3F7 Information Theory and Coding

Howard Mei

October 28, 2020

1 Probability and Entropy

1.1 Discrete Random Variables

- Probability mass function (pmf): $P_X(x) = Pr(X = x)$. $x \in \chi$ is a *realisation* of the random variable x
- Cumulative distribution function (cdf): $F_x(a) = Pr(X \leq a) = \sum_{x \leq a} P_X(x)$
- *Expected value*: $\mathbb{E}X = \sum_a a \cdot P_x(a)$
- *Variance*:$Var(X) = \mathbb{E}[(X - \mathbb{E}X)^2] = \mathbb{E}[X^2] - (\mathbb{E}X)^2$.
- $Var(aX) = a^2 \cdot var(X)$
- A function $g(X)$ of rvX is also an rv
- *Expected value* of functions of random variables : $\mathbb{E}[g(X)] = \sum_a g(a) \cdot P_X(a)$

1.2 Jointly distributed random variables

1.2.1 Discrete rvs X,Y

Marginal distributions :

$$P_X(x) = \sum_y P_{XY}(x, y), \qquad P_Y(y) = \sum_x P_{XY}(x, y)$$

Conditional distribution of Y given X :

$$P_{Y|X}(y|x) = \sum_y P_{XY}(x, y), \quad \text{for x such that} \quad P_X(x) > 0$$

1.2.2 Key properties of jointly distributed rvs

- Product rule:

$$\begin{aligned} P_{XYZ} &= P_X P_{Y|X} P_{Z|YX} \\ &= P_Y P_{Z|Y} P_{X|ZY} \\ &= P_Y P_{X|Y} P_{Z|XY} \\ &= P_Z P_{X|Z} P_{Y|XZ} \\ &= P_Z P_{Y|Z} P_{X|YZ} \end{aligned}$$

- Sum rule (marginalization):

$$P_{XY}(x, y) = \sum_z P_{XYZ}(x, y, z)$$

$$P_X(x) = \sum_{y,z} P_{XYZ}(x, y, z) = \sum_y P_{XY}(x, y)$$

These properties extend naturally to multiple jointly distributed rvs $(X_1, ..., X_n)$

1.2.3 Continuous random variables

- Joint density function $f_{XY}(x, y)$
- $Pr(a \leq X \leq b,\ c \leq Y \leq d) = \int_{x=a}^{b} \int_{y=c}^{d} f_{XY}(x, y)\, dxdy$
- For jointly Gaussian rvs, specified by mean vector and covariance matrix
- Conditional density, product and sum rule analogous to discrete case with density replacing pmf and integrals instead of sums

1.2.4 Independence

Discrete random variables $X_1, ..., X_n$ are *statistically independent* if

$$P_{X_1 ... X_n}(x_1, ...x_n) = P_{X_1}(x_1) \cdot P_{X_2}(x_2) \ ... \ P_{X_n}(x_n) \qquad \forall (x_1, ..., x_n)$$

From *product rule* :

$$P_{X_1 ... X_n}(x_1, ..., \dot{x}_n) = P_{X_1}(x_1) \cdot P_{X_2|X_1}(x_2|x_1) \ ... \ P_{X_n|X_{n-1} ... X_1}(x_n|x_{n-1}, ..., x_1)$$

Therefore, when independent:

$$P_{X_i|\{X_j\}_{j \neq i}} = P_{X_i}$$

Often, *independent* and *identically distributed* (*i.i.d.*) random variables are considered in this course

1.3 Entropy

1.3.1 Define

The entropy of a discrete random variable X with *pmf* $P4$ is

$$H(X) = \sum_x P(x) \cdot \log_2 \frac{1}{P(x)} \quad \text{bits}$$

- $H(X)$ can be written as $\mathbb{E}[\log_2 \frac{1}{P(x)}]$
- $H(X)$ can be seen as the **uncertainty** associated with the rv X.

1.3.2 Properties

Let X be discrete random variable takes M different values with different probability. Then:

- $H(X) \geq 0$
- $H(X) \leq \log M$
- Equiprobable distribution $(\frac{1}{M}, ..., \frac{1}{M})$ has the maximum entropy equal to $\log M$. Seen as equal probability gives maximum uncertainty in the outcome

Proof:

- For any $x \in \chi,\ 0 \leq P(X) \leq 1$, $\frac{1}{P(X)} \geq 1$, and hence $\log \frac{1}{P(X)} \geq 0$, $H(X) \geq 0$

  Notes when there is a certain probability of $P(X = a) = 1$, $H(X) = 0$ means no uncertainty in outcome.

- *Proof* of $H(X) \leq \log M$

$$\begin{aligned} H(X) - logM &= \sum_x P(x) log \frac{1}{P(X)} - \sum_x P(x) logM \\ &= \frac{1}{ln2} \sum_x P(x) ln \frac{1}{MP(X)} \\ &\leq \frac{1}{ln2} \sum_x P(x) \left(\frac{1}{MP(x)} - 1 \right) \quad \text{By inequality rule: } lnx \leq (x-1) \\ &= \frac{1}{ln2} \left(\sum_x \frac{1}{M} - \sum_x P(x) \right) = 0 \end{aligned}$$

Hence $H(X) - logM \leq 0 \qquad H(X) \leq \log M$

- *Proof* of maximum entropy Equiprobable distribution:

$$\begin{aligned} H(X) &= logM \text{ when} \\ ln\frac{1}{MP(X)} &= \left(\frac{1}{MP(x)} - 1\right) \text{ when} \\ MP(x) &= 1 \\ P(x) &= 1/M \end{aligned}$$

1.3.3 Joint and Conditional Entropy

- The *joint* entropy of X, Y is

$$H(X,Y) = \sum_{x,y} P_{XY}(x,y)\, log\frac{1}{P_{XY}(x,y)}$$

- The *conditional* entropy of Y given X is

$$H(Y|X) = \sum_{x,y} P_{XY}(x,y)\, log\frac{1}{P_{Y|X}(y|x)}$$

  - Can be seen as the uncertainty for Y is different for different X, $H(Y|X)$ is the average uncertainty in Y given X

$$H(Y|X) = \sum_{x} P_X(x) \underbrace{\sum_{y} P_{Y|X}(y|x)\, log\frac{1}{P_{Y|X}(y|x)}}_{\text{H(Y—X=x) another entropy equation}}$$

  - $H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$
  - When X,Y are independent $H(Y|X) = H(Y)$ as uncertainty of Y is not changed if independent.
  - When $Y = f(X)$, $H(Y|X) = H(f(X)|X) = 0$ as we can predict any f(X) from X, no uncertainty. However, inversely $H(X|Y) \geq 0$ zeros only when the function is one to one.

$$H(X,Y) = H(X|Y) + H(Y) = H(Y|X) + H(X) = H(X)$$

1.3.4 Joint Entropy of Multiple RVs

-

$$H(X_1, ..., X_n) = \sum_{x_1,..,x_n} P_{X_1..X_n}(x_1, .., x_n)\, log\frac{1}{P_{X_1..X_n}(x_1, .., x_n)}$$

- Chain rule of joint entropy

$$\begin{aligned} H(X_1, ..., X_n) &= H(X_1) + H(X_2|x_1) + H(X_n|X_{n-1}, .., X_1) \\ &= \sum_{i=1}^{n} H(X_i|X_{i-1}, .., X_1) \end{aligned}$$

where the conditional entropy

$$H(X_i|X_{i-1}, .., X_1) = \sum_{x_1,..,x_i} P_{X_1..X_i}(x_1, .., x_i)\, log\frac{1}{P_{X_i|X_1,..,X_{i-1}}(x_i|x_1, .., x_{i-1})}$$

- If independent, then

$$H(X_1, ..., X_n) = \sum_{i=1}^{n} H(X_i)$$

- proof of Chain rule

$$P(x_1, ..., x_n) = P_{X_1}(x_1)P(x_2|x_1)...P(x_n|x_{n-1}, ..., x_1) = \prod_{i=1}^{n} P(x_i|x_{i-1}, ..., x_1)$$

$$
\begin{aligned}
H(X_1, ..., X_n) &= \sum_{x_1,..,x_n} P(x_1, .., x_n)\, log \frac{1}{P(x_1, .., x_n)} \\
&= -\sum_{x_1,..,x_n} P(x_1, .., x_n)\, logP(x_1, .., x_n) \\
&= -\sum_{x_1,..,x_n} P(x_1, .., x_n)\, log \prod_{i=1}^{n} P(x_i|x_{i-1}, ..., x_1) \\
&= -\sum_{x_1,..,x_n} \sum_{i=n}^{n} P(x_1, .., x_n)\, logP(x_i|x_{i-1}, ..., x_1) \\
&= -\sum_{i=n}^{n} \sum_{x_1,..,x_n} P(x_1, .., x_n)\, logP(x_i|x_{i-1}, ..., x_1) \\
&= -\sum_{i=n}^{n} \sum_{x_1,..,x_i} P(x_1, .., x_i)\, logP(x_i|x_{i-1}, ..., x_1) \\
&= \sum_{i=1}^{n} H(X_i|X_{i-1}, .., X_1)
\end{aligned}
$$

2 Law of Large Numbers, Typicality, Data Compression

2.1 Estimating Tail probability

- We want to bound the probability of rare events, corresponding to probability mass in the 'tails' of the pmf/density function

  e.g. What is the **bound** for $P(X > 20)$ for average 5 cars/minute without knowing the distribution?

- Markov and Chebyshev inequalities are ways to bound tail probabilities with limited information.
  - Markov is for non-negative rvs and requires only the mean
  - Chebyshev is for general rvs and requires mean and variance

2.1.1 Markov's Inequality

- For a non-negative rv X and any $a > 0$,

$$P(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$$

- *Proof:*

$$\mathbb{E}[X] = \sum_{r \geq 0} rP(X = r) = \sum_{0 \leq r \leq a} rP(X = r) + \sum_{r \geq a} rP(X = r)$$

$$\text{For } r \geq a, \quad \sum_{r \geq a} rP(X = r) \geq \sum_{r \geq a} aP(X = r)$$

$$\mathbb{E}[X] \geq \sum_{0 \leq r \leq a} rP(X = r) + \sum_{r \geq a} aP(X = r)$$

$$= \sum_{0 \leq r \leq a} rP(X = r) + aP(X \geq a)$$

$$\mathbb{E}[X] \geq aP(X \geq a)$$

2.1.2 Chebyshev's inequality

- Bound the tail probabilities of deviations around the mean
- For any rv X and $a > 0$,

$$P(|X - \mathbb{E}X| \geq a) \leq \frac{Var(X)}{a^2}$$

- *Proof:*

$$P(|X - \mathbb{E}X| \geq a) = P(|X - \mathbb{E}X|^2 \geq a^2)$$

$$\text{Let } Y = |X - \mathbb{E}X|^2$$

$$\text{Apply Markov's,} \qquad P(Y \geq a^2) \leq \frac{\mathbb{E}Y}{a^2} \qquad \text{Note that } \mathbb{E}Y = Var(X)$$

$$P(|X - \mathbb{E}X| \geq a) \leq \frac{Var(X)}{a^2}$$

2.1.3 Weak Law of Large Numbers (WLLN)

- "Empirical average converges to the mean"
- Let X1,X2, be a sequence of i.i.d. rvs with finite mean μ. $S_n = \frac{1}{n}\sum_{i=1}^{n} X_i$
  - Informal WLLN: $S_n \to \mu$ as $n \to \infty$
  - Formal WLLN: For any $\epsilon > 0$, $lim_{n\to\infty} P(|S_n - \mu| \geq \epsilon) = 0$
- *Proof:*

  By Chebyshev's inequality:

$$P(|S_n - \mu| \geq \epsilon) \leq \frac{Var(S_n)}{\epsilon^2}$$

Also:

$$Var(S_n) = 1/n^2 \cdot Var(\sum_i X_i) = 1/n^2 \cdot \sum_{i=1}^{n} Var(X_i) = 1/n^2 \cdot n\sigma^2 = \frac{\sigma^2}{n}$$

Hence:

$$P(|S_n - \mu| \geq \epsilon) \leq \frac{\sigma^2}{n\epsilon^2}$$

2.2 Typical Set

- Simple Example: Consider an i.i.d. Bernoulli($\frac{1}{4}$) source.

$$P(X_i = 1) = \frac{1}{4} \quad P(X_i = 0) = \frac{3}{4} \text{ for } i = 1, 2, 3, ...$$

1. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2. 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0

   - 16 bits sequence
   - Probability of first sequence = $(\frac{3}{4})^1 6$
   - Probability of second sequence = $(\frac{1}{4})^4(\frac{3}{4})^1 2$
   - The first sequence is $\mathbf{3^4}$ more likely than second!!

- **Typical Sequences**:
  - Though less likely than first sequence, the second is more "typical" of the $(\frac{1}{4}, \frac{3}{4})$ source If $X_1, ...X_n$ are chosen i.i.d. Bernoulli(p), then for large n:
  - With high probability, the fraction of ones observed sequence will be close to p by WLLN
  - With high probability the observed sequence will have probability close to $p^{np} * (1-p)^{n(1-p)}$
  - Any number can be written as $2^{\log a}$ Hence,

$$p^{np} * (1-p)^{n(1-p)} = (2^{\log p})^{np}(2^{\log(1-p)})^{n(1-p)} = 2^{-nH_2(p)}$$

  - For large n, with high probability we will observe a typical sequence . Informally, a typical sequence is one whose probability is close to $2^{-nH_2(p)}$

- **Asymptotic Equipartition Property (AEP)**
  The AEP makes this for any i.i.d. discrete source not just Bernoulli sequences.

  - If $X_1, ...X_n$ are chosen i.i.d. P_x, then for any $\epsilon > 0$

$$\lim_{n\to\infty} Pr\left(\left|\frac{-1}{n}\log P_X(X_1, X_2, ..., X_n) - H(X)\right| < \epsilon\right) = 1$$

  - $\frac{-1}{n}\log P_X(X_1, X_2, ..., X_n)$ is a random variable, a fucntion of rvs is a rv
  - AEP says this rv converges in probability to $H(X)$ a constant, as $n \to \infty$
  - Proof:
    - let $Y_i = -\log P_X(X_i)$
    - Functions of independent rvs are also independent rvs
    - WLLN for Y_i's says that for any $\epsilon > 0$ (Note the $<$ sign replaced the $\geq$)

$$\lim_{n\to\infty} Pr(|\frac{1}{n}\sum_i Y_i - \mathbb{E}[Y_i]| < \epsilon) = 1$$

    - Sum of log become log of multiply:

$$\sum_i Y_i = -\log[P_X(X_1)...P_X(X_n)] = -\log P_X(X_1, X_2, ..., X_n)$$

* $\mathbb{E}[Y_i] = H(X)$

- **The typical set**
  - Definition:
    The typical set $A_{\epsilon,n}$ with respect to P is the set of sequence $x_1, ..., x_n \in \mathcal{X}^n$ with the property

$$2^{-n(H(X)+\epsilon)} \leq P(x_1, ..., z_n) \leq 2^{-n(H(X)-\epsilon)}$$

    "Sequences with probability concentrated around $2^{-n(H(X)}$"
  - Dependence on n and ϵ A sequence belonging to the typical set is called an ϵ-type sequence

- Properties of the **Typical Set**
  - Property 1:
    * if $X^n = (X_1, ..., X_n)$ is generated i.i.d. P, then

$$Pr(X^n \in A_{\epsilon,n}) \xrightarrow{n\to\infty} 1$$

    * The sequence $X_1, , ..., X_n$ observed us very likely to belong to the typical set.
    * Proof:

$$X^n = (X_1, ..., X_n) \Leftrightarrow 2^{-n(H(X)+\epsilon)} \leq P(X^n) \leq 2^{-n(H(X)-\epsilon)}$$
$$\Leftrightarrow H(X) - \epsilon \leq -\frac{1}{n}\log P(X^n) \leq H(X) + \epsilon$$

      By AEP:

$$Pr(X^N \in A_{\epsilon,n}) = Pr\left(H(X) - \epsilon \leq -\frac{1}{n}\log P(X^n) \leq H(X) + \epsilon\right) \xrightarrow{n\to\infty} 1$$

  - Property 2:
    * Let $|A_{\epsilon,n}|$ denote the number of elements in the typical set $A_{\epsilon,n}$. Then:

$$|A_{\epsilon,n}| \leq 2^{n(H(X)+\epsilon)}$$

    * Proof:

$$\begin{aligned} 1 &= \sum_{x^n \in \mathcal{X}^n} P(x^n) \\ &\geq \sum_{x^n \in A_{\epsilon,n}} P(x^n) \\ &\geq \sum_{x^n \in A_{\epsilon,n}} 2^{-n(H(X)+\epsilon)} \\ &= 2^{-n(H(X)+\epsilon)}|A_{\epsilon,n}| \end{aligned}$$

  - Property 3:
    * For sufficiently large n, $|A_{\epsilon,n}| \geq (1-\epsilon)2^{n(H(X)-\epsilon)}$
    * Proof: from Property 1, $Pr(X^n \in A_{\epsilon,n}) \xrightarrow{n\to\infty} 1$
      This means for any $\epsilon > 0$, $Pr(X^n \in A_{\epsilon,n}) > 1 - \epsilon$ for sufficiently large n:

$$\begin{aligned} 1 - \epsilon &< Pr(X^n \in A_{\epsilon,n}) \\ &= \sum_{x^n \in A_{\epsilon,n}} P(x^n) \\ &\leq \sum_{x^n \in A_{\epsilon,n}} 2^{-n(H(X)-\epsilon)} \\ &= 2^{-n(H(X)-\epsilon)}|A_{\epsilon,n}| \end{aligned}$$

- Summary of the properties:
  - Suppose generated $X_1, ..., X_n$ i.i.d. P. With high probability, the sequence will be typical, i.e., its probability is close to $2^{-n(H(X)}$
  - If the pmf P assigns non-zero probabilities to the symbols denoted a,b,c etc., the typical set is essentially the set of sequences whose fraction of a's is close to $P(a)$, fraction of b's is close to $p(b)$, and so on
  - The number of typical sequences is close to $2^{n(H(X)}$

2.3 Compression

GOAL: To compress a source producing symbols $X_1, X_2, ... \in \chi$ that are i.i.d P.

- To each source sequenceX_n, the code assigns a *unique* binary sequence $c(X^n)$
- $c(X^n)$ is called the *codeword* for the source sequence $Let X^n$.
- $l(X^n)$ be the length of the codeword assined to X^n i.e., the number of bits in $c(X^n)$
- The expected code length is defined as:

$$\mathbb{E}[l(X^n)] = \sum_{x^n} P(x^n) l(x^n)$$

2.3.1 A Naive Compression Code

English: 26 chars $/H(X) = 4$ bits

- List all the $|\chi|^n$ possible length n sequences
- Index these as $\{0, 1, ..., |\chi|^n - 1\}$ using $\lceil \log |\chi|^n \rceil$ bits. English - 5n bits
- Expected code length $\mathbb{E}[l(X^n)] = n \log |\chi|$
- Expected number of bits/symbol: $\mathbb{E}[l(X^n)]/n = \log |\chi|$ English - 5 bits per symbol

How can we do better?

2.3.2 Compression via the Typical Set

- Compression scheme:
  - At most $2^{n(H(X)+\epsilon)}$ $\epsilon-$ typical sequences.
  - Index each sequence in $A_{\epsilon,n}$ using $\lceil \log 2^{n(H(X)+\epsilon)} \rceil$ bits. Prefix each of these by a flag bit 0.

$$\text{Bits/typical seq.} = \lceil n(H(X) + \epsilon) \rceil + 1 \leq n(H(X) + \epsilon) + 2$$

  - Index each sequence not in $A_{\epsilon,n}$ using $\lceil \log |\chi|^n \rceil$ bits. Prefix each of these by a flag bit 1.

$$\text{Bits/non-typical seq.} = \lceil n \log |\chi| \rceil + 1 \leq n \log |\chi| + 2$$

- Expected code length:

$$\begin{aligned}
\mathbb{E}[l(X^n)] &= \sum_{x^n} P(x^n) l(x^n) \\
&= \sum_{x^n \in A_{\epsilon,n}} P(x^n) l(x^n) + \sum_{x^n \notin A_{\epsilon,n}} P(x^n) l(x^n) \\
&\leq \sum_{x^n \in A_{\epsilon,n}} P(x^n)(n(H(X) + \epsilon) + 2) + \sum_{x^n \notin A_{\epsilon,n}} P(x^n)(n \log |\chi| + 2) \\
&\leq 1 \cdot n(H(X) + \epsilon) + \epsilon \cdot n \log |\chi| + 2 \left(\sum_{x^n \in A_{\epsilon,n}} P(x^n) + \sum_{x^n \notin A_{\epsilon,n}} P(x^n) \right) \\
&= n(H(X) + \epsilon) + \epsilon n \log |\chi| + 2 \\
&= n(H(X) + \epsilon')
\end{aligned}$$

- Fundamental limit of Compression
  - For n sufficiently large, there exists a code that maps sequences x^n of length n into binary strings such that the mapping is one-to-one and
$$\mathbb{E}[\frac{1}{n}l(X^n)] \leq H(X) + \epsilon$$
  - In fact the expected length of any uniquely decodable code satisfies
$$\mathbb{E}[\frac{1}{n}l(X^n)] \geq H(X)$$
  - Hence entropy is the fundamental limit of loseless compression

3 Prefix-free codes, Kraft inequality, Loseless source coding theorem

3.1 Prefix-free codes

- Definition: A code is called prefix-free or instantaneously decodable if no codeword is the prefix of another.
  1. $\{0, 00, 10\}$? No
  2. $\{0, 10, 11\}$? Yes
  3. $\{00, 010, 011, 0101, 1\}$? No
- Extension Codes and Unique Decodability
  - Given a binary code C for alphabet χ, the extension code C^n is the code applied symbol-by-symbol to strings:
$$C(x_1x_2...x_n) = C(x_1)C(x_2)...C(x_n)$$
  - The extension code C^n is uniquely decodable if for each binary codeword in it, there is only one possible source string can produce it
  - Prefix-free codes are uniquely decodable, but not all uniquely decodable codes are prefix-free
- We will focus only on designing and analysing prefix-free codes as we want fast encoding and decoding algorithms.

3.2 Kraft Inequality

- The number of leaves at a level is 2^l
- $$\sum_{i=1}^{N} 2^{l_{max}-l_i} \leq 2^{l_{max}} \Rightarrow \sum_{i=1}^{N} 2^{-l_i} \leq 1$$
- Derived from: Total number of unusable codes at level max ¡ Total number of codes at level max. While each codeword of length l leads to s set of $2^{l_{max}-l_i}$ unusable leaves at depth $2^{l_{max}}$
- A necessary condition for any prefix-free code with length$\{l_1...l_N\}$
- A sufficient condition, if a set of l length satisfy it, we can always construct a prefix-free code with these length.

3.3 Coding theorem for a random variable

Let X be a random variable taking values in χ with entropy $H(X)$ The expected codeword length L of any binary prefix-free code for X satisfies
$$L \geq H(X)$$

Proof: Denote the probability of symbols as $p_1, p_2, ...$ and the corresponding codeword length as $l_1, l_2, ...$

$$\begin{aligned} H(X) - L &= \sum_i p_i \log_2(1/p_i) - \sum_i p_i l_i \\ &= \sum_i p_i \log_2(2^{-l_i}/p_i) \\ &= \frac{1}{\ln 2} \sum_i p_i \ln_2(2^{-l_i}/p_i) \\ &\leq^{(a)} \frac{1}{\ln 2} \sum_i p_i \left(2^{-l_i}/p_i - 1\right) \\ &= \frac{1}{\ln 2} \left(\sum_i 2^{-l_i} - \sum_i p_i \right) \\ &\leq^{(b)} \frac{1}{\ln 2}(1-1) = 0 \end{aligned}$$

(a) $\ln(x) \leq x - 1$
(b) Kraft's inequality

3.4 Coding theorem for blocks of source symbols

Instead of assigning codeword to each source symbol, we want to assign codeword to blocks of N source symbols.

Denoting the block by $X^N := (X_1, ..., X_N)$ Hence the Expected length of the code satisfies:

$$E[l(X^N)] \geq H(X^N) \qquad \frac{E[l(X^N)]}{N} \geq \frac{H(X^N)}{N}$$

If X is an iid source, then

$$H(X^N) = NH(X) \rightarrow \frac{E[l(X^N)]}{N} \geq H(X)$$

Remarks:

1. We showed for iid source any prefix-free code has average code length :

$$\frac{E[l(X^N)]}{N} \geq H(X)$$

   This is also true for any uniquely decodable code, since all we used is Kraft's inequality which holds for uniquely decodable codes as well.

2. From fundamental limit of compression we have:

$$\mathbb{E}[\frac{1}{N} l(X^N)] \leq H(X) + \epsilon$$

   for sufficiently large N

Above together form **Shannon's lossless source coding theorem for iid sources**

- You can construct a uniquely decodable code with expected length arbitrarily close to the entropy (By taking block length N large enough)
- Conversely, you cannot construct a uniquely decodable code with expected length than the source entropy.

4 Shannon-Fano coding, Huffman coding, Arithmetic coding

4.1 Shannon-Fano Coding

- Expected code length $L \geq H(X)$

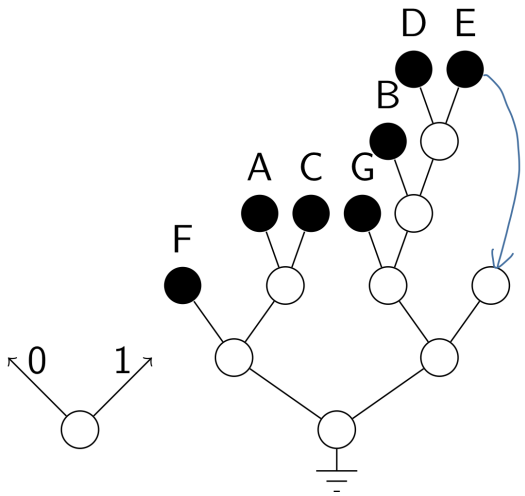
$$L = \sum_{i=1}^{m} p_i l_i \geq \sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}$$

- When can this inequality become an equality?
- How do we pick lengths to make it as tight as possible

- Since l_i are integers, an obvious way to choose them is:
  -
  $$l_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$$
  - Note that $x \leq \lceil x \rceil < x + 1$ Therefore
  $$L = \sum_i p_i l_i < \sum_i p_i \left(\log_2 \frac{1}{p_i} + 1 \right) = H(X) + 1$$
  $$L < H(X) + 1$$

- Verify Shannon-Fano Coding satisfy Kraft inequality
$$\sum_i 2^{l_i} = \sum_i 2^{-\left\lceil \log_2 \frac{1}{p_i} \right\rceil} \leq \sum_i 2^{-\log_2 \frac{1}{p_i}} = \sum_i p_i = 1$$

x	p_i	$\log(1/p_i)$	ℓ_i
A	0.15	2.73	3
B	0.07	3.83	4
C	0.17	2.55	3
D	0.06	4.05	5
E	0.06	4.05	5
F	0.31	1.68	2
G	0.18	2.47	3



- Clearly, we can do better by pushing B/D/E to the length two

4.2 Properties of Optimal prefix-free code

1. The lengths are ordered inversely with the probabilities, if $p_j > p_k$, then $l_j \leq l_k$
2. The two last probable symbols have the same length and are on neighboring leaves in the binary tree (i.e. they differ only in the last digit).

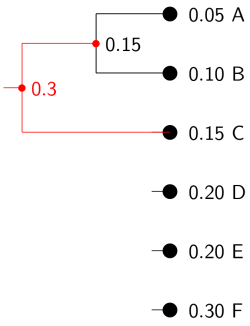
4.3 Huffman Coding

- An algorithm gives an optimal prefix-free code for a given set of probabilities.
  1. Take the two least probable symbols in the alphabet. these two symbols will be given the longest codewords, which will have equal length, and differ only in the last digit
  2. Combine these two symbols into a single symbol, and repeat.

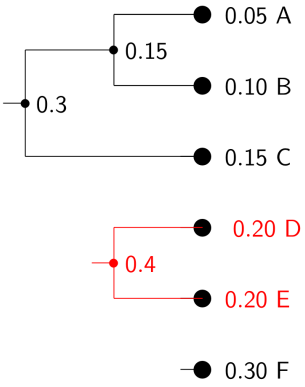
- A source produces symbols from the set {A,B,C,D,E,F} with probabilities {0.05,0.1,0.15,0.2,0.2,0.3}
  1. List these symbols in increasing order of their probabilities.
  2. Combine the two simple with the smallest probabilities, and form a "super-symbol" with the sum of their probabilities.

(a) Step 1

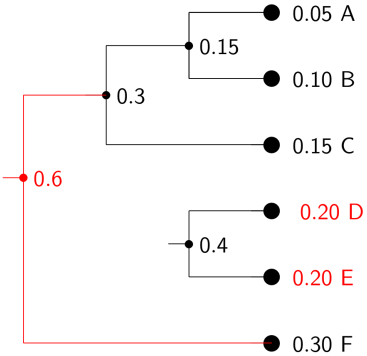
(b) Step 2



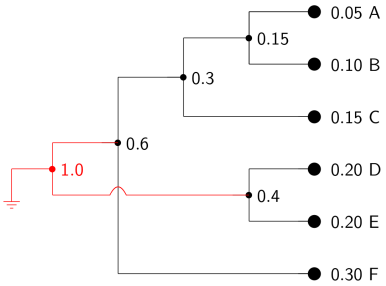
(c) Step 3



(a) Step 4



(b) Step 5



(c) Step 6

3. We now have five symbols with probabilities {0.15,0.15,0.2,0.2,0.3}. Again combine the two symbols with the smallest probabilities...
4. Among the four remaining symbols, D,E have the smallest probabilities, so combine
5. Among the three remaining symbols, the smallest probabilities are 0.3,0.3, so combine them...
6. Finally, combine the remaining two symbols
7. The symbols are the leaves of a tree. The final step is to assign the codewords to the symbols using the tree.

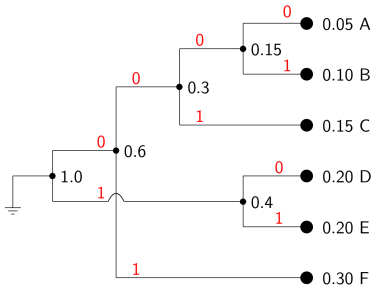


Figure 3: Assign codes

The Huffman code is:

$$F \to 01, E \to 11, D \to 10, C \to 001, B \to 0001, A \to 0000$$

- Properties of the Huffman Code

1. For a source with alphabet of size m, the Huffman algorithm requires $m-1$ steps of combining the two smallest probabilities at each stage.

2. The Huffman code for a given source may not be unique: Swapping 0/1, also 3 same prob can take any two. However, the expected code length should be the same.

- Optimality of Huffman coding: For a given set of probabilities, there is no prefix-free code that has smaller expected length than the Huffman code.
- Huffman codes are optimal for coding a single random variable X, and have expected code length that is less than $H(X) + 1$ But they have some weaknesses:
  - To reduce this overhead of up to 1 bit/symbol, we could design a Huffman code for blocks of k symbols. This would give us an overhead of 1 bit/k symbols, or 1/k bits/symbols.
  - But this comes with the expense of increased complexity. For blocks of k symbols, the binary tree is much larger.
  - These defects are addressed by Arithmetic coding, a scalable algorithm whose expected code length is very close to the source entropy for large sequences. It can also easily deal with non-iid source like text.

4.4 Interval Coding

- Key idea: Each symbol can be represented as an interval inside $[0, 1]$ with length of the interval equal to the symbol probability.

  A source with m symbols with probabilities $\{p_1, ..., p_m\}$ is represented using m intervals

$$[0, p_1), [p_1, p_1 + p_2), ..., [\sum_{i=1}^{m-1} p_i, \sum_{i=1}^{m-1} p_i + p_m).$$

- Example: To represent interval $[0.17, 0.43)$ convert the end-points to binary and find a binary interval lies completely inside this interval. say $[010, 011]$ is the interval of $[0.25, 0.375]$ We choose **010** as the codeword for $[0.17, 0.43)$.
- In general, the binary codeword for a symbol with probability p represented by the interval $[a, a+p)$ can be obtained as follows:
  1. Find the largest *dyadic interval* of the form $[\frac{j}{2^l}, \frac{j+1}{2^l})$ that lies within $[a, a + p)$
  2. Take the binary representation of the lower end-point of the dyadic interval as the codeword.
- Code length: With l code bits, the dyadic intervals have length 2^l. The dyadic interval has to be contained within an interval of length p. Hence

$$2^l \leq p \rightarrow \lceil \log_2(1/p) \rceil \leq l$$

  But sometimes we'll need $l = \lceil \log_2(1/p) \rceil + 1$
  The expected code length can therefore be bounded as

$$L = \sum_i p_i l_i \leq \sum_i p_i (\lceil \log_2(1/p_i) \rceil + 1) < H(X) + 2$$

- Performance: In terms of expected code length, the analysis above shows that interval codes are in general not as good as Huffman codes or even Shannon-Fano codes. But interval coding is the basis for arithmetic coding, which is a powerful technique for long source sequences.

4.5 Arithmetic Coding

- Explain with an example. Consider a source producing symbols $X_1, X_2, ..., X_n$ which are i.i.d., with each X_i taking values in $\{a, b, c\}$ with probabilities $\{0.2, 0.45, 0.35\}$
- Key ideas:
  - Each length n string $(x_1, ..., x_n)$ is represented by a disjoint interval with length equal to the probability of the string.
    - E.g. for $n = 2$, $X_1 = b, X_2 = c$ corresponds to the probability of length $0.45 * 0.35 = 0.1575$ and $X_1 = b, X_2 = a$ of length $0.45 * 0.2 = 0.09$.
  - The interval for $(X_1 = x, X_2 = y)$ is a sub interval of the interval for $X_1 = x$

    - E.g. $X_1 = b$ is the interval $[0.2, 0.65)$ and $(X_1 = b, X_2 = c)$ is the subinterval of length 0.1575 within $[0.2, 0.65)$ To calculate, we rewrite the interval as $\{0, 0.2, 0.65, 0.1\}$ as the order of a,b,c.
    - The subinterval becomes $0.2 + (0.65 - 0.2) * 0.65 \to 0.2 + (0.65 - 0.2) * 1$ That is $[0.4925, 0.65)$ Also we can verify that $0.65 - 0.4925 = 0.1575$
  - Similarly, for any symbols x, y, z, $P(X_1 = x, X_2 = y, X_3 = z)$ is a sub interval of $P(X_1 = x, X_2 = y)$, and so on.

- Decoding: Using binary codeword to sequentially zoom in to the interval, decoding symbols as you go along.

- Expected code length:
  Arithmetic coding can be performed in any sequence of length n, so that any sequence $x_1, ..., x_n$ can be represented by an interval of length $p(x_1, ..., x_n)$, which gives a binary codeword of length at most $\lceil \log_2 \frac{1}{p(x_1,...,x_n)} \rceil + 1$.
  Therefore the expected code length for length n sequences is bounded as:

$$L_n = \sum_{X_n} p_{x_n} l_{x_n} \leq \sum_{x_n} p_{x_n} (\log_2 (1/p(x_1, ..., x_n)) + 2) = H(X^n) + 2$$

  Therefore the expected code length per symbol is

$$\frac{L_n}{n} < \frac{H(X^n)}{n} + \frac{2}{n} = H(X) + \frac{2}{n},$$

  Where the last equality hold for *iid* P_X

4.6 Arithmetic coding for non-iid sources

- Consider a source that produces symbols in alphabet $\{a_1, ..., a_m\}$ with a known distribution

$$P(x_1)p(x_2|x_1)...P(x_n|x_1, ..., x_{n-1})$$

  The arithmetic coding algorithm can be easily extended to such sources. As before, consider the source with alphabet $\{a, b, c\}$ with probabilities $\{0.2, 0.45, 0.35\}$.
  Suppose that three conditional distributions $P(X_2|X_1 = a)$, $P(X_2|X_1 = b)$, $P(X_2|X_1 = c)$ and say

$$P(X_2 = a|X_1 = b) = 0.5, P(X_2 = b|X_1 = b) = 0.3, P(X_2 = c|X_1 = b) = 0.2$$

  We do the same thing as before, just change the probability using the conditional probability accordingly.

- Coding Algorithm:
  - Let source alphabet $\mathcal{A}$ be $\{a_1, ..., a_m\}$
  - We are given a source sequence $x_1, x_2, ..., x_n$ where each $x_i \in \mathcal{A}$
  - Both encoder and decoder know the conditional distributions $P_{X_1}, P_{X_2|X_1}, ..., P_{X_n|X_1,...,X_{n-1}}$.
  - The encoding algorithm computes the interval using the following lower and upper cumulative probabilities. For $i = 1, ...m$ and for $k - 1, ..., n$ we define

$$L_k(a_i|x_1, ..., x_{k-1}) = \sum_{i'}^{i-1} P(X_k = a_{i'}|X_1 = x_1, ..., X_{k-1} = x_{k-1}) U_k(a_i|x_1, ..., x_{k-1}) = \sum_{i'}^{i} P(X_k = a_{i'}|X_1 = x_1, ..., X_{k-1} =$$

- Finite precision issues: The arithmetic encoding algorithm above assumes an infinite precision computer:
  - As n grows, the length of the interval corresponding to a sequence $(x_1, ..., x_n)$ shrinks
  - Hence the number of digits needed to accurately store the values lo and hi grows with n.

Summary:

1. Arithmetic coding can achieve compression very close to the source entropy, with complexity scaling linearly with the length of the sequence.
2. It does require you to know the conditional distribution. But this approach fits well with machine learning techniques that can mine huge quantities of text/speech/video data to build good probabilistic models.
3. Note that the assumed distribution of the source doesn't need to be true one, it only needs to be the same at both encoder and decoder.
4. Arithmetic coding works even when you generate the source conditional distribution on the fly, based on what has been observed so far. Given the generating rule, the decoder will also generate required conditional distributions as it reconstructs the sequence.