EE 121: Coding for Digital Communication & Beyond	**Fall 2013**
Lecture 2 — September 3	
Lecturer: Prof. Anant Sahai	*Scribe: Jeff Lievense*

This lecture covers:

- Very Basic Communication
- Communicating a Single Bit
- Noisy Channels (Continuous)

2.1 Introduction

In this lecture, we will consider the problem of trying to communicate one bit. But first, we must establish what a bit is and how it is represented in the real world, and we must provide a mathematical model for what we call communication.

2.1.1 Bits

A bit (short for **bi**nary dig**it**) is the fundamental unit of information. It is something that can be in one of two states: true or false, on or off, high or low, one or zero, etc. By itself, a bit is a very abstract thing. How can we embody a bit in real life? Well, there are many things in the real world that can only be in one of two states. For example, consider a light. It is either on or off.

Let's say you want to communicate one bit of information to a friend who is too far away to hear your voice. With nothing but a flashlight, you could achieve this communication by turning the light on to represent a bit being 'one' and turning the light off to represent a bit being 'zero'. This particular method of communication is called **On-Off Keying** (OOK; 'keying' refers to the telegraph key, a device used in Morse code communication, where OOK originated).

We will now formalize a model for digital communication systems.

2.1.2 Modelling a Digital Communication System

Figure 2.1 shows the model we use to represent a digital communication system.

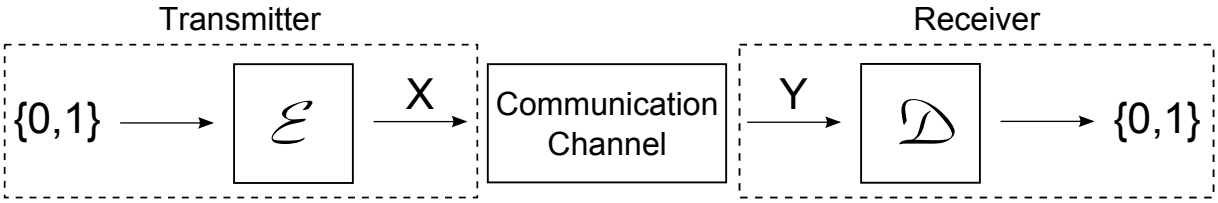


Figure 2.1. A digital communication system consisting of an information source, an encoder, a channel, a decoder, and an information sink.

From left to right, we have: an **information source** ($\{0,1\}$), an **encoder** ($\mathcal{E}$), a **communication channel**, a **decoder** ($\mathcal{D}$), and an **information sink** ($\{0,1\}$). The following describes a general framework for communication problems:

The blocks to the left of the communication channel make up the **transmitter**: The **information source** provides the information we would like to communicate, represented as a sequence of bits. The **encoder** performs the mapping $\mathcal{E} : \{0,1\} \to X$, where X is a signal which has the information in question somehow embedded in it. Next, the **channel** will, in general, alter the signal in a way such that the received signal $Y \neq X$.

The blocks to the right of the communication channel make up the **receiver**: The **decoder** takes the received signal Y and uses some decision rule to come up with $\hat{X}$, an estimate of X. $\mathcal{D}(\hat{X}) = \mathcal{E}^{-1}(\hat{X})$ is then computed and sent to the **information sink**, the final destination of the information being communicated. Note that while we always want the decoder to output bits, there are some situations in which the decoder is unable to make sense of the received signal. When faced with this issue, we will design the decoder to output 0, 1, or "I don't know".

In the case where the communication channel is simply a direct connection between the encoder and the decoder, we have $\hat{X} = Y = X$, and $\mathcal{D}(\hat{X}) = \mathcal{E}^{-1}(\hat{X}) = \mathcal{E}^{-1}(X)$, which is exactly what was sent from the information source. However, we tend to concern ourselves with more interesting and useful problems where the channel alters the signal X in such a way that $Y \neq X$.

Reliable communication is a problem of mapping the information to the signal X in such a way that the information can be recovered from the received signal Y by the decoder with high probability, regardless of how the channel alters the signal.

2.2 Additive Noise Channels

Let us now consider such a situation in which the channel alters the signal X. Specifically, we will be looking at a type of channel which adds **noise** N, an unwanted signal, to the signal we send through the channel, X, resulting in a received signal $Y = X + N$ (See Figure 2.2). For the remainder of this discussion, assume $X, Y, N \in \mathbb{R}$, for all sent signals X, received signals Y, and noise N (For clarity, you may think of X, Y, and N as voltage levels in a circuit).

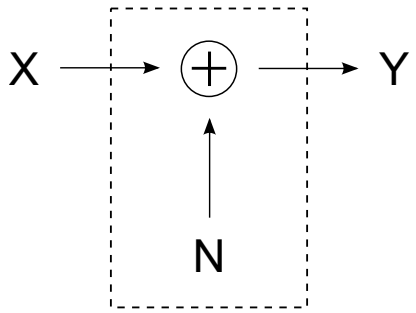


Figure 2.2. An additive noise channel.

With unwanted noise corrupting the transmitted signal X, we have to be careful in designing the decoder. But how can we even think about designing a decoder which filters out the noise if we don't know anything about the noise?

In general, there are two ways noise can behave: it can be out to get us, actively trying to mess with our signal with the intent of making it indecipherable, or it can be random.

- **Adversarial**
  - Knows our encoding/decoding scheme
  - Constrained in its ability to corrupt our signal
- **Random**

- Modeled by a random variable
- Distribution depends on source of noise

2.2.1 Adversarial Noise

We will first look at the case where the noise is out to get us and is aware of our encoding scheme. For now, assume $N \in [0, 1]$.

Perhaps we didn't know that the noise was adversarial when we were designing our encoder, and we chose the following mapping:

$$\begin{array}{c} \mathcal{E} \\ 0 \mapsto 0 \\ 1 \mapsto 0.1 \end{array}$$

In reality, X is some physical quantity like a voltage, so we may have designed our encoder to minimize power dissipation. Unfortunately, this scheme is no good! Since the noise is out to get us and knows our encoding scheme, it will take on the following values:

$$N = \begin{cases} 0.1, & X = 0 \\ 0, & X = 0.1 \end{cases}$$

Now $Y = 0.1$, regardless of the value of X! We have no way of determining what information was sent over the channel – the noise has successfully defeated us. However, just because the noise is adversarial doesn't mean it is impossible to successfully communicate! Note that the noise can only take values on the interval $[0, 1]$. If we know this information, we can re-design our encoder to make communication successful 100% of the time:

$$\begin{array}{c} \mathcal{E} \\ 0 \mapsto 0 \\ 1 \mapsto 1.0001 \end{array}$$

This works! In the worst case, N will take on the following values:

$$N = \begin{cases} 1, & X = 0 \\ 0, & X = 1.0001 \end{cases}$$

Which results in the following received values of Y:

$$Y = \begin{cases} 1, & X = 0 \\ 1.0001, & X = 1.0001 \end{cases}$$

Designing the decoder is easy – the noise can never push a transmitted 0 higher than 1, and a transmitted 1 will always be at least 1.0001 (since the noise is nonnegative). Thus we decode according to the following rule: if Y is strictly greater than 1, output 1. Otherwise, output 0. We call $Y = 1$ the 'decision threshold'.

2.2.2 Random Noise

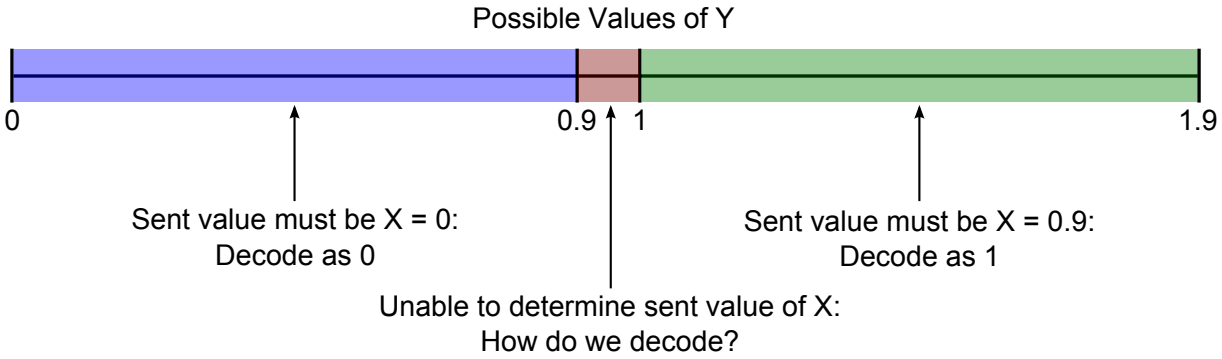
While adversarial noise is a pressing issue in areas such as military communication, most noise is not out to get us (for example, thermal noise in a conductor carrying a voltage signal). We now turn our attention to noise which can be modeled by a random variable.

Uniformly Distributed Noise

Consider $N \sim U[0, 1]$, i.e. N is distributed as a uniform random variable on the interval $[0, 1]$, and suppose we choose the following encoding scheme:

$$\begin{array}{c} \mathcal{E} \\ 0 \mapsto 0 \\ 1 \mapsto 0.9 \end{array}$$

If we receive $Y < 0.9$, obviously $X = 0$ was sent, since the noise is nonnegative (it cannot decrease the value of X). If we receive $Y > 1$, obviously $X = 0.9$ was sent, since the most the noise can add is 1. But what if we receive $Y \in [0.9, 1]$? How do we decide what was sent? Suppose we receive $Y = 0.93$. Maybe $X = 0$ was sent and the noise added $N = 0.93$, or maybe $X = 0.9$ was sent and the noise added $N = 0.03$. We can't know for sure, but we have to tell the decoder what to do for every possible input!



Let's say we put our decision threshold at $Y = 0.95$ (i.e. decode a received value as 1 if it's above 0.95, otherwise decode as 0). Now we can talk about the probability of error – the probability of incorrectly decoding the received signal Y. There are two cases in which we could incorrectly decode: when a 0 is sent but decoded as a 1, and when a 1 is sent but decoded as a 0.

$$P_{\mathcal{E},0} = P(\text{decode 1} \mid \text{send 0}) = P(N > 0.95) = \int_{0.95}^{1} f_N(n) \cdot dn = \int_{0.95}^{1} 1 \cdot dn = 0.05$$

$$P_{\mathcal{E},1} = P(\text{decode 0} \mid \text{send 1}) = P(N < 0.05) = \int_{0}^{0.05} f_N(n) \cdot dn = \int_{0}^{0.05} 1 \cdot dn = 0.05$$

If we knew $P(1 \text{ is sent}) = P(0 \text{ is sent}) = \frac{1}{2}$, we could calculate the average probability of error:

$$\begin{aligned} P_{\mathcal{E},\text{avg}} &= P(1 \text{ is sent}) \cdot P_{\mathcal{E},1} + P(0 \text{ is sent}) \cdot P_{\mathcal{E},0} \\ &= 0.5 \cdot 0.05 + 0.5 \cdot 0.05 \\ &= 0.05 \end{aligned}$$

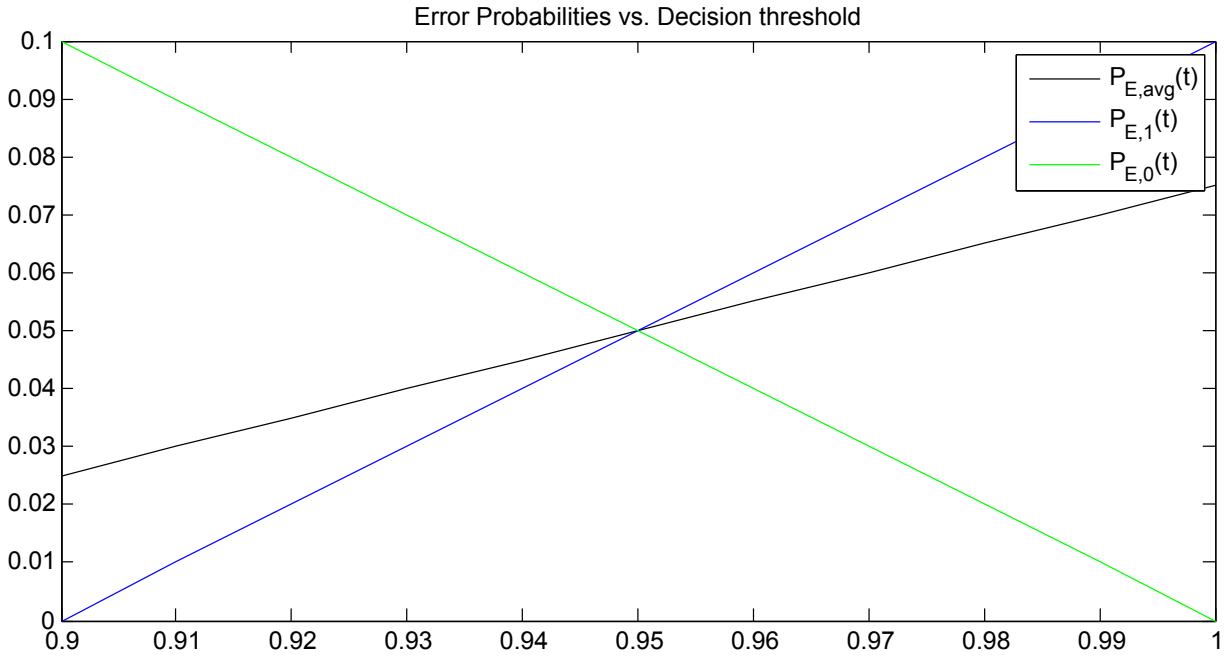
Determining the optimal decision threshold is intuitively simple when ones and zeros are equally likely, but what if we had a different prior on X? We could then weight $P_{\mathcal{E},0}$ and $P_{\mathcal{E},1}$ appropriately or push the decision threshold one way or the other to minimize the total probability of error. For example, suppose we now know $P(1 \text{ is sent}) = \frac{3}{4}$. How should we change the decision threshold in order to minimize the probability of error? Let's write an expression for the probability of error as a function of the decision threshold t. We can then calculate the optimal threshold by finding the value that minimizes the probability of error.

$$\begin{aligned} P_{\mathcal{E},\text{avg}}(t) &= P(1 \text{ is sent})P_{\mathcal{E},1}(t) + P(0 \text{ is sent})P_{\mathcal{E},0}(t) \\ &= \frac{3}{4}\int_{0.9}^{t} 1 \cdot dy + \frac{1}{4}\int_{t}^{1} 1 \cdot dy \\ &= \frac{3}{4}(t - \frac{9}{10}) + \frac{1}{4}(1 - t) \\ &= \frac{1}{2}t - \frac{17}{40} \end{aligned}$$

The probability of error is linear in the position of the decision threshold, so finding the value of t that minimizes $P_{\mathcal{E},\text{avg}}$ is easy – it is simply the lower endpoint. We are limited to the interval $[0.9, 1]$ and the slope is positive, so the left endpoint, $t = 0.9$, is the optimal decision threshold given this information about X.

This method of threshold detecting and choosing the optimal decision threshold is called **Binary Hypothesis Testing**. What we call $P_{\mathcal{E},0}$ is referred to as the probability of a **false alarm** (decoding a 1 when a 0 was sent), and $P_{\mathcal{E},1}$ is the probability of a **missed detection** (decoding a 0 when a 1 was sent).

It is worth noting that minimizing the average probability of error is not always the optimal strategy, depending on the nature of the information that is being communicated. For example, consider a diagnostic test for some disease. In testing for a very serious disease, missed detections can lead to unnecessary deaths. The instrument used to determine whether someone tests positive or negative for a disease will have some sort of decoder in it – in this scenario, that decoder will be designed to minimize $P_{\mathcal{E},1}$.



How we design our decoder depends on what we care more about: false alarms or missed detections?

Exponentially Distributed Noise

Let's now look at noise with a more interesting probability density. Consider $N \sim \text{Exp}(\lambda)$, i.e. N is distributed as an exponential random variable with rate parameter λ, and suppose we choose the following encoding scheme:

$$\begin{array}{c} \mathcal{E} \\ 0 \mapsto 0 \\ 1 \mapsto 0.9 \end{array}$$

When the noise was uniformly distributed over a finite interval, it was easy to think about the possible values of Y given that a zero or one was sent without explicitly drawing the distributions. However, for noise with non-uniform density and infinite support, it is helpful to look at the distributions of Y for each case ($X = 0$, $X = 0.9$; see Figure 2.3):

$$X = 0 \Rightarrow f_Y(y) = \begin{cases} \lambda e^{-\lambda y}, & y \geq 0 \\ 0, & y < 0 \end{cases}$$
$$X = 0.9 \Rightarrow f_Y(y) = \begin{cases} \lambda e^{-\lambda(y-0.9)}, & y \geq 0.9 \\ 0, & y < 0.9 \end{cases}$$

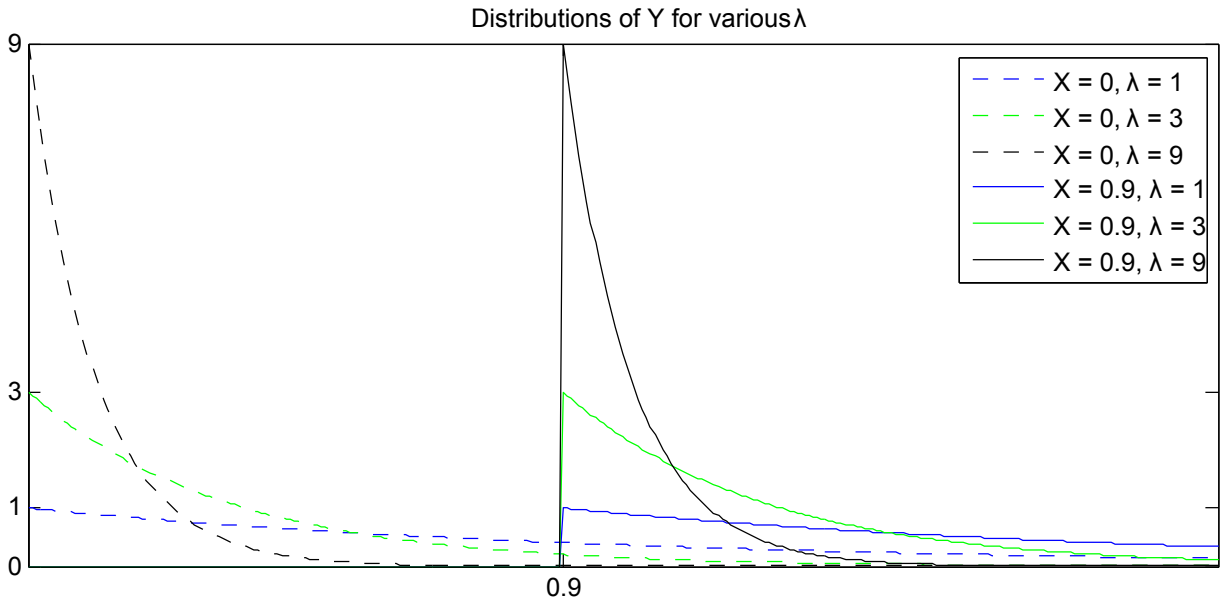


Figure 2.3.

Since the exponential distribution is also nonnegative, a received $Y < 0.9$ still means $X = 0$ was sent. However, if $Y \geq 0.9$, we are unable to say for sure which symbol was sent. Let's assume $P(1 \text{ is sent}) = P(0 \text{ is sent}) = \frac{1}{2}$ and again write an expression for the probability of error as a function of the decision threshold t. Recall $P_{\mathcal{E},1} = P(\text{decode } 0 \mid \text{send } 1)$, $P_{\mathcal{E},0} = P(\text{decode } 1 \mid \text{send } 0)$:

$$\begin{aligned} P_{\mathcal{E},\text{avg}}(t) &= P(1 \text{ is sent}) P_{\mathcal{E},1}(t) + P(0 \text{ is sent}) P_{\mathcal{E},0}(t) \\ &= \frac{1}{2} \int_t^\infty \lambda e^{-\lambda y} dy + \frac{1}{2} \int_{0.9}^t \lambda e^{-\lambda(y-0.9)} dy \\ &= \frac{1}{2} e^{-\lambda t} + \frac{1}{2}(1 - e^{-\lambda(t-0.9)}) \\ &= \frac{1}{2}(1 - e^{-\lambda t}(e^{0.9\lambda} - 1)) \end{aligned}$$

It turns out the probability of error goes something like $1 - \alpha e^{-t}$, a similar shape to the voltage on a capacitor being charged up over time – with no restrictions on the value of t, this function has no minimum! However, we can see that $P_{\mathcal{E},1} = 0$ for $t \leq 0.9$ and $P_{\mathcal{E},0}$ increases as t approaches 0, so we can assert $t \geq 0.9$. With this bound on t, the probability of error now has a minimum. It is a concave function (see Figure 2.4), so its minimum is at its left endpoint, which is $t = 0.9$. We will see in a later lecture that there is a much more intuitive way to come to the same conclusion (**Maximum a posteriori estimation**, a.k.a 'MAP Rule').

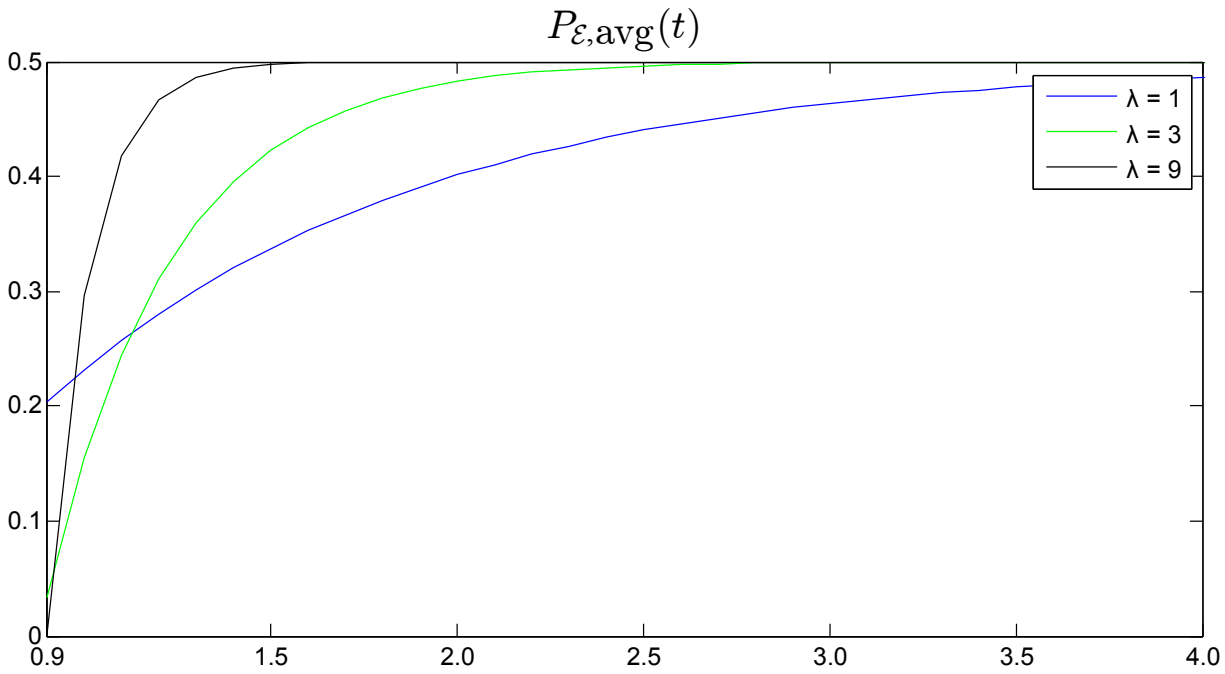


Figure 2.4. $P_{\mathcal{E},\text{avg}}$ versus the decision threshold t for various values of λ.

Normally Distributed Noise

Now consider $N \sim \mathcal{N}(0, \sigma^2)$, i.e. N is distributed as a normal, or Gaussian, random variable with mean 0 and standard deviation σ, and suppose we choose the following

encoding scheme:

$$\begin{array}{c} \mathcal{E} \\ 0 \mapsto 0 \\ 1 \mapsto \mathrm{A} \end{array}$$

Similar to the exponential case, we will first look at the distributions of Y for each case ($X = 0$, $X = A$; see Figure 2.5):

$$X = 0 \Rightarrow Y \sim \mathcal{N}(0, \sigma^2),\ f_Y(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-y^2}{2\sigma^2}}$$
$$X = A \Rightarrow Y \sim \mathcal{N}(A, \sigma^2),\ f_Y(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y-A)^2}{2\sigma^2}}$$

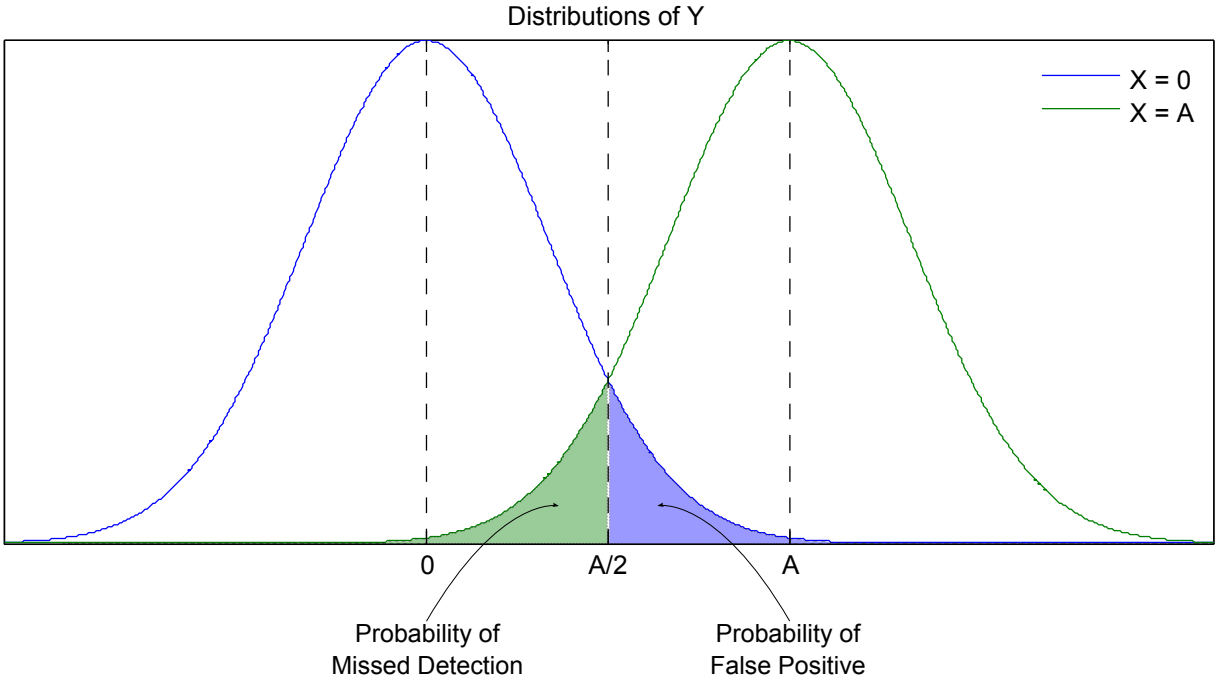


Figure 2.5.

Notice that, unlike the uniform and exponential cases, there are **no** regions where we can say with 100% certainty what value of X was sent! It may seem very clear

which symbol was most likely sent in some regions (e.g. $Y < 0, Y > A$), but it's important to note that there will always be some finite probability of error due to the infinite support of the Gaussian noise.

If we have no prior on X or if we know $P(0 \text{ is sent}) = P(1 \text{ is sent}) = \frac{1}{2}$, intuitively, we should set the decision threshold at $\frac{A}{2}$. We can now calculate the probability of error:

$$P_{\mathcal{E},0} = \int_{\frac{A}{2}}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-y^2}{2\sigma^2}} dy = \int_{\frac{A}{2\sigma}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} dx \triangleq Q(\frac{A}{2\sigma})$$

$$P_{\mathcal{E},1} = \int_{-\infty}^{\frac{A}{2}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y-A)^2}{2\sigma^2}} dy = \int_{-\infty}^{-\frac{A}{2\sigma}} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} dx \triangleq \Phi(-\frac{A}{2\sigma}) = 1 - \Phi(\frac{A}{2\sigma}) = Q(\frac{A}{2\sigma})$$

Note that we converted the distributions of Y to standard normal distributions via the change of variables from y to x (subtract the mean, divide out the standard deviation) – in dealing with standard normals, we can use the functions $\Phi(x)$ and $Q(x)$, the cumulative distribution and tail probability functions of the standard normal (respectively), instead of trying to integrate the distributions of Y directly.

If we know $P(0 \text{ is sent}) = P(\ 1 \text{ is sent}) = \frac{1}{2}$, then we can calculate the average probability of error:

$$\begin{aligned} P_{\mathcal{E},\text{avg}} &= P(1 \text{ is sent}) \cdot P_{\mathcal{E},1} + P(0 \text{ is sent}) \cdot P_{\mathcal{E},0} \\ &= \frac{1}{2} \cdot Q(\frac{A}{2\sigma}) + \frac{1}{2} \cdot Q(\frac{A}{2\sigma}) \\ &= Q(\frac{A}{2\sigma}) \approx \frac{1}{\sqrt{2\pi}} e^{\frac{-A^2}{8\sigma^2}} \end{aligned}$$

The last line is a well-known approximation of the tail probability function of the standard normal.

2.3 Energy

We saw that in an OOK communication system, $P_{\mathcal{E},\text{avg}} = Q(\frac{A}{2\sigma})$. Approximating the tail probability function showed us that $Q(x) \propto e^{-x}$. That means $P_{\mathcal{E},\text{avg}}$ decreases as A increases, so why don't we increase A until the probability of error is as low as we'd like? To answer this question, we must consider the physical implementation of our system: X most likely represents a voltage level, so increasing the transmitted

value of X costs energy!

In designing communication systems, we must consider the trade-offs between performance and efficiency. Let's say we're happy with the probability of error achieved via this On-Off Keying scheme in the presence of Gaussian noise, but we would like to decrease the energy consumption. How can we go about doing this without decreasing the probability of error? First, we need to define what we mean by energy. The energy in a discrete-time signal X is defined as:

$$E_X = \sum_{-\infty}^{\infty} |X[n]|^2$$

Since we are communicating a single bit in discrete time, our signal consists of a single sample, either $X = 0$ or $X = A$:

$$E_{X=0} = 0$$
$$E_{X=A} = A^2$$

Thus the average energy per bit in this particular system is:

$$E_{\text{avg}} = \frac{E_{X=A} + E_{X=0}}{2} = \frac{A^2 + 0}{2} = \frac{A^2}{2}$$

To retain the same probability of error but decrease energy consumption, we can implement a scheme called **Bipolar Phase Shift Keying** (BPSK) instead of OOK. In BPSK, our encoder performs the following mapping:

$$\mathcal{E}$$
$$0 \mapsto -\frac{A}{2}$$
$$1 \mapsto \frac{A}{2}$$

Note that the distributions of Y in this scheme are the same as the ones in OOK, just shifted to the left by $\frac{A}{2}$, so we have the same probability of error. Now let's calculate the average energy per bit:

$$E_{X=-\frac{A}{2}} = (-\frac{A}{2})^2 = \frac{A^2}{4}$$
$$E_{X=\frac{A}{2}} = (\frac{A}{2})^2 = \frac{A^2}{4}$$
$$E_{\text{avg}} = \frac{\frac{A^2}{4} + \frac{A^2}{4}}{2} = \frac{A^2}{4}$$

This is half of the average energy per bit required to implement OOK!

2.3.1 SNR

An important concept in communication systems related to signal energy is the **Signal-to-Noise Ratio** (SNR). It is determined by calculating the ratio of signal energy to noise energy. The SNR for a given communication system is sort of a measure of how well you can communicate – if the SNR is less than one, the noise might overpower your signal!

If the SNR is the ratio of signal energy to noise energy, what's noise energy? Recall that signal energy is the square of the amplitude of the signal. Similarly, noise energy is equal to the variance of the noise – variance is equal to the standard deviation squared. Standard deviation has units of amplitude, so variance has units of amplitude squared. Energy! With this in mind, let's calculate the SNR for our BPSK system:

$$\begin{aligned} E_{\text{signal}} &= \frac{A^2}{4} \\ E_{\text{noise}} &= \sigma^2 \\ \text{SNR} &= \frac{A^2}{4\sigma^2} \end{aligned}$$

SNR is a very important figure of merit that is ubiquitous throughout engineering. To wrap things up, let's examine the probability of error for our BPSK system in the context of its SNR. Since it's the same as the OOK system, we know:

$$P_{\mathcal{E},\text{avg}} = Q(\frac{A}{2\sigma}) \approx \frac{1}{\sqrt{2\pi}} e^{-\frac{A^2}{8\sigma^2}} = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\cdot\frac{A^2}{4\sigma^2}} \propto e^{-\frac{\text{SNR}}{2}}$$

We learned earlier that we could decrease the probability of error by increasing the amplitude A of our signal X. We see now that we were actually increasing the SNR! Thus we see the SNR as an indicator of our system's performance.