Elements of Information Theory

Data Compression(Channel Coding)

Bilingual course
(Chinese taught course)

Information and Communication Eng. Dept.
Deng Ke

Outline

Sequence

- Sequence $u^L = (u_1, u_2, \cdots, u_L)$
- Probability

$$p(u^L) = p(u_1 \cdots u_l \cdots u_L) \overset{\text{memoryless}}{=} p(u_1)\,p(u_2)\ldots p(u_L) = \prod_{l=1}^{L} p(u_l)$$

- Information

$$I(u^L) = \log \frac{1}{p(u^L)} = \log \frac{1}{p(u_1 \cdots u_l \cdots u_L)}$$

$$\overset{\text{memoryless}}{=} \log \frac{1}{p(u_1)\,p(u_2)\ldots p(u_L)}$$

$$= \log \frac{1}{p(u_1)} + \log \frac{1}{p(u_2)} + \cdots + \log \frac{1}{p(u_L)}$$

$$= I(u_1) + I(u_2) + \cdots + I(u_L) = \sum_{l=1}^{L} I(u_l)$$

Sequence

- Entropy

$$H(U^L) = -\sum_{u^L \in U^L} p(u^L) \log p(u^L)$$

$$= \sum_{\underline{u} \in U^L} [\prod_{l=1}^{L} p(u_l)] \log \frac{1}{\prod_{l=1}^{L} p(u_l)}$$

$$= \sum_{\underline{u} \in U^L} [\prod_{l=1}^{L} p(u_l)][\log \frac{1}{p(u_1)} + \log \frac{1}{p(u_2)} + \cdots + \log \frac{1}{p(u_L)}]$$

$$= \sum_{u_1} p(u_1) \log \frac{1}{p(u_1)} [\sum_{u_2} \cdots \sum_{u_L} p(u_2) \cdots p(u_L)]$$

$$+ \sum_{u_2} p(u_2) \log \frac{1}{p(u_2)} [\sum_{u_1} \sum_{u_3} \cdots \sum_{u_L} p(u_1)\,p(u_3) \cdots p(u_L)]$$

$$+ \cdots + \sum_{u_L} p(u_L) \log \frac{1}{p(u_L)} [\sum_{u_1} \cdots \sum_{u_{L-1}} p(u_1) \cdots p(u_{L-1})]$$

Sequence

$$= H(U_1) + H(U_2) + \cdots + H(U_L)$$

$$= \sum_{l=1}^{L} H(U_l)$$

$$= LH(U)$$

- Summary
- The information of a sequence is the sum of the information of the symbols in the sequence
- The entropy of a sequence is the L fold of a symbol where L is the length of the sequence

Fixed-Length Coding

- Let $u^L = (u_1, u_2, \cdots, u_L)$ denote a sequence of L letters, where each letter is a selection from the alphabet $u_l \in \{S_1, S_2, \cdots, S_K\}$
- Code $x^N = (x_1, x_2, \cdots, x_N)$, and the alphabet of fixed-length code is $x_n \in \{a_1, a_2, \cdots, a_D\}$
- Code alphabet contains D symbols and the length of each code is N.
- Only one source sequence can be assigned to each code sequence
- Fixed-length coding $D^N \geq K^L$

Principle

- as few N/L as possible
- Not all source sequence has the corresponding code sequence
- $L \to \infty$
- Typical sequence
- We focus on the typical set $A_\epsilon^{(n)}$
- Only coding the typical sequences
- The number of the typical sequences

$$\left|A_\varepsilon^{(n)}\right| \le 2^{L(H(U)+\varepsilon)}$$

Source Coding Theorem

- The length of code sequence $N \log D \ge L[H(U)+\varepsilon]$
- Source Coding Theorem. Let a DMS have entropy $H(U)$ and consider a coding from sequences of L source letters into sequences of N code letters from a code alphabet of size D. Only one source sequence can be assigned to each code sequence and. Let Pe be the probability of occurrence of a source for which no code sequence has bee provided. Then, for any , if $N \log D \ge L[H(U)+\varepsilon]$, Pe can be made arbitrarily small.

AEP for Data Compression

- What is the problem?
  - No structure
  - complexity The choice of n – length (time, storing space)
- Let $X_1, X_2, \ldots X_n$ be a sequence of i.i.d random variables with distribution $p(X)$
- for any $\varepsilon>0$, we can choose n sufficiently large so that there exists a code that maps sequence $(x_1, x_2, \ldots x_n)$ of length n into binary strings so that
  - The mapping is one-to-one (i.e., invertible)

AEP for Data Compression

$$E\left[\tfrac{1}{n} l(X^n)\right] \le H(X)+\varepsilon$$

- with $l(X^n)$ being the length of codeword corresponding to X^n
- We can represent sequences X^n using $nH(X)$ bits on average.
- We divide all sequences in $A_\epsilon^{(n)}$ into two sets: the typical set and its complement.
- How many bits we need to encode the sequences in $A_\epsilon^{(n)}$?

AEP for Data Compression

$$\lceil n(H(X)+\varepsilon) \rceil < n(H(X)+\varepsilon)+1$$

- How many bits we need to encode the sequences in its complement?

$$\le n \log |\mathcal{X}| + 1$$

- One additional bit to indicate if we are sending sequence in the typical set or outside.

AEP for Data Compression

$$
\begin{aligned}
E(l(X^n)) &= \sum_{x^n} p(x^n) l(x^n) \\
&= \sum_{x^n \in A_\epsilon^{(n)}} p(x^n) l(x^n) + \sum_{x^n \in A_\epsilon^{(n)^c}} p(x^n) l(x^n) \\
&\le \sum_{x^n \in A_\epsilon^{(n)}} p(x^n)(n(H+\epsilon)+2) + \sum_{x^n \in A_\epsilon^{(n)^c}} p(x^n)(n \log |\mathcal{X}| + 2) \\
&= \Pr\left\{A_\epsilon^{(n)}\right\}(n(H+\epsilon)+2) + \Pr\left\{A_\epsilon^{(n)^c}\right\}(n \log |\mathcal{X}| + 2 \\
&\le n(H+\epsilon) + \epsilon n(\log |\mathcal{X}|) + 2 + 2\epsilon \\
&= n(H+\epsilon'),
\end{aligned}
$$

AEP for Data Compression

- **Theorem** 3.2.1 *Let X^n be i.i.d ~ p(x). Let ε >0. Then there exist a code that maps x^n sequences of length n into binary strings such that the mapping is one-to-one* (*and therefore invertible*) *and*

$$E\left[\tfrac{1}{n} l(X^n)\right] \le H(X) + \varepsilon$$

for n sufficiently large.
- Example: The weakness of fixed-length coding
- $\begin{pmatrix} U \\ p \end{pmatrix} = \begin{pmatrix} u_1 & u_2 & u_3 & u_4 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{8} \end{pmatrix}$,Efficiency≥95%,Error rate≤10^{-6}
- , $L_{\min}$ = ?

Example

$$\forall \delta > 0,\ p\left[-\left| \frac{1}{L} \log p(X_1, X_2, \ldots X_L) - H(X) \right| > \delta \right] \le \frac{\sigma^2}{\delta^2},$$
$$\text{where } \sigma^2 = D\left[\frac{1}{n} \log\ p(X_1, X_2, \ldots X_n) \right]$$

$$H(U) = -\sum_{i=1}^{4} p_i \log p_i = 1.75 bit$$

- D[I(u_1)]=0.6875 bit^2 $\quad \sigma^2 = \frac{1}{L} 0.6875$
- According to the efficiency≥95%

$$\frac{H(U)}{H(U)+\delta} = 95\%$$

$$1.75 = 1.75 \times 0.95 + 0.95\delta \Rightarrow \delta \approx 0.092$$

Example

$$\overline{P_e} = \frac{\sigma^2}{\delta^2} = \frac{\mathrm{D}[\mathrm{I}(u_1)]}{L\delta^2} \Rightarrow L_{\min} = \frac{\mathrm{D}[\mathrm{I}(u_1)]}{\delta^2 \overline{P_e}} = \frac{0.6875}{0.092^2 \times 10^{-6}} \cong 0.813 \times 10^8$$

- Too large to use!!
- Applying AEP, we know that if $L \to \infty$, sequences can be compressed up to entropy $nH(X)$, but ...
- What about another coding?

$$\begin{pmatrix} U \\ p \end{pmatrix} = \begin{pmatrix} u_1 & u_2 & u_3 & u_4 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{8} \end{pmatrix}$$

- 0 10 110 111

Variable-Length Coding

- Length $\bar{n} = \sum_{i=1}^{4} p_i n_i = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + 2 \times \frac{1}{8} \times 3 = 1.75$
- Efficiency $\eta = \frac{H(U)}{\bar{n}} = 1$ (100%)
- Perfect
- Most source coding is variable-length in the reality
- Basic principle
- Assign short descriptions to the most frequent outcomes of the data source, and necessarily longer descriptions to the less frequent outcomes.

Variable-Length Coding

- Morse code: The Morse code is a reasonably efficient code for the English alphabet using an alphabet of four symbols: a dot, a dash, a letter space, and a word space.

Short sequences represent frequent letters (e.g., E) long sequences represent infrequent letters (e.g., Q)

Alp	code	Alp	Code	Alp	Code	Alp	Code
A	. —	B	—. . .	C	— . —.	D	— . .
E	.	F	. . —.	G	——.	H	. . . .
I	. .	J	. ———	K	— . —	L	. —. .
M	——	N	—.	O	———	P	. ——.
Q	——. —	R	. — .	S	. . .	T	—
U	. . —	V	. . . —	W	. ——	X	— . . —
Y	—. ——	Z	——. .				

Variable-Length Coding

- Some definitions
- A source code C for a random variable X is a mapping from $\mathcal{X}$, the range of X, to D^*, the set of finite-length strings of symbols from a D-ary alphabet.
- $C(x)$ denote the codeword corresponding to x
- $l(x)$ denote the length of $C(x)$.
- The expected length $L(C)$ of a source code $C(x)$ for a random variable X with probability mass function $p(x)$ is given by $L(C)=\Sigma p(x)l(x)$

Variable-Length Coding

- Non-singular codes: A code is said to be nonsingular if every element of the range of X maps into a different string in D^*; that is,

$$x \neq x' \Rightarrow C(x) \neq C(x')$$

- Uniquely decodable codes
  - Any encoded string comes from an unique sequences of source symbols.
  - for any sequence of source symbols, that sequence can be reconstructed unambiguously from the encoded bit sequence.

Variable-Length Coding

- Extension
  - Extension C^* of a code C is the mapping from finite-length strings of $\mathcal{X}$ to finite-length strings of $\mathcal{D}$ defined by

$$C(x_1x_2\ldots x_n)=C(x_1)C(x_2)\ldots C(x_n)$$

  - A code is uniquely decodable if its extension is nonsingular.
- Decoding the output from a uniquely-decodable code, and even determining whether it is uniquely decodable, can be quite complicated.

Xi'an Jiaotong University

Variable-Length Coding

- A simple class of uniquely decodable codes called prefix codes， have many advantages over other uniquely-decodable codes.
- prefix code: A code is called a prefix code or an instantaneous code if no codeword is a prefix of any other codeword.
- Example
- $C(x_1)$=0 $C(x_2)$=10 $C(x_3)$=110 $C(x_4)$=111
- 010111110101100010....

Xi'an Jiaotong University

Prefix code

- The decoder can decode each codeword of a prefix code immediately on the arrival of the last bit in that codeword.
- If a uniquely-decodable code exists with a certain set of codeword lengths, then a prefix code can easily be constructed with the same set of lengths.
- Given a probability distribution on the source symbols, it is easy to construct a prefix code of minimum expected length.

Xi'an Jiaotong University

Prefix-free codes

- Example

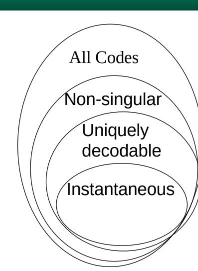
Source		1#	$2^{\#}$	$3^{\#}$	$4^{\#}$
S_1	0.5	0	0	0	0
s_2	0.25	0	1	01	10
S_3	0.125	1	00	011	110
S_4	0.125	10	11	0111	111

- 1#: singular
- 2#:Non-sigular,but not uniquely decodable
- 3#:Uniquely decodable, but not instantaneous
- 4#:Instantaneous, also prefix codes

Xi'an Jiaotong University

Prefix codes

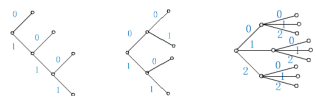
- Relationship



Xi'an Jiaotong University

Prefix codes

- Code tree
- A corresponding binary code tree can be defined which grows from a root on the left to leaves on the right representing codewords. Each branch is labelled 0 or 1 and each node represents the binary string corresponding to the branch labels from the root to that node. The tree is extended just enough to include each codeword.



Code Tree

- Relationship
- Root ⇔ codeword beginning
- # of branches ⇔ #-ary
- Node ⇔ part of a codeword
- Leaves ⇔ a codeword
- Full tree ⇔ fixed length code
- Not full tree ⇔ variable length code
- The prefix condition implies that no codeword is an ancestor of any other codeword on the tree. Hence, each codeword eliminates its descendants as possible codewords.

Kraft Inequality

- We wish to construct instantaneous codes of minimum expected length to describe a given source. It is clear that we cannot assign short codewordsto all source symbols and still be prefix. Then what is the best we can do?
- Theorem 5.2.1 (*Kraft inequality*) *For any instantaneous code (prefix code) over an alphabet of size D, the codeword lengths $l_1, l_2, \ldots, l_m$ must satisfy the inequality*

$$\sum_i D^{-l_i} \le 1$$

- *Conversly, given a set of codeword that satisfy this inequality, there exists an instantaneous code with these word lengths.*

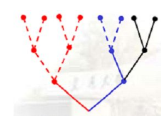
Kraft Inequality

- Proof:
  - l_{max} the length of the longest codeword of the set of codewords
  - A codeword at l_i has $D^{l_{max}-l_i}$ descendants at level l_{max}
  - total number of nodes must less than or equal to $D^{l_{max}}$
  - Hence

$$\sum_i D^{l_{max}-l_i} \le D^{l_{max}}$$

$$\Downarrow$$

$$\sum_i D^{-l_i} \le 1$$



Kraft Inequality

- Any codeword set that satisfies the prefix condition has to satisfy the Kraft inequality. (necessary condition)
  - a prefix code→Kraft inequality holds
- Kraft inequality is a sufficient condition for the existence of codeword set with the specified set of codeword lengths.
  - Kraft inequality→we can find a prefix code
  - A code satisfies Kraft inequality $\not\rightarrow$ it is a prefix code

Kraft Inequality

- Theorem 5.2.2 (*Extended Kraft inequality*): *For any countable infinite set of codewords that form a prefix code, the codeword lengths satisfy the extened Kraft inequality*

$$\sum_{i=1}^{\infty} D^{-l_i} \le 1$$

- *Conversly, given any codeword that satisfy this inequality, we can construct a prefix code with these codeword lengths.*
- Proof:codeword ⇔real number
- $0.y_1 y_2 \cdots y_{l_i} = \sum_{j=1}^{l_i} y_j D^{-j}$ occupy the interval
- $(0.y_1 y_2 \cdots y_{l_i}, 0.y_1 y_2 \cdots y_{l_i} + \frac{1}{D^{l_i}})$ disjoint $\Rightarrow \sum_{i=1}^{\infty} D^{-l_i} \le 1$

Optimal Codes

- Prefix code with the minimum expected length
  - Prefix code codeword length $l_1, l_2, \ldots, l_m$ satisfy Kraft inequality
  - The expected length $L = \sum p_i l_i$ is less than the expected length of any other prefix-free code.

This is a standard optimization problem: Minimize

$$L = \sum p_i l_i$$

over all integers $l_1, l_2, \ldots, l_m$ satisfying

$$\sum_i D^{-l_i} \le 1$$

We neglect the integer constraint on $l_1, l_2, \ldots, l_m$ and assume equality in the constraint.

Optimal Codes

- Hence, we can write the constrained minimization using Lagrange multipliers as the minimization of

$$J = \sum p_i l_i + \lambda \left(\sum D^{-l_i}\right)$$

- Differentiating with respect to l_i, we obtain

$$\frac{\partial J}{\partial l_i} = p_i - \lambda D^{-l_i} \log_e D. \quad D^{-l_i} = \frac{p_i}{\lambda \log_e D}$$

$$\lambda = 1/\log_e D, \qquad p_i = D^{-l_i}, \qquad l_i^* = -\log_D p_i.$$

$$L^* = \sum p_i l_i^* = -\sum p_i \log_D p_i = H_D(X)$$

Optimal Codes

- Theorem 5.3.1 *The expected length L of any instantaneous D-ary code for a random variable X is greater than or equal to the entropy $H_D(X)$*

$$L \ge H_D(X)$$

- *with equality if and only if* $D^{-l_i} = p_i$
- Proof

$$H_D(X) - L = \sum p_i \log_D \tfrac{1}{p_i} - \sum p_i l_i$$
$$= \sum p_i \log_D \tfrac{1}{p_i} + \sum p_i \log_D D^{-l_i}$$
$$= \sum_i p_i \log_D \frac{D^{-l_i}}{p_i}$$
$$\le \log_D \sum D^{-l_i} \le 0$$

Jenson's inequality

Kraft inequality

Optimal Codes

- Theorem 5.4.1 *Let L^* be the associated expect length of an optimal code*($L^* = \sum p_i l_i^*$). *Then*

$$H_D(X) \le L^* < H_D(X) + 1$$

- Proof: let $l_i = \left|\log_D \frac{1}{P_i}\right|$ .Then satisfies the Kraft inequality and from(5.32) we have

$$\sum D^{-\left|\log \frac{1}{P_i}\right|} \le \sum D^{-\log \frac{1}{P_i}} = \sum p_i = 1$$

$$H_D(X) \le L = \sum p_i l_i < H_D(X) + 1$$

- There is an overhead that is at most 1 bit
- We can reduce the overhead per symbol by spreading it out over many symbols.

Optimal Codes

- Consider n consequent symbols to be a supersymbol

$$L_n = \frac{1}{n} \sum p(x_1, x_2, \ldots, x_n) l(x_1, x_2, \ldots, x_n) = \frac{1}{n} El(X_1, X_2, \ldots, X_n).$$

$$H(X_1, X_2, \ldots, X_n) \le El(X_1, X_2, \ldots, X_n) < H(X_1, X_2, \ldots, X_n) + 1.$$

$$H(X) \le L_n < H(X) + \frac{1}{n}.$$

- By using large block lengths we can achieve an expected codelength per symbol arbitrarily close to the entropy.
- Theorem 5.4.2 *The minimum expected codeword length per symbol satisfies*

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} \le L_n^* \le \frac{H(X_1, X_2, \ldots, X_n)}{n} + \frac{1}{n}$$

Optimal Codes

- Coding efficiency $\eta = \frac{H(X)}{L}$

From 5.4.1, we know $\eta \le 1$, and we know, when

$$p(s_k) = D^{-n_k}, \qquad \eta = 1$$

For a sequence with N symbols

$$1 - \frac{1}{NL} < \eta \le 1$$

$$N \to \infty, \quad \eta \to 1$$

Huffman Codes

- So far, we have discussed many properties of the optimal codes.
- The problem is, how to construct an optimal code ?
- Huffman code is an optimal (shortest expected length) prefix code for a given distribution
- There are many optimal codes
- Huffman procedure constructs one such optimal code by a simple algorithm

Huffman Codes

- Lemma 5.8.1 *For any distribution, there exists an optimal instantaneous code* (*with minimum expected length*) *that satisfies the following properties:*
  1. *The lengths are ordered inversely with the probabilities* (*i.e., if* $p_j > p_k$, *then* $l_j \le l_k$).
  - *The two longest codewords have the same length.*
  - *Two of the longest codewords differ only in the last bit and correspond to two least likely symbols*
- *if* $p_j > p_k$, *then* $l_j \le l_k$ When swap codewords

1. Consider $C_m^{'}$, with the codeword j and k of C_m interchanged

Huffman Codes

$$
\begin{aligned}
L(C_m^{'}) - L(C_m) &= \sum p_i l_i^{'} - \sum p_i l_i \\
&= p_j l_k + p_k l_j - p_j l_j - p_k l_k \\
&= (p_j - p_k)(l_k - l_j)
\end{aligned}
$$

- But $p_j - p_k > 0$, and since C_m is optimal, $L(C_m^{'}) - L(C_m) \ge 0$
- Hence, we must have $l_j \le l_k$, Thus, C_m satisfies property 1
- *The two longest codewords have the same length.*
- *Two of the longest codewords differ only in the last bit and correspond to two least likely symbols*

Huffman codes

- Huffman procedure
  - Summarizing, we have shown that if $p_1 \ge p_2 \ge \ldots \ge p_m$, there exists an optimal code with $l_1 \le l_2 \le \ldots \le l_{m-1} = l_m$, and codewords $C(x_{m-1})$ and $C(x_m)$ that differ only in the last bit
  - (1) Arrange the M message in a descent order according to their probabilities.
  - (2) Assign codeword '0'and '1'to the two messages with the least probabilities, and combine these two messages into a new message with the probability the sum of their probabilities.

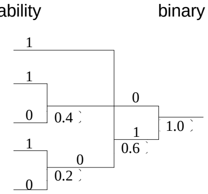
Huffman codes

- Rearrange the new messages in descent order according to their new probabilities. If the new message has the same probability with another message, put the combined message above.
- Take the procedure iteratively until all the messages are combined

- Example
  - Please find the huffman codes of the following source, and the coding efficiency

$$
\begin{matrix} \mathbf{X} \\ P(\mathbf{X}) \end{matrix} : \begin{cases} x_1, & x_2, & x_3, & x_4, & x_5 \\ 0.4, & 0.1, & 0.2, & 0.2, & 0.1 \end{cases}
$$

Huffman codes



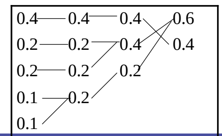
X	Probability	binary code	l_i
x_1	0.4	11	2
x_3	0.2	01	2
x_4	0.2	00	2
x_2	0.1	101	3
x_5	0.1	100	3

Huffman codes

- Average length
  $$L=\sum_i p(x_i)l_i=0.4\times 2+0.2\times 2\times 2+0.1\times 3\times 2=2.2$$
- The entropy of the source
  $$H(X)=-\sum_i p(x_i)\log p(x_i)=-0.4\log 0.4-2\times 0.2\log 0.2-2\times 0.1\log 0.1=2.122\text{bits}$$
- Coding efficiency
  $$\eta=\frac{H(X)}{L}=\frac{2.122}{2.2}=0.964=96.4\%$$

Huffman Codes

$$L(C_m)=\sum_{i=1}^{m} p_i l_i$$
$$=\sum_{i=1}^{m-2} p_i l'_i + p_{m-1}(l'_{m-1}+1)+p_m(l'_{m-1}+1)$$
$$=\sum_{i=1}^{m-1} p_i l'_i + p_{m-1}+p_m$$
$$=L(C_{m-1})+p_{m-1}+p_m$$



- Step m to m-1
- Dynamic programming
- Greedy algorithm

Huffman Code

- D>2
- Lemma: For a D-ary prefix code, the number of leaves in the code tree has $M=D+(l-1)(D-1)$, where l is a positive integer
- Proof: when l=1, $M=D$, corresponds to the case of only have 1 layer.

  when l Increases by 1, also increases (D-1) new leaves

  Hence, we have $M=D+(l-1)(D-1)$

Huffman Code

- Empty leaf, when K=4, D=3, empty leaf occurs
- Empty leaf should be the highest layer
- All the nodes should be used below the leaves, i.e., only leaves can not be fully used. Otherwise, the code can not be optimal. Hence we have to calculate the empty leaves before coding.
- Calculate the nuber of empty leaves, B

  $$B+K=D+(l-1)(D-1)$$

  From the code tree, $0\leqslant B\leqslant D-2$

  Then $0\leqslant D-B-2\leqslant D-2<D-1$

Huffman Code

$$K=(l-1)(D-1)+D-B$$
$$K-2=(l-1)(D-1)+D-B-2$$

Then, we can look on the $D-B-2$ as the remainder of $K-2$ divided by $D-1$

$$D-B-2=R_{D-1}(K-2)$$

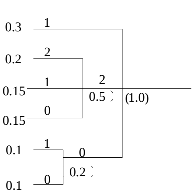
$\therefore\ B=D-2-R_{D-1}(K-2)$

Example 2: Please find the huffman code of the follow source, when D=3

$$\begin{matrix}\mathbf{X} \\ P(\mathbf{X})\end{matrix} : \left\{\begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 0.3 & 0.2 & 0.15 & 0.15 & 0.1 & 0.1 \end{matrix}\right.$$

Huffman Code

X	Probability	tenary code	l_i
x_1	0.3	1	1
x_2	0.2	22	2
x_3	0.15	21	2
x_4	0.15	20	2
x_5	0.1	01	2
x_6	0.1	00	2



Summary

- Sequence
- Source coding theorem: $N\log D \geq L[H(U)+\varepsilon]$, when $L\to\infty$, $P_e\to 0$
- AEP for data compression

$$E\left[\tfrac{1}{n}l(X^n)\right] \leq H(X)+\varepsilon$$

- Prefix code
- Code tree
- Kraft inequality

Summary

$$\sum_i D^{-l_i} \leq 1$$

  - Kraft inequality→we can find a prefix code
  - A code satisfies Kraft inequality $\not\rightarrow$ it is a prefix code
- Optimal Codes

$$H_D(X) \leq L^* < H_D(X)+1$$

- Hufman codes
  - D=2
  - D>2