ECE 361: Lecture 10: Rate-Efficient Communication – Part I

In Lectures 5-6, we considered M-ary amplitude shift keying which allows the transmission of any desired number of bits per channel use over a Gaussian channel with arbitrarily high reliability, but at the cost of exponentially increasing amounts of energy per bit. Such schemes are said to be rate efficient but not energy efficient. In Lectures 7-9, we considered M-ary orthogonal, transorthogonal, and biorthogonal signaling schemes which can send any desired number of bits over a Gaussian channel with arbitrarily high reliability for a fixed amount of energy per bit, but require exponentially increasing numbers of channel uses to do so. Such schemes are said to be energy-efficient but are not rate efficient. The natural question to ask is whether it is possible to have a communication scheme that is *both* rate-efficient *and* energy efficient, and a little thought allows us to conclude that the question is obviously rhetorical, because if such schemes did not exist, there would be little reason to continue with this course!

In this Lecture and the next, we will devise a plausible argument (which can be patched up into a complete proof, if need be) that rate-efficient and energy-efficient signaling schemes exist. The existence of such schemes is demonstrated in *nonconstructive* fashion: we don't construct a specific signaling scheme and prove that it is efficient. Rather, the argument is based on consideration of *all possible signaling schemes* in a certain very large class of signaling schemes. This collection is usually referred to as an *ensemble* of signaling schemes. We will see that, if the rate of transmission (measured in data bits transmitted per channel use) is not too high, then the *average* of the error probabilities achieved by all the signaling schemes in the ensemble is small. Thus, at least one of the signaling schemes in the ensemble must have error probability as small as this average error probability. In fact, since the ensemble includes some demonstrably poor signaling schemes that no one would ever consider using because they have very high error probabilities, it is reasonable to believe that not just one but *many* signaling schemes in the ensemble perform even better than the average. Indeed, it can be shown that most signaling schemes in the ensemble are very good, and only a few are bad. What, then, is the problem? We could just pick one scheme at random from the ensemble and figure out its error probability. If the error probability is small enough to be satisfactory, then we are done. But if were unlucky enough to choose a bad signaling scheme whose error probability just is not good enough, we simply discard the scheme and pick another scheme at random, and figure out its error probability, and so on.... With very high probability, we would end up with a better-than-average signaling scheme after just a few tries. The difficulty is that for almost all of the signaling schemes in the ensemble, the receivers are completely impractical to implement. They require far too much computation or far too much memory to ever be a practical solution to the problem of achieving reliable communication. Thus, the major effort in communication system design has been to come up with signaling schemes that are feasible from the point of view of affordable implementation and that nonetheless provide good performance from the point of view of error probability.

10.1. Coded Communication Systems

10.1.1. Coding at the Transmitter

We wish to transmit k bits over a discrete-time Gaussian channel with high reliability, that is, with very small error probability. Let us assume that there is a peak power constraint that limits the energy to $\mathcal{E}$ per channel use, and suppose that for simplicity, we are using binary signaling, so that each symbol received is a Gaussian random variable with mean $\pm\sqrt{\mathcal{E}}$ with variance σ^2. The obvious method of transmitting k bits with k channel uses (at a rate of 1 bit per channel use does not provide sufficiently high reliability, and so we must try something else. The method used is called a *coding scheme* in which we transmit a *sequence* (or vector) of $n > k$ bits over the channel to tell the receiver which of the 2^k k-bit vectors is the message. Note that there are 2^n different binary vectors of length n, but since there are only $2^k \ll 2^n$ different messages, a *very small* fraction of the 2^n n-bit vectors are actually used by the transmitter. This notion of a coding scheme is by no means new to us. Recall that in Lecture 6, we considered *repetition coding* in which $k = 1$

and the transmitter sent $[+\sqrt{\mathcal{E}}, +\sqrt{\mathcal{E}}, \ldots, +\sqrt{\mathcal{E}}]$ to signal a 0, and $[-\sqrt{\mathcal{E}}, -\sqrt{\mathcal{E}}, \ldots, -\sqrt{\mathcal{E}}]$ to signal a 1 to the receiver. We can think of this as associating the n-bit vectors $[0, 0, \ldots, 0]$ and $[1, 1, \ldots 1]$ with data bit values 0 and 1 respectively, and then transmitting $+\sqrt{\mathcal{E}} = (-1)^0\sqrt{\mathcal{E}}$ for each 0 and $-\sqrt{\mathcal{E}} = (-1)^1\sqrt{\mathcal{E}}$ for each 1 in the n-bit vector.[1] Similarly, in Lecture 8, we considered Hadamard coding in which k bits were transmitted with $n = 2^k$ channel uses, and the n-symbol transmitted vector was one of the rows of the $2^k \times 2^k$ Hadamard matrix H_k, (times $\sqrt{\mathcal{E}}$, of course), *cf.* Eq. (7.3). Once again, we can think of this as first mapping the k data bits to n-bit vectors of 0's and 1's, and then translating the n-bit vector of 0's and 1's into a vector of $\pm\sqrt{\mathcal{E}}$'s for transmission over the channel.

10.1.2. Codes, Codewords, and Hamming Distance

The *set* of 2^k n-bit vectors of 0's and 1's that the transmitter is using is called the *code* $\mathcal{C}$, and each n-bit vector belonging to the set $\mathcal{C}$ called a *codeword* of the code. The *block length* or just the *length* of the code is n, the number of symbols in each codeword, while the *rate* of the code $\mathcal{C}$ is $R = k/n$ bits per channel use since we are sending k bits with n channel uses. Notice that there are 2^n different n-bit vectors of which the code $\mathcal{C}$ comprises only a minuscule subset of size $2^k \ll 2^n$. It is our fondest hope that the codewords (actually the set of 2^k vectors over the alphabet $\{+\sqrt{\mathcal{E}}, -\sqrt{\mathcal{E}}\}$) that we transmit are different enough from each other that the receiver can distinguish between them with high reliability. The vectors of $\pm\sqrt{\mathcal{E}}$'s are called codewords too, since these vectors are also two-valued but over the alphabet $\{+\sqrt{\mathcal{E}}, -\sqrt{\mathcal{E}}\}$ instead of the alphabet $\{0, 1\}$. Note also that if two vectors over $\{0, 1\}$ differ in some coordinate, say the i-th, then the corresponding vectors over the alphabet $\{+\sqrt{\mathcal{E}}, -\sqrt{\mathcal{E}}\}$ (obtained by mapping each 0 to $+\sqrt{\mathcal{E}}$ and each 1 to $-\sqrt{\mathcal{E}}$) also differ in the i-th coordinate. The *Hamming distance* $d_H(\mathbf{x}, \mathbf{x}')$ between two vectors $\mathbf{x}$ and $\mathbf{x}'$ is defined as the number of coordinates in which $\mathbf{x}$ and $\mathbf{x}'$ differ. It should be obvious that for *two-valued* vectors, it does not matter whether we are using the alphabet $\{0, 1\}$ or the alphabet $\{+\sqrt{\mathcal{E}}, -\sqrt{\mathcal{E}}\}$: the Hamming distance is the same. We shall be needing this fact later.

Let us recapitulate what happens at the transmitter and set up some more notation. The k data bits to be transmitted can be thought of as comprising a k-bit vector $\mathbf{u}$. There are 2^k such vectors and we denote them individually as $\mathbf{u}^{(0)}$, $\mathbf{u}^{(1)}$, $\ldots$, $\mathbf{u}^{(2^k-1)}$ where $\mathbf{u}^{(i)}$ is the k-bit binary representation of the integer i, $0 \leq i \leq 2^k - 1$. The vector $\mathbf{u}$ is *encoded* into a n-bit vector $\mathbf{x}$ called a codeword. There are 2^k such codewords comprising the code $\mathcal{C}$ that we are using. More specifically, suppose that $\mathcal{C} = \{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(2^k-1)}\}$ where $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]$ is the codeword into which the code $\mathcal{C}$ encodes the k-bit vector $\mathbf{u}^{(i)}$. The transmitter *modulates* the codeword $\mathbf{x}^{(i)}$ over the alphabet $\{0, 1\}$ into the codeword $\mathbf{X}^{(i)} = [X_1^{(i)}, X_2^{(i)}, \ldots, X_n^{(i)}]$ over the alphabet $\{+\sqrt{\mathcal{E}}, -\sqrt{\mathcal{E}}\}$, and sends this over the channel. Note that $X_j^{(i)} = (-1)^{x_j^{(i)}}\sqrt{\mathcal{E}}$ in accordance with our convention that 0's map to $+\sqrt{\mathcal{E}}$ and 1's to $-\sqrt{\mathcal{E}}$. Note that energy $n\mathcal{E}$ is used to transmit k bits so that $\mathcal{E}_b = n\mathcal{E}/k = R^{-1}\mathcal{E}$, i.e., the coding scheme is energy-efficient as long as we can find long codes of rate R.

10.1.3. Maximum-likelihood Demodulation

Suppose that $\mathbf{X}^{(i)}$ is the transmitted vector. Then the receiver gets the vector $\underline{\mathbb{Y}} = [\mathbb{Y}_1, \mathbb{Y}_2, \ldots, \mathbb{Y}_n]$ of n (conditionally) independent Gaussian random variables of variance σ^2 and mean $\mathsf{E}[\underline{\mathbb{Y}}] = \mathsf{E}[\mathbb{Y}_1, \ldots, \mathbb{Y}_n] = \mathbf{X}^{(i)}$. Of course, the receiver does not know which codeword was transmitted, and so it computes the 2^k different *likelihoods* $\Lambda_0, \Lambda_1, \ldots \Lambda_{2^k-1}$ of the observed value of $\underline{\mathbb{Y}}$ conditioned on $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(2^k-1)}$, the 2^k possible transmitted vectors, or, equivalently, $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(2^k-1)}$, the 2^k possible codewords.[2] Having computed the likelihoods, the receiver determines which codeword has the maximum likelihood. Let Λ_ℓ denote the largest likelihood where the receiver certainly hopes that with high probability ℓ equals i, that is, the codeword actually transmitted fortuitously has the largest likelihood. But it is possible that some other

[1] It is sometimes convenient to think of what happens at the transmitter as two distinct steps: *encoding* maps the data bit (*encodes the data bit*) into the vector $[0, 0, \ldots, 0]$ or $[1, 1, \ldots 1]$, and then *modulation* transforms the n-bit vector into $[+\sqrt{\mathcal{E}}, +\sqrt{\mathcal{E}}, \ldots, +\sqrt{\mathcal{E}}]$ or $[-\sqrt{\mathcal{E}}, -\sqrt{\mathcal{E}}, \ldots, -\sqrt{\mathcal{E}}]$ for transmission over the channel.

[2] Since k can be on the order of hundreds or even thousands of bits, such a computation is completely infeasible in practice, but we can *imagine* that it can be carried out and speculate on the results.

likelihood is larger. In any case, the receiver maps $\mathbf{X}^{(\ell)}$ back onto $\mathbf{x}^{(\ell)}$ and puts out $\mathbf{u}^{(\ell)}$, the k-bit binary representation of ℓ, as the most likely transmitted message. If $\ell = i$, the receiver decision is correct; else the decision is incorrect. We shall consider the probability of error later in this Lecture.

Since $\mathbb{Y}_1, \mathbb{Y}_2, \ldots, \mathbb{Y}_n$ are conditionally independent Gaussian random variables given the transmitted vector, the 2^k likelihoods that the receiver computes are all exponential functions. The receiver is not interested in the exact numerical values of the likelihoods but only in the identity of the largest, and thus it suffices to compare the *arguments* of the exponential functions, rather than compute the exponential functions and then compare them to see which is the largest. In other words, we can compare the *logarithms* of the likelihoods to determine which likelihood is the largest. Now, Λ_j depends only on $\underline{\mathbb{Y}} = [\mathbb{Y}_1, \mathbb{Y}_2, \ldots, \mathbb{Y}_n]$ and $\mathbf{X}^{(j)} = [X_1^{(j)}, X_2^{(j)}, \ldots, X_n^{(j)}]$ and we have

$$-\ln \Lambda_j = \frac{1}{2\sigma^2}\left[\sum_{m=1}^{n} (\mathbb{Y}_m - X_m^{(j)})^2\right] + \frac{n}{2}\ln(2\pi), \quad 0 \le j \le 2^k - 1. \tag{10.1}$$

The *largest* likelihood is the one for which $-\ln \Lambda_j$ is the *smallest*. But notice that the second term in (10.1) is the same for all j and thus does not affect the comparisons. Similarly, the factor $1/2\sigma^2$ multiplying the sum in (10.1) is the same and does not affect the comparisons. Thus it follows that the receiver chooses $\mathbf{X}^{(\ell)}$ as the most likely transmitted vector where

$$\sum_{m=1}^{n} (\mathbb{Y}_m - X_m^{(\ell)})^2 < \sum_{m=1}^{n} (\mathbb{Y}_m - X_m^{(j)})^2 \quad \text{for all } j \neq \ell,$$

that is, among all 2^k vectors $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(2^k-1)}$, $\mathbf{X}^{(\ell)}$ is the one that is at the *smallest squared Euclidean distance* to the received vector $\underline{\mathbb{Y}}$. For nonnegative x, y, $x^2 < y^2$ if and only if $x < y$, and thus $\mathbf{X}^{(\ell)}$ is the *closest in Euclidean distance* to the received vector $\underline{\mathbb{Y}}$. In other words, the receiver maps the received vector $\underline{\mathbb{Y}}$ into the *nearest* transmitted vector. This is a succinct description of the receiver operation, but it is worth emphasizing that in an actual implementation, exponential functions are not computed in comparing likelihoods and neither does anyone waste time taking the square root of $\sum_{m=1}^{n}(\mathbb{Y}_m - X_m^{(j)})^2$ to compute the Euclidean distance in order to find the nearest codeword: the comparison of squared Euclidean distances suffices. In fact, note that

$$\sum_{m=1}^{n} (\mathbb{Y}_m - X_m^{(j)})^2 = \sum_{m=1}^{n} \mathbb{Y}_m^2 + \sum_{m=1}^{n} (X_m^{(j)})^2 - 2\sum_{m=1}^{n} \mathbb{Y}_m \cdot X_m^{(j)} = \sum_{m=1}^{n} \mathbb{Y}_m^2 + n\mathcal{E} - 2\sum_{m=1}^{n} \mathbb{Y}_m \cdot X_m^{(j)}$$

where the first two terms are the same in all such calculations while the last sum on the right can be recognized as the *inner product* $\langle \underline{\mathbb{Y}}, \mathbf{X}^{(j)} \rangle$. Thus, an equivalent and equally succinct description of the receiver operation is that it maps $\underline{\mathbb{Y}}$ into the vector $\mathbf{X}^{(\ell)}$ that has the *largest* inner product with the observation $\underline{\mathbb{Y}}$. This inner product computation is sometimes referred to as a discrete-time correlation calculation: the receiver *correlates* the observation $\underline{\mathbb{Y}}$ with all the $\mathbf{X}$'s and then maps $\underline{\mathbb{Y}}$ into that $\mathbf{X}^{(\ell)}$ with which $\underline{\mathbb{Y}}$ has the largest correlation.

10.2. Error Probability of a Coded Communication System

Let $\mathbf{u}$ denote the data vector being transmitted. Thus, $\mathbf{u}$ takes on values $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(2^k-1)}$ with equal probability 2^{-k}. Similarly, $\mathbf{x}$ denotes the transmitted codeword which takes on values $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(2^k-1)}$ with equal probability 2^{-k}, while $\mathbf{X}$ denotes the transmitted vector which takes on values $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(2^k-1)}$ with equal probability 2^{-k}. The receiver computes the 2^k different likelihoods $\Lambda^{(j)}$ (or. equivalently, the 2^k different inner products $\langle \underline{\mathbb{Y}}, \mathbf{X}^{(j)} \rangle$) and decides that $\hat{\mathbf{X}}$ was the transmitted vector (equivalently, $\hat{\mathbf{x}}$ was the transmitted codeword or $\hat{\mathbf{u}}$ was the transmitted data vector. With E as usual denoting the event that the receiver output is not the same as the transmitter input, we have that $P(E) = P\{\hat{\mathbf{u}} \neq \mathbf{u}\} = P\{\hat{\mathbf{X}} \neq \mathbf{X}\} = P\{\hat{\mathbf{x}} \neq \mathbf{x}\}$.

We now break this expression down into smaller pieces and find upper bounds for each piece. We have that

$$P(E) = \sum_{i=0}^{2^k-1} P\{E \mid \mathbf{X} = \mathbf{X}^{(i)}\} P\{\mathbf{X} = \mathbf{X}^{(i)}\} = \frac{1}{2^k} \sum_{i=0}^{2^k-1} P\{E \mid \mathbf{X} = \mathbf{X}^{(i)}\}$$
$$= \frac{1}{2^k} \sum_{i=0}^{2^k-1} \sum_{j=0, j\neq i}^{2^k-1} P\{\hat{\mathbf{X}} = \mathbf{X}^{(j)} \mid \mathbf{X} = \mathbf{X}^{(i)}\} \tag{10.2}$$

in which all the probabilities are not easy to determine. But, let us consider the summand in (10.2). Let i and $j \neq i$ be fixed integers and suppose that the transmitted vector is $\mathbf{X}^{(i)}$. The receiver will set $\hat{\mathbf{X}} = \mathbf{X}^{(j)}$ if $\Lambda^{(j)}$ is the largest of the 2^k likelihoods, or equivalently, if $\langle \underline{\mathbb{Y}}, \mathbf{X}^{(j)} \rangle$ is the largest inner product of the 2^k such inner products. But, if $\langle \underline{\mathbb{Y}}, \mathbf{X}^{(j)} \rangle$ is the largest inner product, then it is certainly larger than $\langle \underline{\mathbb{Y}}, \mathbf{X}^{(i)} \rangle$. In other words, $\langle \underline{\mathbb{Y}}, \mathbf{X}^{(j)} \rangle$ = max inner product $\Rightarrow \langle \underline{\mathbb{Y}}, \mathbf{X}^{(j)} \rangle > \langle \underline{\mathbb{Y}}, \mathbf{X}^{(i)} \rangle$ and

$$P\{\hat{\mathbf{X}} = \mathbf{X}^{(j)} \mid \mathbf{X} = \mathbf{X}^{(i)}\} < P\{\langle \underline{\mathbb{Y}}, \mathbf{X}^{(j)} \rangle > \langle \underline{\mathbb{Y}}, \mathbf{X}^{(i)} \rangle \mid \mathbf{X} = \mathbf{X}^{(i)}\}. \tag{10.3}$$

The right side of (10.3) can be readily found. Note that $\langle \underline{\mathbb{Y}}, \mathbf{X}^{(j)} \rangle > \langle \underline{\mathbb{Y}}, \mathbf{X}^{(i)} \rangle$ if and only if $\langle \underline{\mathbb{Y}}, \mathbf{X}^{(j)} - \mathbf{X}^{(i)} \rangle > 0$, that is, if and only if

$$\sum_{m=1}^{n} \mathbb{Y}_m (X_m^{(j)} - X_m^{(i)}) > 0. \tag{10.4}$$

But, $X_m^{(j)}$ and $X_m^{(i)}$ have value $\pm\sqrt{\mathcal{E}}$ only, and hence the term in parentheses on the left side of the inequality in (10.4) has value 0 if $X_m^{(j)} = X_m^{(i)}$ and value $\pm 2\sqrt{\mathcal{E}}$ if $X_m^{(j)} \neq X_m^{(i)}$. More concretely, $X_m^{(j)} \neq X_m^{(i)}$ for exactly $d_H(\mathbf{X}^{(j)}, \mathbf{X}^{(i)}) = d_H(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})$ values of m. Thus, the left side of (10.4) is a sum of exactly $d_H(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})$ Gaussian random variables $\mathbb{Y}_m(X_m^{(j)} - X_m^{(i)}) = \mathbb{Y}(-2X_m^{(i)})$ with means

$$\mathsf{E}[\mathbb{Y}_m(-2X_m^{(i)})] = -2X_m^{(i)}\mathsf{E}[\mathbb{Y}_m] = -2(X_m^{(i)})^2 = -2\mathcal{E}$$

since, conditioned on $\mathbf{X}^{(i)}$ being the transmitted vector, $\mathbb{Y}_m$ has mean $X_m^{(i)}$. Also, $\mathsf{var}(\mathbb{Y}_m(-2X_m^{(i)})) = 4(X_m^{(i)})^2\mathsf{var}(\mathbb{Y}_m) = 4\mathcal{E}\sigma^2$, and since the $\mathbb{Y}$'s are conditionally independent conditioned on $\mathbf{X}^{(i)}$ being the transmitted vector, the variances add, giving us that the left side of (10.4) is a Gaussian random variable with mean $-2d_H(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})\mathcal{E}$ and variance $4d_H(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})\mathcal{E}\sigma^2$. Hence,

$$P\left\{\langle \underline{\mathbb{Y}}, \mathbf{X}^{(j)} \rangle > \langle \underline{\mathbb{Y}}, \mathbf{X}^{(i)} \rangle \;\middle|\; \mathbf{X} = \mathbf{X}^{(i)}\right\} = P\left\{\sum_{m=1}^{n} \mathbb{Y}_m(X_m^{(j)} - X_m^{(i)}) > 0 \;\middle|\; \mathbf{X} = \mathbf{X}^{(i)}\right\}$$
$$= Q\left(\frac{2d_H(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})\mathcal{E}}{\sqrt{4d_H(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})\mathcal{E}\sigma^2}}\right)$$
$$= Q\left(\frac{\sqrt{d_H(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})\mathcal{E}}}{\sigma}\right) \tag{10.5}$$

Substituting (10.5) into (10.2), we get

$$P(E) < \frac{1}{2^k} \sum_{i=0}^{2^k-1} \sum_{j=0, j\neq i}^{2^k-1} Q\left(\frac{\sqrt{d_H(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})\mathcal{E}}}{\sigma}\right). \tag{10.6}$$

This bound is also cumbersome to compute since there are $2^k(2^k - 1)$ different Hamming distances that must be found, and this is not a trivial task even for modest values of k. But, the result does allow us to make the inference that maximizing the Hamming distances between the codewords is a good thing, since it reduces the right side of (10.6). In the next Lecture, we shall see that while computing even the bound (10.6) on $P(E)$ is cumbersome for a *specific* code, computing the *sum* of the right sides of (10.6) over *all* possible codes is quite straightforward and leads to the surprising result that the *average* value of the right side of (10.6), averaged over all possible codes, is quite small.