4, has to be greater than 0.

So, it is a redundant condition you do not have to check it, so this is a simplification, but nevertheless you have these two, these two things to check, if this is true then you have possibility. So, this is something you can do, maybe you can even analyse this is not very hard, this is simple enough code you can do it, so you will have similar rules for m being 0 1 for m being 0 1 you will have similar rules etcetera. Is that ok?

(Refer Slide Time: 33:16)



So, let us go to a slightly more complicated example our usually standard code which was defined by this parity check matrix. I have been using it for syndrome decoding a lot, so this is a 6 comma 3 code and if you want I can list out all the 8 code words. So, it is a bit of an effort to even write down the codeword's, but I will do it, so it would not take too much time. Hopefully I am right, If I am making any mistake let me know. So, those are the 8 code words.

Well, the point I am trying to make it is not so important to get all the codeword's exactly right, but I mean you will see what I am trying to say, so it just becomes much more complicated. So, here the last 3 we will take as the systematic part, so that gives you the systematic part. So, suppose I say now what is the condition for m being 0 0 0 you will have 7 different conditions and it is not at all obvious what they are going to be. So, if it is you are going to get strange conditions, so for instance the well it is obvious it is going to be, but it is not very clear what their interrelationship is, so for instance 1 of the conditions will be.

Lt me write down some other conditions, it is not so hard, so r 1 plus r 2 plus r 4 is greater than 0. That will be the comparison with this guy comparison with this guy will be r 1 plus r 3 plus r 5 is greater than 0. So, then the next one will be r 1 plus r 3 plus r 6 is greater than 0, do you see how I am getting this is.

That correct, so you just correlate you have to do a correlation with this and compare it with that correlation wherever it is 0. You are going to cancel it out wherever there is ,1 you are going to get move it to the left side and then you can write r 1 plus r 3 plus r 4 plus r 5 is greater than 0, likewise and there is no obvious way to combine any of it or make sense of any of it, is there something which is redundant? It is a difficult question to answer. There might be you never know, but it is hard to answer, so in general there may not, you may not be able to simplify anything ,so for instance if you have to compute probability of error for the M L decoder it is going to be a nightmare. Why will it be a nightmare? What is it? That you have to do.

Ok, so what will happen is your conditioning on the codeword being all 0. So, what you will end up, getting is a large dimensional jointly Gaussian random variable for which you have to find some integration over the greater than 0 part. So, that integrations are impossible to get a nice close form answer, I mean you would not get a nice answer you

can approximate it. There are union bound techniques to approximate these things, so that is what most people do, but the main lesson I want to convey is, suppose I ask you the question same question I ask for the repetition code which was can you come up with the received vector which will give me different answers for hard M L decoder and a soft M L decoder.

It is a little bit harder already, it is, it is, it is not easy to get good handle on what the M L decoder is doing. That is the big point I want to make particularly as you code becomes larger and larger it is going to become messy. So, very hard to calculate in general the complexity will grow exponentially with k, it is going to be 2 power k. So, in general you need 2 power k correlations and that is very complex to do and you are not going to be able to finish it up in any finite time, which was I mean this was for instance the hard syndrome decoder, it is quite simple in this particular case at least it is a very small decoder, you can very easily do it and you can even easily analyse it, right.

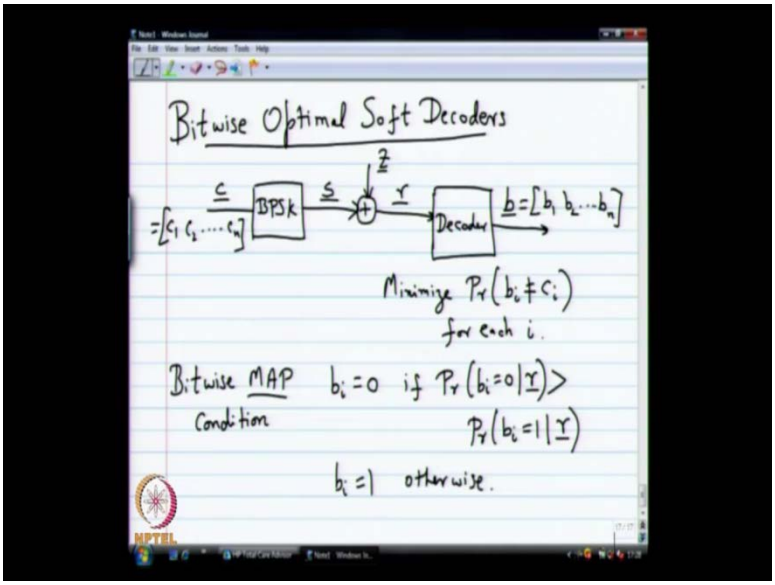
For the binary symmetric channel, you can easily analyse you will get a nice probability of error everything can be done for but, the soft even this is a little bit difficult can do it i am not saying it is impossible theoretically it is a very simply defined problem but, in practice it is difficult to get a good answer. Ok?

So, that is the M L decoder and we will kind of leave the M L decoder here, so we will come back to the M L decoder a little bit later in another context for some other code. So, it turns out for most linear block codes people do not implement M L decoder, so you can see, why it is no? It is not very easy to get a handle on people do not usually implement M L decoders only approximations, do it some approximations are easy even there it is hard what is in fact become much more popular today is the other kind of optimality that I was talking about where you do not try and minimise the probability of block error, but you try and minimise the probability of bit error.

So, it turns out for some reason that that becomes like a better problem because you can iterate on it in the M L decoder. You get the answer in 1 shot or you do not right, so at the entire codeword in 1 shot or you do not, but it turns out when you do this bit bitwise optimal decoders, you can iterate on it. You can first get a slightly better estimate for each bit and then a slightly better 1 and then a slightly better 1 that turns out to be incredibly useful in practice because I mean you can stop whenever you want.

That is something which is nice, I mean slightly better, slightly better eventually you stop, but in M L such ideas have not really taken shape, is there a way to slowly get something and then make it slightly better, make it slightly better definitely moving towards the best possible that people have to the best of my knowledge have not succeeded really in finding an efficient implementation, but for the bitwise optimal things they have succeeded and, so because of that bitwise things are much more popular today, ok?

(Refer Slide Time: 41:04)



So, let us see let us see how they work, so in a short time that we have I will only be able to define it. Then, I will define we will see, we will see some examples and we will stop with that. So, what we will see is bitwise optimal decode soft decision, decoder soft decision you can ask me what about bitwise optimal hard decoders they also exist. So, they are also implemented, but soft decoders is what we will see, so we will see hard decoders a special case also. It is not, it is not much more difficult to handle, so we will do this first, ok?

So, this optimality now, so remember now I picture again, I have a codeword c which goes through B P S K became as to which the noise gets added. Then, you get r i want to have a decoder which outputs. I will say some u, so let us not say u is a bit misleading. I do not want to use c cap because it is not necessarily a codeword, so let us just say what

else can you use here any ideas, we will do b b is better b for bit. So, it will output a vector b b 1 b 2 b n my goal here is that I want to minimise.

So, let us say c is c 1 c 2 c n. I want to minimise probability that b i is not equal to c i. Let us say for each i, so that is what I want to do, so there is some transmitted codeword bit, I want to output a bit in the ith position, so that the probability that my output is not equal to the transmitted codeword bit is minimum that is what I want to minimise. So, clearly there is no constraint that the entire vector should be a codeword, so that is a big difference between bitwise optimal decoders and M L decoders, M L decoders will definitely output only a codeword by definition.

They will only look for other codeword's bitwise optimal is different that way fundamentally it is different you only try to look at each bit and then i output each bit which is the best for that together. There is no constraint that should be a codeword that is not imposed that is not necessary in the bitwise optimal decoder in fact the best decoder which minimises this probability of error will not necessarily output a codeword.

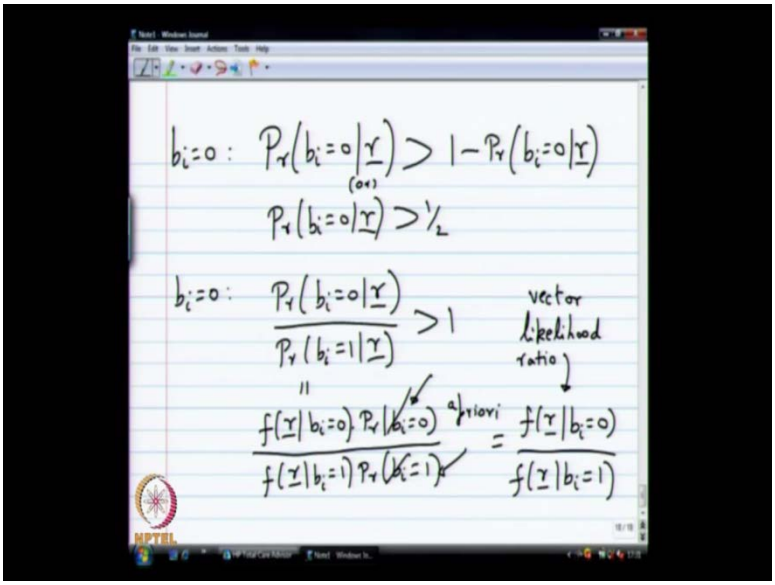
All the time which is very surprising, if you think about it if the codeword was definitely transmitted what it is not, it is not, it is not constraint to do that because particularly in the case that he was pointing out for instance 1 value becomes extremely noisy in those cases it becomes make sense to output something which is not a codeword. So, that is why it handles such things in a different way, ok?

So, how do we do this? actually the way to do it is not very hard I mean you know how to minimise any probability of error you can go through the expression you have to condition on the r and go through everything you will get this condition. The condition you will get is what is known as the bitwise m a p condition you basically say b i is 0. If probability that b i is 0 given r is greater than probability. That b i is 1 given r and you say b i is 1 otherwise it is very simple, I mean you can use the similar arguments before to show that the decision you have to make is like this, so given an r you have to compute these 2 probabilities which are known as the by which you can say are bitwise m a p probabilities what is m a p by the way?

Maximum a posterior decoders, so these are basically a posterior probabilities for each bit. You compute the a posterior probability for b i and see if b i equal to 0 is greater than probability. That b i is 1 and if that is true then you simply decide 0 or 1 now you know

that probability that b i equal to 1 will be what 1 minus probability that b i is 0 given r. So, you can, in fact only compute the left hand side and figure out if this will be greater, ok?

(Refer Slide Time: 46:38)



So, how do I do that? So, let me see that, so all I have to do is probability that b i equals 0 given r is greater than, so b i is 0 if it is greater than 1 minus, so am i getting that right? probability that b i is 0 given r, so you bring it to this side you get probability that b i is 0 given r is greater than half. If it is greater than half clearly that is going to be greater than the other probability, so this is another equivalent condition, ok?

Another very popular equivalent condition uses what is known as the log likelihood ratios, so what you do is b i is 0, so you go back and see this probability of b i is 0 is given r greater than probability of b i is 1 given r you divide both sides by the right hand side. So, you will get here probability that b i is 0 given r divided by probability that b i is 1 given r is greater than 1. So, this guy it is let me, so let me simplify this guy a little bit. I will call it something now this guy on the left hand side we can write as using Bayer's rule we can write it as f of r given b i is 0 times probability that b i is 0, of course divided by f of r, but anyway that will cancel the denominator.

So, i do not have to worry about it f of r given b i is 1 times probability that b i is 1. Remember, what are these guys these are a priori probabilities, if you assume your message is equally likely and if you assume your code is reasonable, what do I mean by

code is reasonable is code would not be such that some bit is 0 all the time. So, you do not want to send a code where 1 bit is 0 all the time. We might as well not send it, so if you do that it turns out for binary linear codes both of these will be equal and they will be equal to .5. You can show that it is not too difficult to show, so these two will cancel

So, this probably is 1 of your tutorial questions you can go back and check that there may be a proof for that also, these two will cancel and you will be left with this. So, this ratio actually becomes the same as the ratio of the likelihood of r given b i is 0 and the likelihood of r given b i is 1, so this guy is called the likelihood ratio. We can call it the vector likelihood ratio just to say that you are taking the likelihood ratio for the entire vector r. So, remember that is I mean there is a cancellation that I am assuming here, but like I said for linear block codes it is true, but in some cases it would not be true.

Then, you cannot do the cancellation you have to worry about what that means. So, but usually for B P S K A W G N you can cancel it is no problem, so this is the likelihood ratio and we will stop here about 30 seconds left. I do not want to introduce the next thing, so we will stop here and pick up from here in the next class.