ECE 598 - Fall 2025

Lecture 12: Introduction to Data Compression

7th October, 2025

Instructor: Shubhanshu Shekhar

With this lecture, we begin our discussion of a new set of topics related to compression, gambling, and portfolio optimization. As we will see, these seemingly disparate tasks essentially involve the same challenge of assigning accurate probabilities to future outcomes.

1 Variable-Rate Data Compression

We begin by introducing the problem of variable rate data compression (or source coding). Let $X_1, X_2, \ldots, X_n$ denote n i.i.d. random variables drawn from a distribution P_X, taking values in a finite alphabet $\mathcal{X}$.

Definition 1.1. A variable-rate zero-error coding system with parameters (n, R_n) consists of an encoder-decoder pair (e_n, f_n), with $e_n : \mathcal{X}^n \to \{0,1\}^*$, and $f_n : \{0,1\}^* \to \mathcal{X}^n$, with $\{0,1\}^* = \cup_{k=1}^{\infty}\{0,1\}^k$. Each block of symbols x^n is mapped to a codeword $e_n(x^n)$ of length $\ell(x^n) = |e_n(x^n)|$, and the average codeword length (L) and the rate (R_n) of this coding system is

$$L(e_n) = \mathbb{E}[\ell(X^n)], \quad \text{and} \quad R_n = \frac{L(e_n)}{n}.$$

The probability of error of this coding system is $\mathbb{P}\left(f_n \circ e_n(X^n) \neq X^n\right)$, which is required to be exactly equal to 0. Essentially, this is equivalent to the requirement that $f_n \circ e_n$ is the identity map (with probability 1).

In practice, we may work with sequences of blocks of length n each, say $(X^n, X_{n+1}^{2n}, \ldots, X_{kn+1}^{(k+1)n}, \ldots$, which are encoded into sequences $e_n(X^n), e_n(X_{n+1}^{2n}, \ldots, e_n(X_{kn+1}^{(k+1)n}, \ldots$. Unlike the input sequences which have constant block-lengths (equal to n each), the encoded blocks can have different lengths depending on the realizations. This introduces a new challenge at the decoder, as it is not obvious where a particular codeword ends.

Example 1.2. $n = 1$, $\mathcal{X} = \{A, B, C, D\}$, *and let* $e(A) = 0$, $e(B) = 0$, $e(C) = 1$, $e(D) = 1$. *This is an example of a singular code: a code whose encoding function* e *is not one-to-one.*

Clearly such codes cannot achieve zero error, and thus, we will restrict our attention to non-singular codes.

Example 1.3. $n = 1$, $\mathcal{X} = \{A, B, C, D\}$, *and let* $e(A) = 0$, $e(B) = 010$, $e(C) = 01$, $e(D) = 10$. *While this is a nonsingular code, it leads to another kind of ambiguity; it is not uniquely decodable.*

Definition 1.4 (Extensions and Unique Decodability)**.** Given a code with $n = 1$, we define its extension e^*; that maps $\mathcal{X}^*$ to $\{0,1\}^*$ by concatenation. That is, $e^*(x_1, x_2, \ldots, x_m) = (e(x_1), e(x_2), \ldots, e(x_m))$.

A code with encoder e is said to be uniquely decodable if e^* is non-singular.

Example 1.5. *Uniquely decodable:* $e_3(A) = 10$*,* $e_3(B) = 00$*,* $e_3(C) = 11$*,* $e_3(D) = 110$*.*

This is however not instantaneously decodable. For example, suppose we receive a sequence of bits

$$11000000011.$$

If we were to decode it greedily, we could split it into $C, B, B, \{011\}?$*, and we run into an error. The correct decoding of this string is* D, B, B, B, C*. Thus, for such codes, we have to wait till the end of the string before we can start decoding. In fact, we can construct examples, where we may have to wait arbitrarily long before we can decode the bit sequence.*

We want to construct codes for which we don't have to wait till the end of the string to start decoding — we want the code to be instantaneously decodable.

Example 1.6. $e_4(A) = 0$, $e_4(B) = 10$, $e_4(C) = 110$, $e_4(D) = 111$. *This code has the special property that no codeword is the prefix of another (contrast this with the third coding system).*

Definition 1.7. A code is said to be prefix-free or instantaneous if no codeword is a prefix of another.

Claim 1.8. *Prefix-free codes are uniquely decodable.*

Proof. If a code is not UD, then there must exist a string of bits that can be decoded in two ways. Given $a^m \in \{0,1\}^*$, suppose it can be decoded into $A_1, \dots, A_k$ and $B_1, \dots, B_j$. Let r denote the first index at which $A_r \neq B_r$. Then, the either B_r or A_r codewords should be a prefix of the other, which is a contradiction. □

2 Kraft's Inequality

Theorem 2.1. *Let* $e : \mathcal{X} \to \{0,1\}^*$ *denote a prefix-free code with lengths* $\ell_i = \ell(x_i)$ *for* $x_i \in \mathcal{X}$*. Suppose* $|\mathcal{X}| = m$*. Then, the following must be true:*

$$\sum_{i=1}^{m} 2^{-\ell_i} \leq 1.$$

Remark 2.2. Kraft's inequality is a converse statement analogous to Fano's inequality in estimation: ideally we want to construct prefix-free codes with small ℓ_i values, and Kraft's inequality tells us that if we want a prefix free code, we cannot make $\{\ell_i : i \in [m]\}$ small without limit. The lengths must be large enough to satisfy the above inequality.

First proof of Kraft's inequality. We utilize a connection between variable length codes and tree codes (where each codeword corresponds to a leaf on a binary tree). Let $\ell_{\max} = \max_{x_i \in \mathcal{X}} \ell(x_i)$ denote the maximum of the (finitely many) codeword lengths. Then, for each codeword of length $\ell_i \equiv \ell(x_i)$, we have removed $2^{\ell_{\max} - \ell_i}$ possible leaves at the level $\ell_{\max}$. Since the total number of leaves in a complete binary tree at level $\ell_{\max}$ is $2^{\ell_{\max}}$, the following must be true

$$\sum_{i=1}^{m} 2^{\ell_{\max} - \ell_i} \leq 2^{\ell_{\max}}.$$

Dividing both sides by $2^{\ell_{\max}}$ gives us the required inequality.

Another proof of Kraft's inequality. Another useful approach to proving this result is to map each codeword to a number in $[0,1)$. In particular,

- Let $e(x_i) = (a_1^{(i)}, \ldots, a_{\ell_i}^{(i)})$ for $a_j^{(i)} \in \{0,1\}$. Then, to each $e(x_i)$, assign a real number $A_i := 0.a_1^{(i)}a_2^{(i)}\ldots a_{\ell_i}^{(i)} = \sum_{j=1}^{\ell_i} a_j^{(i)} \times 10^{-j} \in [0,1)$.
- Interestingly, the prefix-free property implies that no other A_j can lie in the interval $[A_i, A_i + 2^{-\ell_i})$.
- Since all these intervals are contained in $[0,1)$, the sum of their lengths must be no larger than 1; that is,

$$\sum_{i=1}^{m} 2^{-\ell_i} \leq 1.$$

Remark 2.3. The other direction is also true: given any integer valued ℓ_i satisfying Kraft's inequality, we can construct a prefix-free code with those lengths. Hence, there is an exact one-to-one map from prefix-free codes and (integer-valued) lengths ℓ_i satisfying Kraft's inequality.

Remark 2.4. Surprisingly, we can also show that any uniquely decodable code must also satisfy Kraft's inequality!

3 Performance Limit of Variable Rate Source Codes

Definition 3.1. A rate $R \geq 0$ is said to be achievable with prefix-free codes if for all $\epsilon > 0$, there exists an $n(\epsilon)$, such that for all $n \geq n(\epsilon)$, there exists a variable-rate prefix-free code (e_n, f_n) with $\mathbb{E}[\ell(X^n)] \leq n(R+\epsilon)$, and $P_e = \mathbb{P}(X^n \neq e(f(X^n))) = 0$.

In other words, a rate R is said to be achievable if there exists a sequence of prefix-free variable rate codes of parameter n, such that $\lim_{n\to\infty} \mathbb{E}[\ell(X^n)]/n = R$, and $\mathbb{P}(X^n \neq f(e(X^n))) = 0$ for all n.

Theorem 3.2. *Let $R_s := \inf\{R : R \text{ is achievable}\}$ denote the minimum rate of variable rate source coding. Then, we have $R_s = H(X)$.*

Converse ($\boldsymbol{R_s \geq H(X)}$). Let a rate R be achievable; that is, there exists a sequence of zero-error codes with limiting rate equal to R.

For an $\epsilon > 0$, we have

$$\begin{aligned} n(R+\epsilon) > \mathbb{E}[\ell(X^n)] &= \mathbb{E}[\ell(X^n)] - nH(X) + nH(X) \\ &= \sum_{x^n} \boldsymbol{p}_{X^n}(x^n)\ell(x^n) + n\sum_{x^n} \boldsymbol{p}_{X^n}(x^n)\log \boldsymbol{p}_{X^n}(x^n) + nH(X) \\ &= -\sum_{x^n} \boldsymbol{p}_{X^n}(x^n)\log 2^{-\ell(x^n)} + n\sum_{x^n} \boldsymbol{p}_{X^n}(x^n)\log \boldsymbol{p}_{X^n}(x^n) + nH(X). \end{aligned}$$

Let us introduce $A = \sum_{x^n} 2^{-\ell(x^n)}$, and define $\boldsymbol{q}_{X^n}(x^n) = 2^{-\ell(x^n)}/A$, and we get

$$n(R+\epsilon) > \log(1/A) + \sum_{x^n} \boldsymbol{p}_{X^n}(x^n) \log\left(\frac{\boldsymbol{p}_{X^n}(x^n)}{\boldsymbol{q}_{x^n}(x^n)}\right) + nH(X).$$

By Kraft's inequality, we know that $A \leq 1$, and thus $\log(1/A) \geq 0$. Similarly, due to the nonnegativity of relative entropy, $D_{\mathrm{KL}}(\boldsymbol{p}_{X^n} \parallel \boldsymbol{q}_{X^n}) \geq 0$. Hence, we have proved that

$$n(R+\epsilon) > nH(X), \quad \text{or} \quad H(X) \leq R.$$

Achievability ($\boldsymbol{R_s \leq H(X) + \epsilon}$). We can prove this by using an argument based on "typical sets" constructed using the Law of large numbers for i.i.d. sources (**?**, Chapter 3).

We can also use Kraft's inequality to directly obtain upper and lower bounds on the lengths of prefix-free codes. For instance, suppose $e : \mathcal{X} \to \{0,1\}^*$ is a prefix-free encoder with lengths $\{\ell_i : 1 \leq i \leq |\mathcal{X}| = m\}$, and $L = \sum_{i=1}^m p_i \ell_i$. Then, we must have

$$L \geq H(X) = -\sum_i p_i \log p_i.$$

To see why this is true, observe that $A = \sum_i 2^{-\ell_i} \leq 1$ by Kraft's inequality and define Q with $Q(\{x_i\}) = q_i = 2^{-\ell_i}/A$. Hence, we have

$$\begin{aligned} 0 \leq D_{\mathrm{KL}}(P_X \parallel Q) &= \sum_i p_i \log(p_i/q_i) = \sum_i p_i \log p_i + \log A - \sum_i p_i \log q_i \\ &= -H(P_X) + \log A + \sum_i p_i \ell_i = -H(P_X) + \log A + L. \end{aligned}$$

Since $A \leq 1$, we have $\log A \leq 0$, which leads to the required $L \geq H(X) \equiv H(P_X)$.

To see the other direction, let us consider the optimization problem

$$L^* := \min_{\ell_i \in \mathbb{N}} \sum_i p_i \ell_i, \quad \text{subject to} \quad \sum_i 2^{-\ell_i} \leq 1.$$

Now, let us define $\ell_i = \lceil \log(1/p_i) \rceil \leq \log(1/p_i) + 1$. It is easy to verify that it satisfies Kraft's inequality:

$$\sum_i 2^{-\ell_i} = \sum_i 2^{-\lceil \log 1/p_i \rceil} \leq \sum_i 2^{-\log p_i} = \sum_i p_i = 1.$$

Furthermore, the expected length satisfies

$$L^* \leq L = \sum_i p_i \ell_i \leq \sum_i p_i \left(\log 1/p_i + 1\right) \leq H(X) + 1.$$

Hence, there exist feasible codeword lengths which are within 1 bit of the optimal.

Remark 3.3. The above discussion also indicates how to construct near-optimal codes given P_X: we simply look at the midpoints of the jumps in the CDF ($y_i = (F_X(x_{i-1}) + F_X(x_i))/2$ and assign codewords that are the first $\lceil \log(1/p_i) \rceil + 1$ bits of the binary representation of y_i. By design, this results in a prefix-free code that is within 2 bits of the entropy, and this gap can be made arbitrarily small by working with longer blocks.

This also highlights an equivalence that is key to our subsequent discussions: variable-rate coding is essentially equivalent to assigning probabilities to the elements of the alphabet. If we know p_i, then $\ell_i^* \approx \log(1/p_i)$, and furthermore, each near-optimal code ℓ_i^* defines a probability distribution $q_i \approx 2^{-\ell_i^*}$.

Remark 3.4. The equivalence between coding and assigning probabilities also leads to an operational characterization of relative entropy as the cost of misspecification; that is, suppose $X \sim P$, but we wrongly assume that $X \sim Q$ for some $Q \neq P$. Our constructed codewords will have lengths $\ell_i \approx \log(1/q_i)$ instead of $\ell_i^* \approx \log(1/p_i)$. Thus, the *redundancy* or extra bits on average used due to this misspecification is equal to

$$\begin{aligned} \mathrm{Red}(P, Q) = L - L^*(P_X) &= \sum_i p_i \log(1/q_i) - \sum_i p_i \log(1/p_i) = \sum_i p_i \log\left(\frac{p_i}{q_i}\right) \\ &= D_{\mathrm{KL}}(P \parallel Q). \end{aligned}$$

In a couple of lectures, we will consider a "compression game" in which the goal would be to sequentially assign probabilities (or equivalently assign codewords) to symbols in order to minimize the redundancy w.r.t. the best distribution from a given class of distributions in hindsight.