6.441 Transmission of Information Feb 28, 2006

Lecture 06

Lecturer: Madhu Sudan *Scribe: Chung Chan*

1 Highlight of Previous Lectures

1.1 Lecture 1: Satellite Problem

In the first lecture, we are faced with the problem of transmitting the temperature measurements by a satellite back to the Earth through a noisy binary channel. We realized that encoding the temperature or the temperature difference one-by-one will lead to a binary sequence too long to be transmitted through the channel in real time, let alone the problem of data recovery in the presence of noise.

Intuitively, the problem of symbol-by-symbol encoding in this case is the integer constraint: we can only affort one bit per time unit or not assigning any bit at all. Perhaps we could solve this by encoding a sequence of symbols instead. If we assume expected number of occurrences of each temperature difference is the actual number of occurrences, we found that it is possible to encode 100 temperature changes in 77 bits. The remaining 33 bits is more than enough to correct error due to the noisy channel, given that we have perfect feedback of where the erroneous bit is, and that the actual number of error is its expected value.

Implicitly, we have broken the satellite communication problem into two: first, we try to compress the sequence of temperature changes by removing redundancy due to the integer constraint; second, we try to correct error due to the noise by injecting redundancy back into the sequence by resending the erroneous bits. Would this affect the overall optimality? Shall we compress by simply assuming that expectation is reality? The first question is to be answered by the source channel separation theorem, while the second will be taked in these two lectures.

1.2 Lecture 2: Entropy

In the second lecture, we are faced with the problem of quantifying the uncertainty associated with a random variable. Intuitively, a higher degree of randomness requires a longer description to completely resolve the uncertainty. We therefore constructed a coding scheme for a sequence of n i.i.d. Bernoulli(p) random variables $Z_1, \ldots, Z_n$, where p is unknown to the encoder. Such coding scheme is called universal because the source statistics is not perfectly known. Using the stirling approximation and Chernoff bound, the expected length of the codeword is approximately $-p\log p-(1-p)\log(1-p)$, which we take as the base case to induce the entropy formula by the grouping axiom that $H[p_1,\ldots,p_m] = H[p_1]+(1-p_1)H[\frac{p_2}{1-p_1},\ldots,\frac{p_m}{1-p_m}]$. The grouping axiom can be interpreted as the equivalence between asking which elements in $\{1,\ldots,m\}$ Z is and asking whether Z is 1 and if not, which elements in $\{1,\ldots,m\}$ Z is.

Although this derivation points out the close relationship of entropy and expected codeword length, we don't know how general the universal coding scheme it. More precisely, we have not proven whether the scheme is the best we could achieve in minimizing the expected length. Furthermore, the approximates and bound we use is tight only when n goes to infinity. That leaves us wonder what happens for finite n. Can we encode below entropy? The answer, as we will see in these two lectures, is yes, but not very much below entropy.

Although question 4 on p.42 of Cover indicates that we can replace this base case by the normalization condition $H_2[\frac{1}{2},\frac{1}{2}] = 1$ and the continuity condition of $H[p,1-p]$, this axiomatic formulation gives us less practical meaning of the entropy. In other words, the axiomatic formulation does not relate entropy to the expected codeword length at all.

1.3 Lecture 3: Properties of Entropy and Mutual Information

In the third lecture, we explored the mathematical properties of entropy and mutual information, in an attempt to justify our intuition of what information is. The conditioning does not increase entropy intuitively because additional knowledge can only help resolve ambiguity on average. Fano's inequality states that the error probability of estimating X from Y is bound to be large is the equivocation $H(X|Y)$, or the uncertainty in X after knowing Y remains large. Data processing theorem corresponds to the fact that further processing on an observation can only produce a second-hand information that is no better than the original in an estimation/detection problem.

Although these properties does not concretely define the practical meaning of entropy and mutual information, they agree with how we think randomness and information ought to behave.

1.4 Lecture 4: Asymptotic Equipartition Property (AEP)

The solution to Satellite problem assumed expectation is reality. Under what scenario is this assumption valid? By the law of large number for i.i.d. random variables (or the ergodic theory for the more general stationary ergodic process), reality indeed converges to expectation with probability one. More precisely, the n-sample probability converges to 2^{-nH} almost surely in the first order (in n) of the exponent. As a result, we can define a small ($\doteq 2^{nH}$) highly probable ($\Pr > 1 - e^{f(\epsilon)n}$ where $f(\epsilon) > 0$) set with an almost uniform probability distribution ($\doteq 2^{-nH}$).

As it will become clear later, typicality is an important concept in channel coding.

Typicality suggests a natural source coding scheme. If we encode just the typical set with $\lceil nH + \epsilon \rceil$ bits, the error probability will be less than $e^{f(\epsilon)n}$, meaning that the code will be asymptotically lossless as n increases. If we encode also the atypical set with $\lceil \log |\mathscr{X}| \rceil$ bits, we obtain a lossless code (not necessarily uniquely decodable[1]) with an expected length arbitrarily close to $\lceil nH + \epsilon \rceil$. The cost of going from lossy to lossless is therefore the weakening of the statement on the lengths of every codeword to the statement on the expected codeword length. But in both cases, we can draw a relationship between entropy and codeword length.

To make this relationship vigorous, we still have to prove whether typical set encoding is optimal. Intuitively, symmetry seems to support the use of fixed-length code for the typical sequences. But how could we argue that variable length could not do better? Furthermore, typical set encoding is just an asymptotic result for the the stochastic processes with AEP. What can we say about the case when n is finite, or when AEP does not hold? How general is AEP?

1.5 Lecture 5: Entropy Rate

To answer how general AEP is, we need to find the sufficient conditions for a stochastic process so that AEP holds. And to show that AEP holds for a stochastic process $\{X_i\}$, it is sufficient and necessary to prove that $-\frac{1}{n}p(X_1, \ldots, X_n)$ converges to its expectation $H(\mathscr{X}) := \lim_{n\to\infty} \frac{1}{n} H(X_1, \ldots, X_n)$, called entropy rate.

Stationarity of a process guarantees not only the existence of the entropy rate, but also that the rate converges to the limit $H'(\mathscr{X}) := \lim_{n\to\infty} H(X_n|X_{n-1}, \ldots, X_0)$. In the special case of a stationary Markov process generated by running a time-invariant, aperiodic and irreducible Markov chain with its unique stationary distribution μ_i, we showed that the entropy rate simplifies to $\sum_{i,j} \mu_i P_{ij} \log \frac{1}{P_{ij}}$ where $\mathbf{P}$ is the transition probability matrix.

Existence of entropy rate, however, does not imply that AEP holds. Consider the stationary process $\{X_i\}$ that is i.i.d. Bernoulli(1/2) with probability 1/2 and is deterministically 0.5 with probability 1/2. stationarity guarantees existence of the entropy rate. However, the sequence of 0.5 has probability 1/2 regardless on n while other sequences of 0's and 1's are equiprobable with a total probability accounting

[1] To make it uniquely decodable, we can add one additional bit to the front of every codeword to distinguish between the typical sequences and the atypical ones. Effectively, the bit makes the code prefix-free.

for the remaining 1/2. AEP does not hold for this process because 0.5 cannot be typical as it is the only sequence with probability 1/2, and the other binary sequences cannot be typical as their probability is only 1/2. It turns out that an additional constraint the the process being ergodic guarantees AEP.

On the contrary, a Markov chain being periodic or reducible exclude the existence of entropy rate. For examples, the periodic Markov chain with $P_{01} = P_{10} = 1$ and the reducible Markov chain with $P_{01} = P_{10} = 1/2$ and $P_{11} = 1$ have a unique stationary distribution, from which the entropy rate of the corresponding stationary Markov processes can be calculated. AEP also holds for both cases because the processes are ergodic.

2 Data Compression/Source Coding

What is data compression/source coding? In the satellite problem, we compressed the sequence of temperature differences into a binary sequence short enough to be transmitted through a channel. More generally, data compression refers to the process of generating a compact representation of the input data to match some interface.

Why is it possible compress data? As we concluded from the satellite problem, batching the input symbols allow us to remove redundancy due to the integer constraint, even though the input symbols are independent. Redundancy can also be due to patterns in the data. These patterns can be ascribed to the memory of the process. For example, the 2-state stationary Markov process with $P_{01} = P_{10} = 1$ has an alternating pattern. If we have to encode the process into a binary sequence symbol-by-symbol, one optimal way would be to encode 0 to 0 and 1 to 1 since the input symbols are equiprobable. However, if we batch the input symbols into sequences, we need only one bit to encode the two possible alternating pattern. From this example, we realize that redundancy can be due to the memory of the process rather than the integer constraint.

2.1 Introduction

A common (though not the most general) notion of code is a mapping from a countable support set of a discrete random variable to a D-ary sequence. More precisely,

Definition 1 (Code) *A code C of a random variable X taking values from $\mathscr{X}$ is the mapping $\mathscr{X} \mapsto \mathscr{D}^*$, where $\mathscr{D} := \{0, \dots, D-1\}$ is the D-ary alphabet set, and $\mathscr{D}^* := \bigcup_{n \in \mathbb{Z}^+} D^n$ is the set of all possible codeword lengths.*

2.2 Non-singular Code

The objective of data compression is usually to minimize the expected codeword length subjected to certain constraints on decodability. If $C(x)$ is a codeword with length $l(x)$ for an input symbol $x \in \mathscr{X}$, the expected length is $\sum_{x \in \mathscr{X}} p(x) l(x)$. A non-singular/lossless/decodable code C is defined as follows

Definition 2 (Non-singular code) *A code C is nonsingular if*

$$\exists Dec \, \forall x \in \mathscr{X} \; Dec(C(x)) = x$$

Dec is called the decompressor/decoder.

In many cases, we allow the code to be singular because the probability of error is small (e.g. the lossy typical set encoding), the error is neglegible (e.g. slightly distorted pixels sparsely distributed in an image), or it is simply infeasible to recover the input symbol error-free. The probability of error can be thought of as the expectation $E[1_{\{x \in \mathscr{X} : Dec(C(x)) = x\}}(X)]$, where 1_A is the indicator function $\mathscr{X} \mapsto \{0, 1\}$ that returns one when its argument is in A and zero otherwise. Another common choice is the mean

squared error. These types of measurement is called distortion/fidelity. The studies of the relationship between distortion and rate is the rate distortion theory.

To construct an example of non-singular code, consider the random variable X which takes the value $1, 2, 3, 4$ with probability $p_1 = 1/2, p_2 = 1/4, p_3 = 1/8, p_4 = 1/8$ respectively. An optimal fixed-length non-singular code would be,

$$\begin{aligned} C_1(1) &= 00 \\ C_1(2) &= 01 \\ C_1(3) &= 10 \\ C_1(4) &= 11 \end{aligned}$$

with expected length $L_{FL} = 2$ bits.

The fixed-length requirement is so stringent that the optimal codeword lengths are determined by matching the cardinalities rather the probabilities of each symbol. If we allow variable length code, an optimal non-singular code would be,

$$\begin{aligned} C_2(1) &= 0 \\ C_2(2) &= 1 \\ C_2(3) &= 00 \\ C_2(4) &= 11 \end{aligned}$$

with expected length $L_{NS} = 1.25$ bits. Optimality can be argued from the greedy algorithm of assigning the more probable symbols first to the shortest codeword not yet allocated.

2.3 Uniquely Decodable Code

Is there any weakness associated with the optimal non-singular code? To see this, let us define notion of an extended code,

Definition 3 (Extended code) *An extended code $C^{(k)}$ of a code C is a mapping $\mathscr{X}^k \mapsto \mathscr{D}^*$ such that*

$$C^{(k)}(x_1, \ldots, x_k) = \underbrace{C(1) \cdots C(k)}_{concatenated}$$

If $C^(k)$ is non-singular, it can be used to encode k symbols losslessly. This is more attractive then redesigning a non-singular code for the concatenated input symbols because doing so would generate a codebook with size exponential in k rather than constant with respect to k. For the optimal non-singular code example above, we have $C_2^{(2)}(13) = C_2^{(2)}(31) = 00$ and $C_2^{(3)}(113) = C_2^{(3)}(311) = 000$. Since the $\gcd(2, 3) = 1$, $C_2^{(k)}$ is non-singular for any $k > 1$ by induction. In otherwords, we cannot extend the optimal non-singular code to encode input sequences losslessly. How can we fix this?

Let us first define our desired code formally as follows,

Definition 4 (Uniquely decodable code) *A code C is uniquely decodable if its extended code $C^{(k)}$ is non-singular for all $k \geq 1$.*

It can be verified that this is equivalent to the condition that any extended codeword can be decoded even if the number of concatenations k is unknown. In other words, the union of all extended codes, called the extension $C^* : \mathscr{X}^* \mapsto D^*$, where $\mathscr{X}^* := \bigcup_{n \in \mathbb{Z}^+} \mathscr{X}^n$, is a non-singular code. Hence, it is unnecessary for the decoder to know the number encoded symbols apriori. It can figure that out directly by decoding the received codeword. For notational simplicity, we will often use $C(x_1, \ldots, x_k)$ instead of C^* or $C^{(k)}$ because the number of arguments tell us the order of the extension.

The reason why the optimal non-singular code C_2 is not uniquely decodable stems from the problem that $C_2(1)C_2(1) = C_2(3)$, which requires that $C_2(1)$ be a prefix of $C_2(3)$. To see it more clearly, consider the following code,

$$\begin{aligned} C_3(1) &= 0 \\ C_3(2) &= 10 \\ C_3(3) &= 110 \\ C_3(4) &= 111 \end{aligned}$$

with the expected length $L_{PF} = 1.75 = H(X)$. This is called the prefix-free/instantaneous code because no codeword is a prefix of another codeword. This leads to the following definition,

Definition 5 (Prefix-free code) *A code C is prefix-free if no codeword is a prefix of another codeword.*

Intuitively, extended prefix-free code should be prefix-free, which implies non-singularity. If the codeword of the extended code has a prefix that is a codeword of the unextended code, the prefix cannot be a concatenation of more than one codeword because of the prefix-free condition. Thus, we can decode it immediately, remove it from the codeword, and repeat the process until the entire codeword is decoded. This explains not only why the code is uniquely decodable, but also why it is called instantaneous. Let's state this more precisely as a theorem,

Theorem 6 *Any prefix-free code is uniquely decodable.*

Proof Suppose C is prefix-free. If two distinct input sequences $a_1 \cdots a_m$ and $b_1 \cdots b_n$ have the same codeword $c_1 \cdots c_l$. i.e.

$$\begin{aligned} C^*(a_1, \ldots, a_m) &= c_1 \cdots c_l \\ &= C^*(b_1, \ldots, b_n) \end{aligned}$$

Then, we have $a_1 = b_1$ by the following proof of contradiction. If $a_1 \neq b_1$, $C(a_1) \neq C(b_1)$ by non-singularity, implying that $C(a_1)$ must be a prefix of $C(b_1)$ or vice versa in order to have the same codewords for the two input sequences. This immediately lead to a contradiction because C is prefix-free.

Suppose, without loss of generality, that $C(a_1) = C(b_1) = c_1 \cdots c_k$. Then,

$$\begin{aligned} C^*(a_2, \ldots, a_m) &= c_{k+1} \cdots c_l \\ &= C^*(b_2, \ldots, b_n) \end{aligned}$$

By induction, we have $m = n$ and $a_i = b_i$ for all $i \leq n$. Thus, any two distinct sequences must map to two distinct codewords, implying that extension of C is non-singular. ■

Is the prefix-free requirement too strong for unique decodability? There are certainly uniquely decodable codes that are not prefix-free. For instance, reversing the codewords in C_3 lead to the suffix-free code that is still uniquely decodable because its extended code consists of the reversed codewords of the extended prefix-free code that is non-singular. For example,

$$\begin{aligned} C_4(1) &= 0 \\ C_4(2) &= 10 \\ C_4(3) &= 110 \\ C_4(4) &= 111 \end{aligned}$$

is a uniquely decodable code that is suffix-free but not prefix-free. Can a suffix-free code be prefix-free? Indeed, it can be easily verified that all fixed length non-singular codes are both suffix-free and prefix-free. Can uniquely decodable code be neither suffix-free nor prefix-free? It's possible. For example,

$$C_4(1) = 01$$
$$C_4(2) = 10$$
$$C_4(3) = 011$$
$$C_4(4) = 110$$

is uniquely decodable but $C_4(1)$ is a prefix of $C_4(3)$ and $C_4(2)$ is a suffix of $C_4(4)$. The proof of unique decodability for a code that is neither prefix-free nor suffix-free can be quite tricky. Should we concern ourselves with this type of codes? Does it gain us anything compared to the prefix-free code? We will see in the next lecture that prefix-free condition does not further increase the expected length compared to the unique decodability condition, and unique decodability does not increase the expected length by a significant amount compared to the non-singlarity condition.