Digital Communication Using GNU Radio

Prof Kumar Appaiah

Department of Electrical Engineering

Indian Institute of Technology Bombay

Week-04

Lecture-19

Welcome back to this lecture on Digital Communication Using GNU Radio. My name is Kumar Appiah and I belong to the Department of Electrical Engineering at IIT Bombay. In this lecture, we are going to continue our discussion on demodulation and signal space. If you remember, in the last lecture, we saw a detailed calculation of the basis vectors for a set of signals using Gram-Schmidt orthogonalization and I also made some remarks about how you can sometimes just by inspection infer some of the orthogonal basis vectors. Once we have a handle on the orthogonal basis vectors for the signals, the key idea that we are going to see is that the projection of the noise onto these orthogonal basis vectors is a sufficient statistic which in other words means that only getting the numbers by integrating with these size, integrating the noise with the size that is enough to determine optimally which symbol was sent or make the optimal choice. So let us look at projecting the signal and noise onto the signal space.

So we have

$$y(t) = s_i(t) + n(t)$$

with basis signals

ψ_1 , ψ_2 up to ψ_n .

If we now consider

$$\mathbf{y} = [\langle y, \psi_1 \rangle, \langle y, \psi_2 \rangle, \ldots, \langle y, \psi_n \rangle]^T ,$$

what this gives you is, this gives you a vector which basically gives a set of numbers obtained by projecting

$y(t)$ onto ψ_1 , projecting $y(t)$ onto ψ_2 and so on. In this case, I didn't add the subscript s_i , but you can just assume these s s to be s_i 's. So you obtain another set of n numbers by projecting

$s(t)$ onto ψ_1 , $s(t)$ onto ψ_2

and so on.

This is fine because your $s_i(t)$ was designed to be lying within the space of the signal ψ_1 to ψ_n and this is what we saw in the previous class that you constructed your orthonormal basis vectors ψ_1 through ψ_n to capture all the s_i 's. But the trouble is over here. Here the problem is that $n(t)$ is something which is a completely different beast. Its signal obtained because of noise is not predictable and you can most definitely say that you cannot write $n(t)$ as a linear combination of ψ_1 , ψ_2 , ψ_3 . That is $n(t)$ definitely has some components and variations that cannot be captured by just expressing it as the linear combination of

ψ_1 , ψ_2 up to ψ_n .

This leads to the question which is by projecting $n(t)$ onto ψ_1 , ψ_2 and ψ_n , which is what you are doing indirectly by projecting y onto ψ_1 , ψ_2 , ψ_n , am I losing some information? Is it that some information is lost when doing this particular operation that results in your not being able to make the optimal decision on which s_i was sent? The key idea is that we are trying to convert it to a vector problem as we mentioned. It is now instead of a waveform problem, we have converted it to a vector problem. What can we say about this n and its contribution to the vector decision problem? The first thing we are going to look at is what is the distribution of n ? If you remember we just did this particular exercise in a couple of lectures ago. If you now study the correlation

$\text{cov}(\langle n, \psi_k \rangle, \langle n, \psi_l \rangle) = \sigma^2 \langle \psi_k, \psi_l \rangle$.

So, inner product

$\langle \psi_l, \psi_l \rangle = 0$ if $k \neq l$ and

$\langle \psi_l, \psi_l \rangle = 1$ if $k = l$.

So this n is a Gaussian random vector consisting of iid entries. The variance of each component is $N_0/2$. In other words, you are going to get this vector n_1 , n_2 and so on up to n . Pardon me for the bad notation because this n was for number, this n was for noise, but all of these entries are iid. In other words, the covariance matrix is

$\sigma^2 I$.

So, these are iid entries. So the first thing that we are going to note is whenever you make a decision on which one was sent and so on, you know, these ψ_1 , ψ_2 , ψ_n capture that part of $n(t)$ into n_1 , n_2 and you know and so on and these n s are orthogonal because your ψ s are orthogonal. That is the first thing and this orthogonality translates to independence. So in other words, the entries of n are iid. The next question that we are going to ask is, is the restriction to the signal space optimal? That is what I was mentioning.

By restricting $n(t)$ to just look at those parts of n of t that are along ψ_1 , ψ_2 , ψ_n , are we losing something? Are we losing our ability to make an optimal decision on which a ψ was sent? The key concept is that we are saying no. The idea is that no, we are not losing any information. The part of $n(t)$ that is not along ψ_1 , ψ_2 , ψ_n is actually irrelevant. It is not at all, it has no bearing on your optimal decision to decide which ψ was sent. The concept can be also, you know, this concept is also known as the theorem of irrelevance and you know, irrelevant statistics and so on and it is not very difficult to see how.

What you are doing is, you are essentially, you have $y(t)$, you do not have $n(t)$ separately, you have only $y(t)$. You take $y(t)$ and project it onto ψ_1 , ψ_2 , ψ_n and then find out the residual signal that is you have $y(t)$, you project it onto ψ_1 , you project it onto ψ_2 , then construct the signal

$\langle y_1, \psi_1 \rangle \psi_1(t)$, $\langle y_2, \psi_2 \rangle \psi_2(t)$

and so on and then take it away from $y(t)$ and see what is there. So for example over here, if you do that,

$$y^{\perp}(t) = y(t) - \sum_{j=1}^{n} \langle y, \psi_j \rangle \psi_j(t) \quad ,$$

That is, I am taking away that part of y that is along the ψ_j 's to get the residual part and you know that your y is actually $s_i + n$. So, if you substitute it, you get

$$y^{\perp}(t) = y(t) - \sum_{j=1}^{n} \langle s_i, \psi_j \rangle \psi_j(t) - \sum_{j=1}^{n} \langle n, \psi_j \rangle \psi_j(t) \quad .$$

So, because your

$y = s + n$,

this part is the component of y that is contributed by s s, this part is that contributed by n s, this part is fully captured while this part is not fully captured. How does this reflect? Because if you now write

$y(t) = s(t) + n(t)$,

$s(t)$ and this particular part will cancel directly because $s(t)$ is fully captured, this part is actually $s(t)$. So, if you now look at

$y(t) - s(t)$,

you get

$n(t) - \langle n, \psi_j \rangle \psi_j(t)$

and let us call this $n^{\perp}(t)$, that is, that part of the noise that is not along the part that you observe which is along ψ_1 , ψ_2 , ψ_n . So, we are saying this n perpendicular which is the part that is lost because of your projections is irrelevant. How? The irrelevance of $n^{\perp}(t)$ can be ascertained by looking at the covariance of $n^{\perp}(t)$ and each of these n and ψ_k 's, that is because these are the parts that are going to determine which ψ was sent.

So, let us look at that. So if you now just evaluate the covariance of $n^{\perp}(t)$ and the part that is along the ψ_k , you will find that that is 0. In other words, you can just use the fact that the $n^{\perp}(t)$ is uncorrelated with each of the components of $\boldsymbol{n}$ which you get by projecting n along ψ_1 , ψ_2 , ψ_k and this uncorrelatedness translates to independence and this means that this $n^{\perp}(t)$ is irrelevant. So this detailed proof is something we are not looking at over here, but you can look at the references and get them, but the key idea is that because the part that is not along ψ_1 , ψ_2 is independent of the parts that actually matter when you compute your matrix to decide what is sent, this part is irrelevant. So one remark that we want to make is that whenever we perform this

$\langle s, \psi_i \rangle$,

if it is real signal, it is just integral, if it is complex then you have a star, it does not matter.

This is called a correlation operation. A correlation operation is basically where you multiply two signals and evaluate the integral maybe with a shift also, in this case we are

just not adding a shift that is completely fine, but because we are dealing with communication systems which are also signal processing systems in internally many operations or signal processing operations, it is convenient to express these in the form of a convolution. That is because for example even from an implementation perspective, if you implement some algorithms on a DSP, convolution is often implemented in a very efficient way. So these DSPs are designed to perform convolution in hardware and they are very fast and things like that. So given this, it is often the case that the matched filtering is implemented using a convolution.

So to do this what you need to do is, you want to find, actually this should not be $r_i(t)$. So $r_i(\tau)$ that is yes. So let us find this

$r_i(\tau)$ which is a signal obtained by performing,

$$\int_{-\infty}^{\infty} y(\tau)\,\psi^*(\tau - t)\,d\tau \quad .$$

Why is it $\psi^*(\tau - t)$? The reason is because you are essentially to perform correlation you are flipping the signal. If you flip and convolve that is same as correlation, because you know if you just do this kind of, let me actually just rewrite it in a proper way.

You are essentially performing

$$\int y(\tau)\,\psi^*(t + \tau)\,dt \quad .$$

This actually will be

$$y(t) * \psi^*(-t) \quad .$$

So this is like, maybe I will just express it in a correct way. So this is equal to

$$y(t) * \psi^*(-t) \quad .$$

That is what is happening.

So correlation can be obtained by performing flipped convolution. And if you want to just evaluate this quantity, you need to just substitute your $t = 0$. If you substitute your $t = 0$, you essentially get $r_i(0)$. So what you typically do is you perform the convolution with the flipped version of ψ and sample at the 0th point. So what you typically have is you have a bank of what are called matched filters.

So what are these matched filters? Basically you write ψ_0 or ψ_0^* depending on

whether it is real or complex. ψ_0 , ψ_1 , ψ_2 and you keep writing them up to $\psi_{n-1}(-t)$. Sometimes I use ψ_1 to ψ_n . In this case I use ψ_0 to ψ_{n-1} . So in this case you are going to get now several numbers r_0 , r_1 up to r_n which are essentially the inner products with each of the basis elements.

So this essentially is going to give you a set of numbers which you can collate into

$$\begin{bmatrix} r_0 \\ \vdots \\ r_{n-1} \end{bmatrix}$$

and this vector is all that you are going to need to make an optimal decision. Why? Because even though the part of $y(t)$ which is noise is not fully captured along ψ_0 to ψ_{n-1} , it is irrelevant to your decision making. There is one alternate interpretation of all this matched filtering and you know irrelevance. These so called matched filters collect all the necessary statistics to get the sufficient statistics from the noisy signal and throw away what is not needed. Let me give you a quick intuition as to why that is the case.

It supports that the signal you send is like something like this. Then what happens is that because of noise you end up getting something like this and intuitively the optimal thing to do is to average these things, average these out because if you just average these out then you can make a good decision on whether you are sending this or you are sending 0 or something like that. What does the matched filter do? The matched filter is actually this. You are multiplying and integrating which is the same as averaging. So the matched filter actually is designed in a way to collect all the necessary information from the noisy signal to make sure that you get the sufficient statistics.

So that is the key intuition that you have to take away from this. So this is the concept of matched filters. It's provided with different perspectives like you know optimal in terms of decision making and so on. But if you want to look at it from the correlation perspective to recover your original symbol that's also the same result. Now let us briefly look at optimal reception in the context of AWGN channels.

We are now performing M -ary signaling in an AWGN channel. So you have

$$\boldsymbol{y}=\boldsymbol{s}_i+\boldsymbol{n}$$

and $\boldsymbol{n}$ is as we just discussed a random vector with 0 mean and identity covariance of course identity multiplied by σ^2 . Now there are two ways to find the optimal symbol sent and both are very related. The first is the maximum likelihood decision rule. In the maximum likelihood decision case what we are saying is why do we maximize the

probability that we want to find that i such that this y was the most likely one.

The second one is the minimum probability of error in which case you also take into account the prior that is suppose that the probability of sending 0 is higher and the probability of sending 1 is lower. The minimum probability of error actually performs a modification for the priors. I will just try to give you an intuition as to why these rules come about. So if you remember the joint distribution I mean your $\boldsymbol{n}$ is actually Gaussian with 0 and $\sigma^2 \boldsymbol{I}$ as the variance which means your $\boldsymbol{r}$ or you know in this particular case we use the notation $\boldsymbol{r}$ where $\boldsymbol{r}$ vector is actually

$\boldsymbol{s}_i+\boldsymbol{n}$

that is what we got which means your $\boldsymbol{n}$ is essentially

$\boldsymbol{r}-\boldsymbol{s}_i$.

So let us just check what symbol we are using there.

We are using $\boldsymbol{y}$,

$\boldsymbol{y}-\boldsymbol{s}_i$.

So let us write the joint distribution of $\boldsymbol{n}$. So

$$f_{\boldsymbol{N}}(n)=\frac{1}{\sqrt{(2\pi)^{n/2}}\sigma}e^{\frac{(\boldsymbol{y}-\boldsymbol{s}_i)^H\boldsymbol{I}(\boldsymbol{y}-\boldsymbol{s}_i)}{2\sigma^2}}$$

So this is essentially what you have and this is the likelihood function also because I have substituted for N in terms of these and I want to maximize this.

So you want to maximize this. This particular thing is a constant that does not depend on i . I want to now find which i was sent. So what I am going to do is I am just going to maximize

$i=0,\ 1,\ 2,\ldots,M-1$

because there is just identity it is

$\boldsymbol{y}-\boldsymbol{s}_i$

it is essentially

$(\boldsymbol{y}-\boldsymbol{s}_i)^T(\boldsymbol{y}-\boldsymbol{s}_i)$

that can be written as

$\frac{\|\boldsymbol{y}-\boldsymbol{s}_i\|^2}{2\sigma^2}$.

But here this does not depend on i so all I need to do is I can take log and if I take log then I get

$$\max_i -\|\boldsymbol{y}-\boldsymbol{s}_i\|^2 = \min_i \|\boldsymbol{y}-\boldsymbol{s}_i\|^2$$

This translates to minimum distance decoding.

Why? Because you are saying let us just assume that my $\boldsymbol{s}_i$'s are vectors in some space and my $\boldsymbol{y}$ is a vector find me that $\boldsymbol{s}_i$ that is closest to this $\boldsymbol{y}$ in terms of squared distance. This means the maximum likelihood decoder is just the minimum distance decoder. This is for ML. Similarly for MPE it is not very difficult. For MPE all you need to do is you need to take into account the prior probability of what was sent.

So without going into too much details I am just going to write it as

$$ce^{\frac{-\|\boldsymbol{y}-\boldsymbol{s}_i\|^2}{2\sigma^2}}\pi(i)$$

where $\pi(i)$ is the prior probability. That means if you have a higher probability of sending 0 let us say that your messages are 0 and 1. Let us say the probability of sending 0 is 0.8 and the probability of sending 1 is 0.2. That is captured in $\pi(i)$. So I want to now maximize this over i which is same as maximizing I am just going to take log over here some monotonic function ignoring the constants

$$-\|\boldsymbol{y}-\boldsymbol{s}_i\|^2 + \log \pi(i) \quad .$$

This is log with respect to log to the base e. This is same as

$$\min_i \|\boldsymbol{y}-\boldsymbol{s}_i\|^2 - 2\sigma^2 \log \pi(i) \quad .$$

This is the MPE rule. The minimum probability of error rule. This comes about because see let me give you an intuition. If you have a very very high probability of 1 being or 0 being sent let us say. So even if you get something which is closer to 1 it could be because 1 was actually sent but the noise event was higher. So the minimum probability of error essentially takes that into account.

So I am skipping over the complete derivation but the key intuition is that if you write the likelihood function and you want to maximize it, the maximizing of the likelihood function essentially results in this particular result and if you go back to our slides it is

very evident this delta ML involves finding the argmin means just find me that i . So it finds that i that minimizes

$\|\boldsymbol{y}-\boldsymbol{s}_i\|^2$.

This is same as minimum distance decoding. Another way to interpret this is you can actually expand this. This becomes

$\boldsymbol{y}^T\boldsymbol{y}+\boldsymbol{s}_i^T\boldsymbol{s}_i-2\langle \boldsymbol{y},\boldsymbol{s}_i\rangle$.

So if you now look at

$\boldsymbol{y}^T\boldsymbol{y}$

that does not have i in it. You take that away and subtract so maybe I just do this once for you. If you look at

$\|\boldsymbol{y}-\boldsymbol{s}_i\|^2$

square that is equal to

$(\boldsymbol{y}-\boldsymbol{s}_i)^T(\boldsymbol{y}-\boldsymbol{s}_i)$

replace Hermitian for complex is equal to

$\boldsymbol{y}^T\boldsymbol{y}+\boldsymbol{s}_i^T\boldsymbol{s}_i-2\langle \boldsymbol{y},\boldsymbol{s}_i\rangle$.

This can be done either using the vectors or using the function integration also. If you want to know max, if you now want to minimize over this, you know you want to minimize over this.

That is same as maximizing and you take this away and put a negative sign and you divide by 2 you get

$\langle \boldsymbol{y},\boldsymbol{s}_i\rangle-\frac{\|\boldsymbol{s}_i\|^2}{2}$.

This is exactly what is being written over here you know and over here there are some advantages if you have constant enveloping meaning if you are all the $\|\boldsymbol{s}_i\|^2$ s are the same you can just find $\langle \boldsymbol{y},\boldsymbol{s}_i\rangle$ and find out which one was sent and you can just do a similar modification over here to get what was the best you know minimum probability of error decision. The only difference is the minimum probability of error takes into account the priors. If all $\pi(i)$'s are equal that is for example if you have M symbols and each of them has a probability of $1/M$ then this rule reduces to this

because there is no dependence on i . So, the MPE and ML are the same when all the original symbols are equally probable which is the case in the most general situations.

If you want to think of some situation where you know you do not have equal priors typically you know because these coding is performed to have the same priors for all the signals. So, like if you have so for example some data where there are many many zeros and few ones many many zeros and few ones typically they compress the data to have equal number of zeros and ones while they capture the original data. But let us suppose that you have a situation where you know it is one only when it rains but zero when it does not something like that then it may be more likely to have you know days when it does not rain you know. So, you have more zeros and ones and things like that or you are capturing a signal from a sensor and it detects some you know whenever there is an earthquake, earthquakes are rare so you have more zeros. Things like that may happen but typically it is the data is encoded in a way so that you have equal probable symbols.

So, this will be more common but this does have its uses whenever it arises. So, if you want a formal proof of this is just exactly what we did right now. $\boldsymbol{y}$ is a Gaussian random variable and its mean is $\boldsymbol{s}_i$ and covariance matrix is $\sigma^2 \boldsymbol{I}$. So, if you write the distribution of $\boldsymbol{y}|i$ is you know under the hypothesis i is

$$\frac{1}{(2\pi\sigma^2)^{n/2}} e^{\frac{-\|\boldsymbol{y}-\boldsymbol{s}_i\|^2}{2\sigma^2}} .$$

I just expanded that $\|\boldsymbol{y}-\boldsymbol{s}_i\|^2$ by applying that $\boldsymbol{C}_x^{-1}$ formula for you.

The ML rule wants you to maximize the quantity with respect to i and you know since log is monotonic you can maximize with respect to the log of this function as well. So, you can clearly see that only $\boldsymbol{y}-\boldsymbol{s}_i$ matters. In the case where you have MPE just add the $\pi(i)$ over here that accounts for the prior and your set. That is basically the proof that the minimum distance decoder is optimal in the case of additive white Gaussian noise. This is an important result and allows you to get a fair idea of what the so called decision regions are.

So, to find out which $\boldsymbol{s}_i$ was sent if you have $\boldsymbol{y}$. Let us look at a very quick example. If we consider on off signaling your $y(t)$ is essentially $s(t)+n(t)$ and if you were sent nothing your $y(t)$ is $n(t)$. So, hypothesis 1 is that some signal was sent, hypothesis 0 is that nothing was sent. So, in this particular case of binary signaling we are considering on off signaling.

So, there is either something sent or nothing sent. Let us compute a statistic Z which is

$\langle y, s \rangle$.

It makes sense because see here what you expect is a signal of about

$\|s\|^2$

while in the second case you expect a signal which is around 0. Because in the first case let us say you send a square wave you want a signal with amplitude around that square wave. In the second case you want a signal whose amplitude is around 0.

That is what you expect. So, the decision rule is to find the energy of the resulting signal and see whether the resulting signals energy is above or below $\|s\|^2/2$. That is the intuition. We do not know if this is correct, but we can use the ML to verify it and if you look at the ML and if you perform the ML in this case it is very easy because you have only a vector y and you have to decide whether it is your basically your s_i s are 0 or you know $s(t)$. So, if you now write the ML probabilities you will easily find that probability of making an error given that 1 was sent was when you sent 1, but for some reason you decided that 0 was sent and similarly you sent 0, but you decided for some reason that 1 was sent. So, for example if you have the situation where you send this because of noise it essentially comes like this then you make a mistake or if you send 0 because of noise it becomes something like this then you make a mistake.

So, these are actually in hypothesis testing these are the so called type there are these type 1, type 2 errors it is very similar. What is the probability that you make a mistake that is what is the probability that you actually sent $s(t)$ but you decided that 0 was sent or what is the probability that you sent 0 and decided that $s(t)$ was sent. So, these are some things that we will have to see for various modulation schemes as well and we will continue this in the forthcoming lectures. Thank you. Thank you.