Electrical Engineering 229A Lecture 9 Notes

Daniel Raban

September 23, 2021

1 Uniquely Decodable Codes

1.1 Uniquely decodable and prefix-free codes

Last time, we talked about lossless data compression.

Definition 1.1. A **lossless data compression scheme** is a sequence $((e_n, d_n), n \geq 1)$ of maps $e_n : \mathscr{X}^n \to \{0,1\}^* \setminus \{\varnothing\}$ and $d_n : \{0,1\}^* \setminus \{\varnothing\} \to \mathscr{X}^n$ such that $d_n \circ e_n$ is the identity for each n.

Equivalently, we could specify $(e_n, n \geq 1)$ and insist that each e_n is one to one. Equivalently, we could specify $e_* : \mathscr{X}^* \setminus \{\varnothing\} \to \{0,1\}^* \setminus \varnothing$ which is one to one on each $\mathscr{X}^n$.

One practical way to create encoding maps is to specify $e : \mathscr{X} \to \{0,1\}^* \setminus \{\varnothing\}$ and define $e_n(x_1^n) = e(x_1)e(x_2)\cdots e(x_n)$.

Example 1.1. If $\mathscr{X} = \{1,2,3\}$ with

$$e(1) = 0, \qquad e(2) = 11, \qquad e(3) = 0110,$$

then we can encode larger words like

$$e(12) = 011.$$

We could also do this by specifying e on blocks of length r for some $r \geq 1$.

Definition 1.2. We'll say e is **uniquely decodable** if each e_n is one to one.

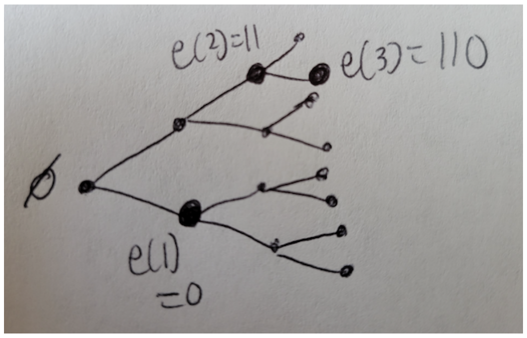
This is equivalent to requiring that e_* is one to one. One way to get unique decodability is if e is *instantaneous* or *prefix-free*.

Definition 1.3. The encoding map e is **instantaneous** or **prefix-free**[1] if for all $x \neq y \in \mathscr{X}$, $e(x)$ is not a prefix of $e(y)$.

[1] Cover and Thomas call this a "prefix code," which is super confusing.

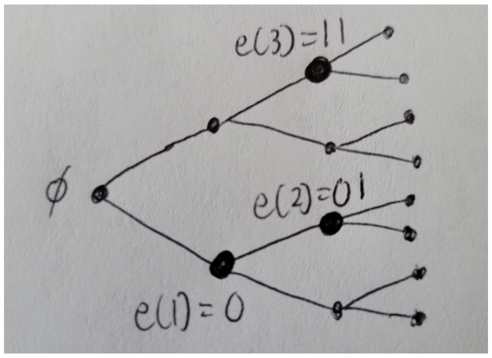
It's easiest to think about this in terms of a binary tree. Prefix-free is equivalent to the requirement that no node $e(x)$ in the coding can lie on the path between the root and another node $e(y)$.

Example 1.2. The previous example is *not* prefix-free.



Example 1.3. The following example is uniquely decodable but not prefix-free:

$$e(1) = 0, \qquad e(2) = 01, \qquad e(3) = 11.$$



If the first received bit is 1 and the next bit is 1, we can parse 11 from the received sequence. If the first received bit is 0, then if the next bit is 0, we can parse off the first 0; if the next bit is 1, we need to wait to figure out the parity of the run of 1s.

1.2 Kraft's inequality

Theorem 1.1 (Kraft's inequality). *For any prefix-free binary code e,*

$$\sum_{x \in \mathscr{X}} 2^{-\ell(e(x))} \leq 1.$$

Proof. A formal proof is in Cover and Thomas. The idea is to add the weights of the $e(x)$, where a node at depth d gets weight 2^{-d}. □

Remark 1.1. There is a version of this for prefix-free D-ary codes $e : \mathscr{X} \to \{0, \dots, D-1\}$. In this case, Kraft's inequality says

$$\sum_{x \in \mathscr{X}} D^{-\ell(e(x))} \leq 1.$$

Here is a generalization by McMillan.

Theorem 1.2. *For every uniquely decodable code* $e : \mathcal{X} \to \{0,1\}^* \setminus \{\varnothing\}$,

$$\sum_{x \in \mathscr{X}} 2^{-\ell(e(x))} \leq 1.$$

Remark 1.2. There is also a version of this for uniquely decodable D-ary codes $e : \mathscr{X} \to \{0, \dots, D-1\}$. In this case, Kraft's inequality says

$$\sum_{x \in \mathscr{X}} D^{-\ell(e(x))} \leq 1.$$

Let's prove the D-ary version:

Proof. We use a generating function technique. Consider the expression

$$\sum_{x_k \in \mathscr{X}} D^{-\ell(e(x_k))}$$

at time k. Then

$$\sum_{x_1^n \in \mathscr{X}^n} D^{-\ell(e_n(x_1^n))} = \sum_{x_1^n \in \mathscr{X}^n} \prod_{k=1}^n D^{-\ell(e(x_k))} = \prod_{k=1}^n \sum_{x_k \in \mathscr{X}} D^{-\ell(e(x_k))}.$$

Take n-th roots on both sides. The key observations are that on the left hand side,

1. The total number of terms that provide D^{-m} for any $m \geq 1$ is at most D^m (by unique decodability).
2. The largest m for which D^{-m} shows up in the left hand side is $n\ell_{\max}$, where $\ell_{\max} = \max_x \ell(e(x))$.

So this tells us that the left hand side is $\leq n\ell_{\max}$. Now observe that

$$(n\ell_{\max})^{1/n} = e^{\frac{1}{n}(\log n + \log \ell_{\max})} \xrightarrow{n \to \infty} 1. \qquad \square$$

1.3 Optimal compression as a linear programming problem

To optimize compression (in bits/symbol) for an iid source ($\mathscr{X}$-valued, marginal distribution $(p(x), x \in \mathscr{X})$), we want to solve

$$\text{minimize: } \sum_{x \in \mathscr{X}} p(x)\ell(e(x))$$

$$\text{subject to: } e \text{ is prefix-free.}$$

This suggests studying the problem

$$\text{minimize: } \sum_{x \in \mathscr{X}} p(x)\ell(e(x))$$

$$\text{subject to: } \sum_{x \in \mathscr{X}} 2^{-\ell(e(x))} \le 1, \quad \ell(e(x)) \ge 1.$$

This, although looking like a weakening of the constraint, actually is equivalent because for every collection of lengths satisfying Kraft's inequality, there is a prefix-free code with those lengths.

Proposition 1.1. *Given any $(\ell(x), x \in \mathscr{X})$ with $\ell(x) \ge 1$ and $\sum_{x \in \mathscr{X}} 2^{-\ell(x)} = 1$, there exists a prefix-free $e : \mathscr{X} \to \{0,1\}^* \setminus \{\varnothing\}$ such that for all x, $\ell(e(x)) = \ell(x)$.*

Proof. Proceed by induction on $|\mathscr{X}|$. Merge the two smallest values $2^{-\ell(x)}$ to reduce the size of the alphabet by 1. □

This new linear program is an integer programming problem because the variables $\ell(x)$ are required. This is computationally difficult. We can relax this to get a tractable problem

$$\text{minimize: } \sum_{x \in \mathscr{X}} p(x)\ell(e(x))$$

$$\text{subject to: } \sum_{x \in \mathscr{X}} 2^{-\ell(e(x))} \le 1,$$

where the $\ell(x)$ can be real-valued. The second condition implies $\ell(x) \ge 0$. We can also replace the inequality by an equality because this only improves the objective.

We can solve this using Lagrange multipliers. Consider the Lagrangian

$$\sum_x p(x)\ell(x) + \lambda \left(\sum_x 2^{-\ell(x)} - 1 \right).$$

Differentiate in each $\ell(x)$, and set the derivative equal to 0 to get

$$p(x) - \lambda \log_e 2 \cdot 2^{-\ell(x)} = 0 \quad \forall x \in \mathscr{X}.$$

This requires

$$\ell(x) = -\log p(x) + k$$

for all $x \in \mathscr{X}$. The condition $\sum_x 2^{-\ell(x)} = 1$ gives us $k = 0$. So the optimal value of the objective is

$$-\sum_{x \in \mathscr{X}} p(x) \log p(x) = H(p).$$

We have just proven the following:

Theorem 1.3. *The expected length of a uniquely decodable binary code is* $\geq H(p)$.

Remark 1.3. Taking $\ell(x) = \lceil \log \frac{1}{p(x)} \rceil$ for $x \in \mathscr{X}$, we have

$$\sum_{x \in \mathscr{X}} 2^{-\ell(x)} \leq 1.$$

Hence, there is a prefix code with these lengths. Such a code is called a **Shannon code**. Its expected length is

$$\sum_x p(x) \left\lceil \log \frac{1}{p(x)} \right\rceil \leq H(p) + 1.$$

For a stationary process working at block length n, we can get lossless compression with bits/symbol at most

$$\frac{1}{n} H(X_1, \ldots, X_n) + \frac{1}{n}.$$

We will discuss this next time.