

Codes

Jonathan L.F. King
 University of Florida, Gainesville FL 32611-2082, USA
 squash@ufl.edu
 Webpage <http://squash.1gainesville.com/>
 23 September, 2017 (at 15:50)

(Folks, help me proofread and correct this.)

§An Overview

Formal languages	1
Codes	1
Trees	2
Inequalities	2
Kraft-McMillan Inequality	2
Completeness Lemma	4
K-M Completeness corollary	4
Expected coding-length	4
Codemap	4
ECL	4
MECL	5
Huffman codes	5
HC-same-ECL Thm	5
Induction step	6
Depth Lemma	6
Huffman's theorem	6
Induction step	6
Entropy/Distropy	7
Jensen's inequality	7
Distropy UD-code Inequality	7
Sardinas-Patterson Algorithm	7
Decoding-delay for UD-codes	7

Index for “JK Codes notes” 8

Formal languages

Use \emptyset for the empty set. An *alphabet* \mathbf{G} is a non-empty set, whose members are called “letters”; usually $2 \leq |\mathbf{G}| < \infty$. A *word* (over an alphabet \mathbf{G}) is a *finite* string of letters; Use \mathbf{G}^* for the set of all words, and use ε for the *nullword*, the unique length-zero word. E.g if $\mathbf{G} = \{a, b\}$, then \mathbf{G}^* equals

$$\{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \dots\}.$$

Write \mathbf{G}^+ for $\mathbf{G}^* \setminus \{\varepsilon\}$.

Concatenation of words $\mathbf{v}, \mathbf{z} \in \mathbf{G}^*$ is written $\mathbf{v} \triangleright \mathbf{z}$ or just \mathbf{vz} . Thus $\mathbf{cat} \triangleright \mathbf{nip} = \mathbf{catnip}$. So \mathbf{G}^* is a *semigroup under concatenation, with ε the identity element*. Use $\text{Len}(\mathbf{v})$ or $|\mathbf{v}|$ for the *length* of word \mathbf{v} , and have $\mathbf{v} \triangleright 3$ mean that $|\mathbf{v}| > 3$. For n a natnum, let \mathbf{v}^n mean the concatenation $\mathbf{vv} \dots \mathbf{v}$. So $\mathbf{v}^0 = \varepsilon$.

A *language* is a subset $\mathcal{L} \subset \mathbf{G}^*$. Here are six distinct languages:

$$\emptyset = \{\}, \{\varepsilon\}, \{\mathbf{catnip}\}, \{\mathbf{cat}, \mathbf{nip}\}, \{\varepsilon, \mathbf{cat}, \mathbf{nip}\}$$

and $\{\mathbf{bc}, \mathbf{bac}, \mathbf{baac}, \mathbf{baaac}, \dots\} = \{\mathbf{ba}^n \mathbf{c}\}_{n=0}^\infty$.

The first five are finite languages, having cardinalities 0, 1, 1, 2, 3. Call \emptyset the *void language* and call $\{\varepsilon\}$ the *nullword language*. The “concatenation of languages” $\mathcal{K}, \mathcal{L} \subset \mathbf{G}^*$ is

$$\mathcal{KL} = \mathcal{K} \triangleright \mathcal{L} := \{\mathbf{v} \triangleright \mathbf{w} \mid \mathbf{v} \in \mathcal{K} \text{ and } \mathbf{w} \in \mathcal{L}\}.$$

[So $\emptyset \triangleright \mathcal{L} = \emptyset = \mathcal{L} \triangleright \emptyset$ and $\{\varepsilon\} \triangleright \mathcal{L} = \mathcal{L} = \mathcal{L} \triangleright \{\varepsilon\}$.] Let \mathcal{L}^n mean $\mathcal{L} \mathcal{L} \dots \mathcal{L}$. Hence $\mathcal{L}^0 = \{\varepsilon\}$, since the nullword language is the identity element for language-concatenation.^{♥1}

Define the *Kleene star* operator by

$$\mathcal{L}^* := \bigcup_{n=0}^\infty \mathcal{L}^n.$$

In particular $\emptyset^* = \{\varepsilon\} = \{\varepsilon\}^*$. Similarly, the *Kleene plus* operator is

$$\mathcal{L}^+ := \bigcup_{n=1}^\infty \mathcal{L}^n.$$

Hence $[\varepsilon \in \mathcal{L}^+] \Leftrightarrow [\varepsilon \in \mathcal{L}] \Leftrightarrow [\mathcal{L}^+ = \mathcal{L}^*]$. Each Kleene op is *idempotent*: $[\mathcal{L}^*]^* = \mathcal{L}^*$ and $[\mathcal{L}^+]^+ = \mathcal{L}^+$.

Prefix/Suffix. For words, say “ \mathbf{v} is a *prefix* of \mathbf{w} ” if there exists a word \mathbf{z} with $\mathbf{vz} = \mathbf{w}$; write $\mathbf{v} \preceq \mathbf{w}$ for this relation. If, also, $\mathbf{v} \neq \mathbf{w}$, then \mathbf{v} is a *proper prefix* of \mathbf{w} , written $\mathbf{v} \prec \mathbf{w}$.

If $\exists \mathbf{z} \in \mathbf{G}^*$ with $\mathbf{zv} = \mathbf{w}$, then “ \mathbf{v} is a *suffix* of \mathbf{w} ”. [However, we have no special symbol for the relation.]

Codes

For the time being, a *code* \mathcal{C} means a non-void subset $\mathcal{C} \subset \mathbf{G}^+$; usually $2 \leq |\mathcal{C}| < \infty$. [Occasionally it is convenient to consider collections \mathcal{C} which *might* own ε . So if all we know is that $\mathcal{C} \subset \mathbf{G}^*$, then we call \mathcal{C} a *nullishcode*. If we can later on prove that $\mathcal{C} \not\ni \varepsilon$, then we’ll have shown \mathcal{C} to be a code.]

Call \mathcal{C} a *block code* if all its codewords have the same length. E.g, $\{\mathbf{FBI}, \mathbf{CIA}\}$ is a blockcode, whereas $\{\mathbf{Go}, \mathbf{Gators}\}$ is *not* a blockcode, –although it *is* (see below) a prefixcode. [Caveat: “block code” is used with

^{♥1}Aside: We already knew that $(\mathbf{G}^*, \triangleright, \varepsilon)$ is a [non-commutative] semigroup. And letting $\mathbb{L} = \mathbb{L}_{\mathbf{G}}$ denote the set of all languages over \mathbf{G} , this generalizes to a [non-commutative] semigroup

$$(\mathbb{L}, \triangleright, \{\varepsilon\}).$$

Of course, we also have the two commutative semigroups $(\mathbb{L}, \cup, \emptyset)$ and $(\mathbb{L}, \cap, \mathbf{G}^*)$.

slightly different meanings in the literature. Perhaps *constant-length code* is a more accurate term.]

A code \mathcal{C} is *uniquely decodable* (a *UD-code*) if each code-message $\mathbf{z} \in \mathcal{C}^*$ has a unique decomposition w.r.t \mathcal{C} . That is, if words $\mathbf{v}_j, \mathbf{w}_k \in \mathcal{C}$ satisfy

$$1.1: \quad \text{If } \mathbf{v}_1\mathbf{v}_2 \dots \mathbf{v}_J = \mathbf{z} = \mathbf{w}_1\mathbf{w}_2 \dots \mathbf{w}_K \\ \text{then } J = K \text{ and } \forall i: \mathbf{v}_i = \mathbf{w}_i.$$

A *prefix code* \mathcal{C} (more accurately called a “prefix-free code”) has no codeword being a proper prefix of another. Prefix-codes are UD-codes since, stronger than (1.1), they have the *RI-UD property* (the “right-infinite-UD property”) that

$$1.2: \quad \left[\begin{array}{l} \mathbf{v}_1\mathbf{v}_2\mathbf{v}_3 \dots = \mathbf{w}_1\mathbf{w}_2\mathbf{w}_3 \dots \\ \text{with each } \mathbf{v}_j, \mathbf{w}_k \in \mathcal{C} \end{array} \right] \Rightarrow \left[\begin{array}{l} \forall i \in \mathbb{Z}_+: \\ \mathbf{v}_i = \mathbf{w}_i \end{array} \right].$$

We have these non-reversible implications

$$1.2': \quad \text{Block} \Rightarrow \text{Prefixcode} \xrightarrow{*1} \text{RI-UD} \xrightarrow{*2} \text{UD}.$$

A code showing (*2) non-reversible has these words

$$1.2'': \quad \mathbf{v} := \mathbf{b}, \mathbf{w} := \mathbf{ba}, \mathbf{z} := \mathbf{aa}.$$

It is uniquely decodable ([Exer. E1](#)), yet fails (1.2), since $\mathbf{vzzz} \dots$ equals $\mathbf{wzzz} \dots$. Finally, that (*1) is non-reversible will be shown by (1.5), the “Chris code”.

A *suffix code* (no codeword is a proper suffix of another) is automatically a UD-code. Dually to (1.2') we have non-reversible implications

$$1.3': \quad \text{Block} \Rightarrow \text{Suffixcode} \Rightarrow \text{LI-UD} \Rightarrow \text{UD},$$

where a *left-infinite-UD-code* (a *LI-UD-code*) satisfies

$$1.3: \quad \left[\begin{array}{l} \dots \mathbf{v}_{-2}\mathbf{v}_{-1} = \dots \mathbf{w}_{-2}\mathbf{w}_{-1} \\ \text{with each } \mathbf{v}_j, \mathbf{w}_k \in \mathcal{C} \end{array} \right] \Rightarrow \left[\begin{array}{l} \forall i \in \mathbb{Z}_-: \\ \mathbf{v}_i = \mathbf{w}_i \end{array} \right].$$

Note (???) is an example of a suffixcode which is not a prefixcode.

Bi-infinite. A bi- ∞ \mathbf{G} -string σ can be viewed as a map $\sigma: \mathbb{Z} \rightarrow \mathbf{G}$. A *\mathcal{C} -parsing* of σ is a sequence

$$\dots < k_{-2} < k_{-1} < k_0 < k_1 < k_2 < k_3 < \dots$$

of integers st. each substring $\sigma|_{[k_\ell \dots k_{\ell+1}]}$ is a codeword, that is, lies \mathcal{C} . Write sequence $(k_\ell)_{\ell \in \mathbb{Z}}$ as $\vec{\mathbf{k}}$.

Say that \mathcal{C} has the *bi-infinite-UD property* (is *BI-UD*) if

1.4: Each bi- ∞ string σ which has a \mathcal{C} -parsing, has only one \mathcal{C} -parsing. I.e, with $\vec{\mathbf{j}}$ and $\vec{\mathbf{k}}$ two \mathcal{C} -parsings of σ , then the sets $\{j_i\}_{i \in \mathbb{Z}}$ and $\{k_\ell\}_{\ell \in \mathbb{Z}}$ are equal.

Slightly weaker, consider two parsings $\vec{\mathbf{j}}$ and $\vec{\mathbf{k}}$, and let $\mathbf{v}_\ell := \sigma|_{[j_\ell \dots j_{\ell+1}]}$ and $\mathbf{w}_\ell := \sigma|_{[k_\ell \dots k_{\ell+1}]}$. The *weak-BI-UD* property asserts

For each σ and parsings as above, there exists a translation $T \in \mathbb{Z}$ so that:

$$1.4^{\text{weak}}: \quad \forall \ell \in \mathbb{Z}: \mathbf{v}_{\ell+T} = \mathbf{w}_\ell.$$

(I.e, one parsing may be a shift of the other, but the codeword sequences are the same.)

Immediately,

$$1.4': \quad \text{BI-UD} \xrightarrow{*3} \text{weak-BI-UD} \xrightarrow{*4} \left[\begin{array}{l} \text{Both LI-UD} \\ \text{and RI-UD} \end{array} \right].$$

The code $\{\mathbf{bbb}\}$ produces $\sigma := \dots \mathbf{bbb} \dots$, which is its only bi- ∞ string. This σ has *three* parsings, since the cutpoints $\vec{\mathbf{j}}$ can all be mod-3 congruent to -1 or 0 or 1. Yet each parsing yields the *same* codeword sequence, namely $\dots \boxed{\mathbf{bbb}} \boxed{\mathbf{bbb}} \boxed{\mathbf{bbb}} \dots$. Hence (*3) is *not reversible*.

The “Pirate code” $\{\mathbf{OH}, \mathbf{HO}\}$ is trivially LI-UD and RI-UD, since it is a blockcode. Yet the Pirate code admits bi- ∞ string $\dots \mathbf{HOHOHOHO} \dots$, which can be parsed as $\dots \boxed{\mathbf{OH}} \boxed{\mathbf{OH}} \boxed{\mathbf{OH}} \dots$ or as $\dots \boxed{\mathbf{HO}} \boxed{\mathbf{HO}} \boxed{\mathbf{HO}} \dots$, two different codeword sequences. Yup; (*4) *ain't reversible either*.

The “Chris code” (evidently a cry for help)

$$1.5: \quad \{\mathbf{S}, \mathbf{SOS}\}$$

is BI-UD, since each occurrence of “O” must lie in $\boxed{\mathbf{SOS}}$, and every other codeword must be $\boxed{\mathbf{S}}$. Not being a prefixcode, (1.5) proves (*1) not reversible.

Trees. Here, a (rooted) *tree* is a set T of nodes, equipped with two operators: $\text{Root}(T)$ is the root-node of T . For each node $v \in T$, let $\text{Kids}(v)$ be the *set* of children of v . A node w is a *leaf-node* if: The set $\text{Kids}(w)$ is empty. A tree has the property that, from the root-node, one can get to an arbitrary node, by applying the $\text{Kids}(\cdot)$ operator finitely-many times.

Trees T and S are (*tree-*)*isomorphic* if *there exists* a bijection $f: T \rightarrow S$ such that:

TI1: $f(\text{Root}(T)) = \text{Root}(S)$.

TI2: For each $v \in T$:

$$\{f(k) \mid k \in \text{Kids}(v)\} = \text{Kids}(f(v)).$$

For a $\Gamma \in \mathbb{Z}_+$, a tree is Γ -**bounded** if each node has at most Γ many children. The tree is Γ -**full** if every node is either has *no* children [is a **leaf-node**], or has precisely Γ many children; otherwise, the tree is Γ -**deficient**.

Inequalities

Kraft proved (2a) for *prefix-codes*, as well as its converse, (2b). McMillan strengthened (2a) to UD-codes.

2: Kraft-McMillan Inequality. Consider a countable code \mathcal{C} over finite alphabet \mathbf{G} . If \mathcal{C} is a UD-code then

$$2a: \quad \sum_{\mathbf{v} \in \mathcal{C}} 1/\Gamma^{\text{Len}(\mathbf{v})} \leq 1,$$

where Γ is the number of letters in \mathbf{G} .

Conversely, consider posints $\vec{\ell} = (\ell_1, \ell_2, \dots, \ell_R)$.

2b: If $\sum_{j=1}^R 1/\Gamma^{\ell_j} \leq 1$ then there exists a prefix \mathbf{G} -code $\mathcal{C} = (\mathbf{v}_1, \dots, \mathbf{v}_R)$ with each $\text{Len}(\mathbf{v}_j) = \ell_j$.

[The also result holds for *infinite* tuples $\vec{\ell} = (\ell_1, \ell_2, \ell_3, \dots)$ that satisfy $[\sum_{j=1}^{\infty} 1/\Gamma^{\ell_j}] \leq 1$.] \diamond

Exer. E2. Give an example of a code, \mathcal{X} , that violates (2a). [So \mathcal{X} must fail to be UD.] \square

Defn. A code \mathcal{C} is **weakly-UD** if the following holds. For each posint N and words $\mathbf{v}_i, \mathbf{w}_i \in \mathcal{C}$:

1.1': If $\mathbf{v}_1 \mathbf{v}_2 \dots \mathbf{v}_N = \mathbf{w}_1 \mathbf{w}_2 \dots \mathbf{w}_N$
then $\forall i: \mathbf{v}_i = \mathbf{w}_i$.

Contrast this with the (1.1) defn of **UD**. \square

Exer. E3. POSTING RACE: Who can be the first to post a code which is weakly-UD, but not UD? \square

Preliminaries for (2a). The below proof uses $T_{\ell,n}$, the number of length- ℓ words which are concatenations of

n many codewords. E.g, consider a code $\mathcal{C} = \{\mathbf{v}, \mathbf{w}, \mathbf{z}\}$ have lengths 5, 7, 8, respectively.

$$\begin{aligned} T_{15,1} &= |\emptyset| = 0. & T_{15,2} &= |\{\mathbf{wz}, \mathbf{zw}\}|. \\ T_{15,3} &= |\{\mathbf{v}\mathbf{v}\mathbf{v}\}| = 1. & T_{15,4} &= |\emptyset| = 0. \end{aligned}$$

Indeed, $T_{15,n}$ is zero for each $n \geq 4$. As for $T_{15,2}$: If $\mathbf{wz} = \mathbf{zw}$ then $T_{15,2} = 1$, else $T_{15,2} = 2$. \square

Proof of (2a). WLOGenerality, \mathcal{C} is finite. (Exer. E4)

With $\Gamma := |\mathbf{G}|$, our goal is

$$2a': \quad \sum_{\mathbf{v} \in \mathcal{C}} 1/\Gamma^{\text{Len}(\mathbf{v})} \stackrel{?}{\leq} 1.$$

WELOG, suppose the shortest and longest words in \mathcal{C} have lengths 3 and 7. For $n = 1, 2, \dots$, each word in \mathcal{C}^n has a length, ℓ , in $[3n .. 7n]$; let $T_{\ell,n}$ be the number of such words. Certainly $T_{\ell,n} \leq \Gamma^\ell$, the number of *all* length- ℓ words over \mathbf{G} . So the “generating function”

$$F_n(x) := \sum_{\ell=3n}^{7n} T_{\ell,n} \cdot \frac{1}{x^\ell}$$

satisfies, for $x > 0$, that $F_n(x) \leq \sum_{\ell=3n}^{7n} \Gamma^\ell \cdot \frac{1}{x^\ell}$. Thus

$$F_n(\Gamma) \leq \sum_{\ell=3n}^{7n} \Gamma^\ell \cdot \frac{1}{\Gamma^\ell} \stackrel{\text{note}}{=} 1 + 7n - 3n \stackrel{\text{note}}{\leq} 5n,$$

for each posint n .

Using uniqueness. Fix n and an $\ell \in [3n .. 7n]$. The coefficient of $1/x^\ell$ in $[F_1(x)]^n$ is the number of *ways of writing* a length- ℓ word as a concatenation of n many codewords. But, by the UD-hypothesis [actually, only “weakly-UD” is being used], each such length- ℓ word can be written as a codeword-concatenation in *only one* way. Thus

$$[F_1(x)]^n = F_n(x),$$

as rational functions. From above, then,

$$[F_1(\Gamma)]^n \leq 5n.$$

The LhS is exponential in n , whilst the RhS is linear. Thus $F_1(\Gamma) \leq 1$. Finally, observe that $F_1(\Gamma)$ is a rewriting of LhS(2a). \diamond

Proof of (2b). We'll show the idea for $\Gamma = 2$. Arrange the lengths as $\ell_1 \leq \ell_2 \leq \dots \leq \ell_R$. On the full binary-tree of depth $D := \ell_R$, put weight $1/2^D$ on each leaf-node. All the nodes start as **free**; we will iteratively mark some as **busy** as we create words $\mathbf{v}_1, \mathbf{v}_2, \dots$. Call a node **very-free** if it and all its children are **free**, i.e not **busy**.

Let \mathbf{v}_1 be the leftmost path down to depth ℓ_1 ; so $\mathbf{v}_1 = 000 \dots 0$. Mark \mathbf{v}_1 and all its children as **busy**. This action creates busy leaf-nodes of total weight

$$2^{D-\ell_1} \cdot \frac{1}{2^D} \stackrel{\text{note}}{=} 1/2^{\ell_1}.$$

With $d := \ell_1$, note that

*: Each free node at depth $\geq d$ is very-free.

Let \mathbf{v}_2 be the leftmost path to a free node at depth ℓ_2 . [So \mathbf{v}_2 has $\ell_1 - 1$ many 0s, then a 1, then $\ell_2 - \ell_1$ many 0s.] Mark \mathbf{v}_2 and its descendants as **busy**. Now the total weight of busy leaf-nodes is

$$\frac{1}{2^{\ell_1}} + \frac{1}{2^{\ell_2}}.$$

Moreover, with $d := \ell_2$, note (*) holds, since $\ell_2 \geq \ell_1$.

We'd like to continue using depth ℓ_3 , depth ℓ_4, \dots , depth ℓ_k, \dots . The only obstruction at a stage k , is if there is no free node at depth ℓ_k . But the total leaf-weight we've used up so far, is

$$W := \sum_{j=1}^{k-1} 1/2^{\ell_j}.$$

Since this sum is *strictly* less than 1, there exists a free-node at depth ℓ_{k-1} . (Indeed, the number of such free-nodes is precisely $[1 - W]/2^{k-1}$.) Finally, since $\ell_k \geq \ell_{k-1}$, there is certainly a free-node at depth ℓ_k . ♦

2c: Defn. For a Γ -code with lengths $\vec{\ell} = (\ell_1, \dots, \ell_R)$, use

$$\Sigma(\vec{\ell}) := \Sigma_{\Gamma}(\vec{\ell}) := \sum_{j=1}^R 1/\Gamma^{\ell_j}$$

for its **Kraft-sum**. Kraft's thm says –if the code is UD– that $\Sigma(\vec{\ell}) \leq 1$. If equality, then the code (ditto the tuple) is **complete**, otherwise it is **redundant**; more precisely, Γ -**complete** and Γ -**redundant**.

Given tuples $\vec{\ell} = (\ell_1, \dots, \ell_R)$ and $\vec{s} = (s_1, \dots, s_R)$, write $\vec{\ell} \preceq \vec{s}$ if $\boxed{\forall j: \ell_j \leq s_j}$ and write $\vec{\ell} \prec \vec{s}$ if at least one of these inequalities is strict. □

Exer. E4.8. A finite Γ -bounded tree T with R many leaves, yields a length-spectrum $\vec{\ell} = (\ell_1, \dots, \ell_R)$; so terms “ Γ -complete” and “ Γ -redundant” makes sense for the tree. Prove:

2d: Completeness Lemma. A finite Γ -bounded tree, T , is Γ -complete IFF it is Γ -full. ♦

In (2e), below, we consider only *binary* prefix-codes; $\Gamma = 2$.

2e: K-M Completeness corollary. Given a tuple \vec{s} , there exists a complete prefix-code with tuple $\vec{\ell} \preceq \vec{s}$. ♦

Proof. We need but produce a complete $\vec{\ell} \preceq \vec{s}$, since Kraft's thm will hand us a prefix-code with lengths $\vec{\ell}$.

It suffices, given a redundant \vec{s} , to produce an $\vec{\ell} \prec \vec{s}$; after all, there are only finitely-many tuples $\prec \vec{s}$, so iterating will eventually halt, at a complete tuple.

WLOG, $T := s_1$ is a max length in \vec{s} ; so each $1/2^{s_j}$ is a multiple of $1/2^T$, hence so is $\Sigma(\vec{s})$. As \vec{s} is redundant, the gap $1 - \Sigma(\vec{s})$ dominates $1/2^T$. So define $\vec{\ell}$ by $\ell_2 := s_2, \dots, \ell_R := s_R$, and $\ell_1 := s_1 - 1$. ♦

Exer. E5. POSTING RACE: Does (2e) hold for larger alphabet-sizes? If so, how does the proof need to be modified? □

Expected coding-length

A **probability distribution** on a codeword-set \mathcal{C} is a map $P: \mathcal{C} \rightarrow [0, 1]$ st.

$$3a: \quad \sum_{\mathbf{v} \in \mathcal{C}} P(\mathbf{v}) = 1.$$

We will usually discard from the code all probability-zero words. In practice, then, a “probability distribution” is a map $P: \mathcal{C} \rightarrow (0, 1)$ fulfilling (3a)

The **expected^{♥2} coding length** of \mathcal{C} is

$$3b: \quad \text{ECL}(\mathcal{C}) := \sum_{\mathbf{v} \in \mathcal{C}} P(\mathbf{v}) \cdot \text{Len}(\mathbf{v}).$$

E.g, consider code $\mathcal{C} := \{\mathbf{w}_1, \dots, \mathbf{w}_4\}$ where

$$3c: \quad \mathbf{w}_1 := 00, \mathbf{w}_2 := 010, \mathbf{w}_3 := 011, \mathbf{w}_4 := 1,$$

where $P(\mathbf{w}_4) = \frac{1}{2}$, and the other three words have probability $\frac{1}{6}$. Then $\text{ECL}(\mathcal{C})$ is then

$$\frac{1}{2} \cdot 1 + \frac{1}{6} \cdot [2 + 3 + 3] = \frac{11}{6}.$$

^{♥2}“Expected” is what probabilists use for “average”.

Codemap. A *source alphabet* Ω , also called a “message set”, might be

$$\{a, b, \dots, z, \cdot, \text{Space}\},$$

or might be $\{\text{tank}, \text{ship}, \dots, \text{plane}\}$. Fixing a *code-alphabet* \mathbf{G} , a map $f: \Omega \rightarrow \mathbf{G}^+$ is a *codemap* (or *cipher*) if

i: f is injective, and

ii: $\mathcal{C} := \text{Range}(f)$ is a code. [Phrased this way, so that if we change our defn of “code” for a given context, then the defn of *codemap* changes with it.]

Every adjective applying to a code, also applies to a codemap; e.g, “a *block/prefix/UD* codemap”.

ECL. Consider a [finite or countably-infinite] message set Ω and a probability distribution $P: \Omega \rightarrow [0, 1]$. A codemap $f: \Omega \rightarrow \mathbf{G}^+$ puts a probability-distribution on $\mathcal{C} := \text{Range}(f)$ by assigning, for $\mathbf{w} \in \mathcal{C}$,

$$4a: \quad P(\mathbf{w}) := P(f^{-1}(\mathbf{w})).$$

Thus the code has an *expected coding-length*, which we may write as

$$\text{ECL}(\mathcal{C}) \quad \text{or} \quad \text{ECL}(f).$$

MECL. Use *MECL* for Minimum ECL. Consider a *finite* prob-vector $\vec{p} = (p_1, \dots, p_L)$. A code [for the moment, assume a binary code] $\mathcal{C} = (\mathbf{v}_1, \dots, \mathbf{v}_L)$ has

$$3b': \quad \text{ECL}(\mathcal{C}) = \sum_{j=1}^L p_j \cdot \text{Len}(\mathbf{v}_j).$$

The minimum of (3b') taken over *all* prefix-codes, or over all UD-codes, we will call

$$4b: \quad \text{PC-MECL}(\vec{p}) \quad \text{and} \quad \text{UD-MECL}(\vec{p}),$$

respectively. Evidently

$$4c: \quad \text{PC-MECL}(\vec{p}) \geq \text{UD-MECL}(\vec{p})$$

since, for UD-codes, we are taking a minimum over the larger collection of codes. By the way, I'll sometimes use $\text{MECL}(\vec{p})$ as a synonym for $\text{UD-MECL}(\vec{p})$.

The minimum in (3b') *depends on* $\Gamma := |\mathbf{G}|$, the number of letters in our code alphabet. [We can compress English more by coding into a 3-letter alphabet, rather than a 2-letter alphabet.] To indicate the dependency on cardinality Γ , we may write

$$4d: \quad \text{PC-MECL}_\Gamma(\vec{p}) \quad \text{and} \quad \text{UD-MECL}_\Gamma(\vec{p}).$$

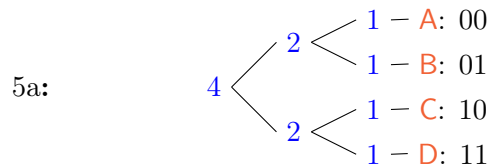
Huffman codes

(Binary HCs were described in class.)

Interpret a tuple such as (3:A 1:B 5:C) as putting prob-distribution $(\frac{3}{9}, \frac{1}{9}, \frac{5}{9})$ on letters (A, B, C); the 9 is the sum of the *weights*, 3 + 1 + 5.

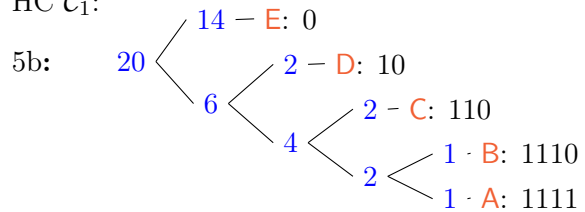
Our convention is that the branch going up-right is labeled with bit 0, and the up-left with bit 1.

Non-uniqueness of Huffman Codes. Frequency-tuple $F := (1:A 1:B 1:C 1:D)$ admits HC



But F also admits each other permutation of {A, B, C, D} being attached to those leaves. So this Freq-tuple admits several HCs.

For a more interesting example, consider Frequency-tuple $F' := (1:A 1:B 2:C 2:D 14:E)$. This admits HC \mathcal{C}_1 :

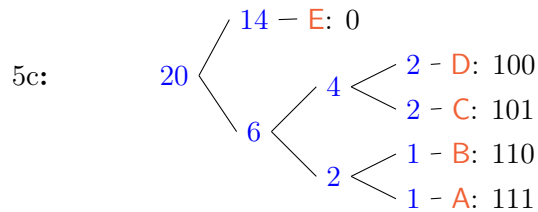


So $20 \cdot \text{ECL}(\mathcal{C}_1)$ equals

$$\overbrace{1 \cdot 2 \cdot 4}^{B,A} + \overbrace{2 \cdot 1 \cdot 3}^C + \overbrace{2 \cdot 1 \cdot 2}^D + \overbrace{14 \cdot 1 \cdot 1}^E = 32.$$

Thus $\text{ECL}(\mathcal{C}_1) = \frac{32}{20} = \frac{8}{5}$ bits-per-letter.

Our F' also admits HC \mathcal{C}_2 :



Thus $20 \cdot \text{ECL}(\mathcal{C}_2)$ equals

$$\overbrace{1 \cdot 2 \cdot 3}^{B,A} + \overbrace{2 \cdot 2 \cdot 3}^{D,C} + \overbrace{14 \cdot 1 \cdot 1}^E = 32.$$

We see that $\text{ECL}(\mathcal{C}_2) = \text{ECL}(\mathcal{C}_1)$. It is worth noticing that codes \mathcal{C}_1 and \mathcal{C}_2 are not only different, they are not even tree-isomorphic. \square

6: HC-same-ECL Thm. Fix a probability L -vec \vec{p} , with $L \geq 2$. Then all \vec{p} -HCs have the same ECL. \diamond

Proof. We proceed by induction on L , with proposition

$R(L)$: For every prob. L -vec \vec{q} : Each two \vec{q} -HCs have the same ECL.

The base $L=2$ case is easy, since the only Huffman-tree is $\text{Root} \leftarrow \begin{smallmatrix} \text{Prob.} \\ \text{Prob.} \end{smallmatrix}$ whose ECL is 1.

Induction step. Fix an $L \geq 3$ st. $R(L-1)$.

Let $J := L-2$. Given \vec{p} , let α, β denote its two lowest probabilities,^{♥3} and write \vec{p} as $(\alpha, \beta, p_1, \dots, p_J)$.

Consider two HCs, \mathcal{C} and \mathcal{X} , with length-spectra that I have written above and below \vec{p} , here.

$$\begin{array}{l} \mathcal{C} : \quad D \ D \ d_1 \ d_2 \ \dots \ d_J \\ \quad (\alpha, \beta, p_1, p_2, \dots, p_J) \\ \mathcal{X} : \quad Y \ Y \ y_1 \ y_2 \ \dots \ y_J. \end{array}$$

So code \mathcal{C} assigns length- D codewords to the first two nodes it joins, which have probs α and β . Computing

$$\begin{array}{l} \text{ECL}(\mathcal{C}) = D \cdot \alpha + D \cdot \beta + \sum_{i=1}^J [d_i \cdot p_i]; \\ \dagger: \\ \text{ECL}(\mathcal{X}) = Y \cdot \alpha + Y \cdot \beta + \sum_{i=1}^J [y_i \cdot p_i]. \end{array}$$

After joining two nodes, the codes now recursively act on $\vec{q} := (\alpha+\beta, p_1, p_2, \dots, p_J)$ and assign length-spectra as follows:

$$\begin{array}{l} \mathcal{C} : \quad D-1 \ d_1 \ d_2 \ \dots \ d_J \\ \quad (\alpha+\beta, p_1, p_2, \dots, p_J) \\ \mathcal{X} : \quad Y-1 \ y_1 \ y_2 \ \dots \ y_J. \end{array}$$

Since \vec{q} is an $[L-1]$ -vector, proposition $R(L-1)$ says that the above two ECLs are equal, i.e

$$\begin{array}{l} \ddagger: \\ [D-1] \cdot [\alpha+\beta] + \sum_{i=1}^J [d_i \cdot p_i] \\ = [Y-1] \cdot [\alpha+\beta] + \sum_{i=1}^J [y_i \cdot p_i]. \end{array}$$

And this implies equality in the two RhSs of (\ddagger). \diamond

^{♥3}They might be equal; indeed, perhaps $\beta = \alpha$, with 8 nodes all having probability α . We are not picking two *nodes*; we are picking two **probabilities**. In particular, I am not assuming that HCs \mathcal{C} and \mathcal{X} join the same two nodes, at the first step.

7a: Depth Lemma. Fix a probability L -vector \vec{p} , and a \vec{p} -PC-MECL. Consider two leaf-nodes with probabilities x and α , at depths D and D' , respectively. If $x > \alpha$, then necessarily $D \leq D'$. \diamond

Exer. E6. Prove the above Depth Lemma. \square

7b: Huffman's theorem.

i: HCs are PC-MECLs.

ii: HCs are UD-MECLs. \diamond

Pf of (i). We induct on L , with proposition

$\text{HUFF}(L)$: Each probability L -vector \vec{q} , admits a Huffman Code which is a PC-MECL.

The base $L=2$ case is immediate, since the only tree is $\text{Root} \leftarrow \begin{smallmatrix} \text{Prob.} \\ \text{Prob.} \end{smallmatrix}$, which is a Huffman-tree.

Induction step. Fix an $L \geq 3$ st. $\text{HUFF}(L-1)$. Fix \vec{p} , a prob. L -vector, and consider a \vec{p} -PC-MECL, viewed as a tree.

Let $\alpha \leq \beta$ denote the two smallest probabilities of \vec{p} . At the tree's deepest level, D , consider two joined leaf-nodes, and call their probabilities x and y . It suffices to show:

We can permute the probabilities of the leaves,
 *: without changing the ECL, so that, now, these two nodes have probabilities α and β .

For then, we collapse these two into a single node, producing prob.-vec $\vec{q} := (\alpha+\beta, p_2, p_3, \dots, p_{L-1})$. By the induction hypothesis, there is a \vec{q} -HC which is a \vec{q} -PC-MECL. Expanding the collapsed node back into $\leftarrow \begin{smallmatrix} \alpha \\ \beta \end{smallmatrix}$ automatically produces a Huffman-tree^{♥4} which is a \vec{p} -PC-MECL. And all HCs have the same ECL, by (6).

^{♥4}The permuting of probabilities, because it is done recursively, can permute interior-nodes of the tree. So the final Huffman-tree can be non-isomorphic to the original PC-MECL tree. This kind of argument is called *tree surgery*.

Establishing (*). If $x = \alpha$, then leave that leaf-node alone. Otherwise, $x > \alpha$. Now the Depth Lemma, (7a), says that α can't be shallower than x , so [since x is at max depth], every α -node has to be at D , the deepest level. Switch some α -leaf with our x -leaf.

This does not change the ECL, since the nodes are at the same depth.

Now our joined-pair is $\prec \frac{\alpha}{y}$. Do the same operation with y w.r.t β . Now our joined-pair is $\prec \frac{\alpha}{\beta}$, as desired. \blacklozenge

Exer. E7. Prove (ii), that every HC is a UD-MECL. \square

Exer. E8. POSTING RACE: Give Proof or CEX: If prefix code \mathcal{C} is a PC-MECL, then \mathcal{C} is a Huffman code. \square

Entropy/Distropy

Define $\eta: [0, 1] \rightarrow [0, \infty)$ by $\eta(x) := x \cdot \log_2(1/x)$, and extend by continuity, so that $\eta(0) = 0$. (Use l'Hôpital's rule, if you like.)

The **distribution entropy**, which I call **distropy**, of a probability-vector \vec{v} is

$$\mathcal{H}(\vec{v}) := \sum_{p \in \vec{v}} \eta(p).$$

For a probability-distr $P()$ on a code^{♥5} \mathcal{C} , then, $\mathcal{H}(P)$ equals $\sum_{\mathbf{v} \in \mathcal{C}} \eta(P(\mathbf{v}))$.

8: Jensen's inequality. On an interval $J \subset \mathbb{R}$, consider points $Q_{\mathbf{v}} \in J$, for each \mathbf{v} in a countable indexing-set \mathcal{C} . We have a probability-distr $P()$ on \mathcal{C} . Then for each convex-down fnc $\mathcal{L}: J \rightarrow \mathbb{R}$

$$8a: \quad \mathcal{L}\left(\sum_{\mathbf{v} \in \mathcal{C}} P(\mathbf{v}) \cdot Q_{\mathbf{v}}\right) \geq \sum_{\mathbf{v} \in \mathcal{C}} P(\mathbf{v}) \cdot \mathcal{L}(Q_{\mathbf{v}}).$$

Now suppose \mathcal{L} is strictly convex-down. Then:

8b: Equality in (8a) IFF the probability-distr is concentrated on a single point.

IOWords, having removed all zero-probability elements from \mathcal{C} , the map $\mathbf{v} \mapsto Q_{\mathbf{v}}$ is constant. \blacklozenge

^{♥5}For comparison with (binary) distropy/entropy, we will usually be examining a **binary code**; a code over a 2-symbol alphabet, \mathbb{B} . (Typically, $\mathbb{B} = \{0, 1\}$.) So a binary code is a subset $\mathcal{C} \subset \mathbb{B}^+$.

Proof. See picture on blackboard. \blacklozenge

9: Distropy UD-code Inequality. Fix a binary code \mathcal{C} and probability distribution $P: \mathcal{C} \rightarrow (0, 1)$. If \mathcal{C} is uniquely decodable, then

$$9a: \quad \text{ECL}(\mathcal{C}) \geq \mathcal{H}(P).$$

There is equality in (9a) IFF

$$9b: \quad \forall \mathbf{v} \in \mathcal{C}: P(\mathbf{v}) = 1/2^{\widehat{\mathbf{v}}}.$$
 \blacklozenge

Pf of (9a). Let “ $\sum_{\mathbf{v}}$ ” mean “ $\sum_{\mathbf{v} \in \mathcal{C}}$ ” and $\widehat{\mathbf{v}}$ mean $\text{Len}(\mathbf{v})$. With $\mathcal{L}() := \log_2()$, note $\text{ECL}(\mathcal{C})$ equals $\sum_{\mathbf{v}} P(\mathbf{v}) \widehat{\mathbf{v}}$, which equals $\sum_{\mathbf{v}} P(\mathbf{v}) \mathcal{L}(2^{\widehat{\mathbf{v}}})$. Consequently, we can write $\mathcal{H}(P) - \text{ECL}(\mathcal{C})$ as

$$\begin{aligned} & \left[\sum_{\mathbf{v}} P(\mathbf{v}) \mathcal{L}\left(\frac{1}{P(\mathbf{v})}\right) \right] - \left[\sum_{\mathbf{v}} P(\mathbf{v}) \mathcal{L}(2^{\widehat{\mathbf{v}}}) \right] \\ & = \sum_{\mathbf{v}} P(\mathbf{v}) \mathcal{L}\left(\frac{1}{P(\mathbf{v})} \cdot \frac{1}{2^{\widehat{\mathbf{v}}}}\right). \end{aligned}$$

Since $\mathcal{L}()$ is strictly convex-down, Jensen's inequality applies to say

$$\begin{aligned} \mathcal{H}(P) - \text{ECL}(\mathcal{C}) & \leq \mathcal{L}\left(\sum_{\mathbf{v}} P(\mathbf{v}) \cdot \frac{1}{P(\mathbf{v})} \cdot \frac{1}{2^{\widehat{\mathbf{v}}}}\right) \\ \dagger: & \quad \quad \quad \stackrel{\text{note}}{=} \mathcal{L}\left(\sum_{\mathbf{v}} 1/2^{\widehat{\mathbf{v}}}\right). \end{aligned}$$

By (2a) the Kraft-McMillan inequality, $\sum_{\mathbf{v}} 1/2^{\widehat{\mathbf{v}}} \leq 1$. And $\mathcal{L}()$ is order-preserving. Thus the above yields

$$\mathcal{H}(P) - \text{ECL}(\mathcal{C}) \leq \mathcal{L}(1) = 0,$$

as desired. \blacklozenge

Pf of (9b). Suppose $\text{ECL}(\mathcal{C}) = \mathcal{H}(P)$. This forces equality in Kraft, so $\sum_{\mathbf{v}} 1/2^{\widehat{\mathbf{v}}} = 1$, and in Jensen's, so the map $\mathbf{v} \mapsto \frac{1}{P(\mathbf{v})} \cdot \frac{1}{2^{\widehat{\mathbf{v}}}}$ is constant; say κ . Thus $P(\mathbf{v}) \cdot \kappa = 1/2^{\widehat{\mathbf{v}}}$, for each $\mathbf{v} \in \mathcal{C}$ implies that $1 \cdot \kappa = 1$. Hence $\kappa = 1$. \blacklozenge

Sardinas-Patterson Algorithm. An example of a UD-code [indeed, it is a suffixcode], for which the SarPat algorithm eventually cycles (as it must), but not with the empty prefix-list, is

$$\{\text{bc}, \text{b}, \text{Xc}, \text{cX}\}.$$

(On hold...)

Decoding-delay for UD-codes. Consider a long word w which is the initial part. . .
(On hold. . .)

§Index for “JK Codes notes”

This is a test of the pre-note.

\mathcal{C} -parsing, 2

alphabet, 1

bi-infinite-UD property, 2

BI-UD, 2

block code (constant-length), 1

cipher, 5

code, 1

codemap, 5

complete code, 4

distribution entropy, 7

distropy, 7

entropy, 7

expected coding-length, ECL, 4

HC, Huffman code, 5

idempotent, 1

Kleene star/plus, 1

Kraft-sum, 4

language, 1

leaf-node, 2

left-infinite-UD-code, 2

LI-UD-code, 2

MECL, 5

nullishcode, 1

nullword, 1

nullword language, 1

Posting race, 3, 4, 7

prefix code, 2

probability distribution, 4

redundant code, 4

RI-UD property, 2

source alphabet, 5

suffix code, 2

tree, 2

Γ -complete/redundant, 4

Γ -full/deficient, 3

Γ -bounded, 3

isomorphism, 2

surgery, 6

UD, uniquely decodable, 2

void language, 1

weak-BI-UD, 2

weakly-UD, 3

word, 1

Filename: Problems/NumberTheory/jk-codes.tex
As of: Tuesday 15Mar2016. Typeset: 23Sep2017 at 15:50.