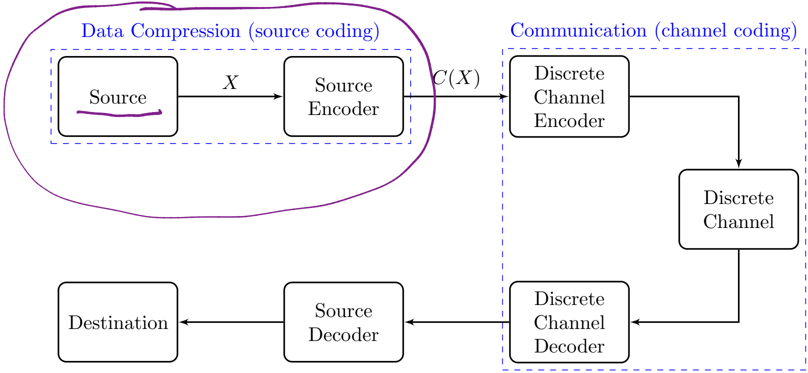
ECE 231A Discussion 3

TA: Hengjie Yang

Email: hengjie.yang@ucla.edu

04/17/2020

Why data compression?



Goal of data compression: To design an "efficient" representation of the source such that it can be losslessly reconstructed from its representation.

Source-channel separation theorem: the above two-stage method is as good as any other method of transmitting information over a noisy channel if the source generates symbols from a finite alphabet and satisfies AEP.

Source code

Notation setup:

1. $\mathcal{X}$: a finite alphabet
2. $\mathcal{D}^*$: the set of finite-length strings of symbols from a D-ary alphabet.
3. $C : \mathcal{X} \to \mathcal{D}^*$: a source code that maps a r.v. $X \in \mathcal{X}$ to $\mathcal{D}^*$
4. $C(x) \in \mathcal{D}^*$: the codeword for $x \in \mathcal{X}$;
5. $l(x)$: the length for codeword $C(x)$, $x \in \mathcal{X}$

Nonsingular: A code is nonsingular if $\forall x, x' \in \mathcal{X}$, $C(x) \neq C(x')$.

Extension: The extension C^* of a code C is given by

$$C(x_1 x_2 \cdots x_n) = C(x_1)C(x_2)\cdots C(x_n).$$

Uniquely decodable: A code is uniquely decodable if C^* is nonsingular.

Instantaneous (prefix-free): No codeword is a prefix of any other codeword.

Let $\mathcal{X} = \{a, b, c, d\}$ and $\mathcal{D} = \{0, 1\}$.

Source code 1	Source code 2	Source code 3	Source code 4
$a \to 0$	$a \to 0$	$a \to 10$	$a \to 0$
$b \to 0$	$b \to 010$	$b \to 00$	$b \to 10$
$c \to 0$	$c \to 01$	$c \to 11$	$c \to 110$
$d \to 0$	$d \to 10$	$d \to 110$	$d \to 1110$

Source code 1: singular, cannot reconstruct elements in $\mathcal{X}$.

Source code 2: nonsingular, but not uniquely decodable, e.g., 00.

Source code 3: uniquely decodable, but not prefix-free.

Source code 4: prefix-free.

~~aa~~ a d

0|10 → b

Relationship of codes:
Prefix-free codes $\subseteq$ Uniquely decodable codes $\subseteq$ Nonsingular codes $\subseteq$ All codes

Kraft inequality for prefix-free (or uniquely decodable) codes

Theorem

For any prefix-free (or uniquely decodable) code over an alphabet of size D, the codeword lengths $l_1, l_2, \ldots, l_m$, $m = |\mathcal{X}|$, must satisfy

$$\sum_{i=1}^{m} D^{-l_i} \leq 1.$$

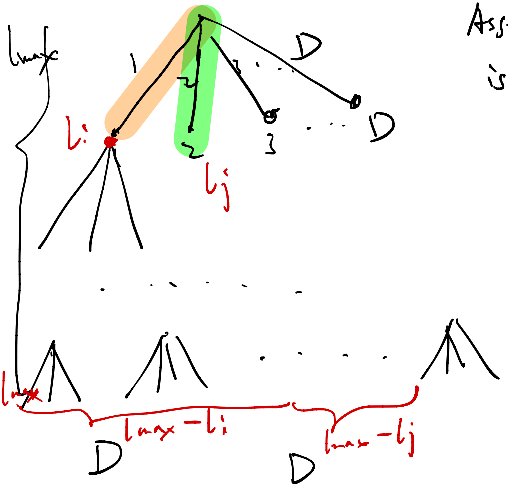
Conversely, given a set of codeword lengths that satisfy this inequality, there exists a prefix-free (or uniquely decodable) code with these word lengths.

Proof(prefix-free): Assume the longest code length is $l_{\max}$. Then level l_i occupies $D^{l_{\max}-l_i}$ nodes at level $l_{\max}$. Due to prefix-free, all these nodes are disjoint. Hence, $\sum_{i=1}^{m} D^{l_{\max}-l_i} \leq D^{l_{\max}}$.

Implications:

1. The class of uniquely decodable codes does not offer any further choices for the set of codeword lengths than the class of prefix-free codes.
2. $H(X)$ is the fundamental limit of lossless data compression.

Proof of Kraft inequality (prefix free)



Assume the longest code length is l_{max}

$$\sum_{i=1}^{m} D^{l_{max}-l_i} \le D^{l_{max}}$$

$$\Downarrow$$

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

Discrete memoryless source and optimal codes

Discrete memoryless source: a finite alphabet $\mathcal{X}$ with a fixed distribution $p(x), x \in \mathcal{X}$. For $(x_1, \ldots, x_n) \in \mathcal{X}^n$, $p(x_1, \ldots, x_n) = \prod_{i=1}^n p(x_i)$.

Expected length L: $L \triangleq \sum_{x\in\mathcal{X}} p(x)l(x)$ (Intuition: $\frac{1}{n}\sum_{i=1}^n l(x_i) \to L$)

Goal: Minimize expected length $L = \sum_{x\in\mathcal{X}} p(x)l(x)$ for prefix-free codes.

$$\begin{aligned} \min_{l(x), x\in\mathcal{X}} \quad & \sum_{x\in\mathcal{X}} p(x)l(x) \\ \text{s.t.} \quad & \sum_{x\in\mathcal{X}} D^{-l(x)} \leq 1 \end{aligned}$$

Solution: $l^*(x) = -\log_D p(x)$ and $L^* = \sum_{x\in\mathcal{X}} p(x)l^*(x) = H_D(X)$.

Theorem: Given a source with distribution $p(x)$, the expected length L of any prefix-free D-ary code satisfies

$$L \geq \sum_{x\in\mathcal{X}} p(x)\log_D \frac{1}{p(x)} = H_D(X).$$

with equality iff $D^{-l(x)} = p(x)$.

Bounds on optimal code lengths

Theorem: Given a source with distribution $p(x)$, the expected length L of the optimal prefix-free D-ary code satisfies

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

Proof: Choose $l(x) = \lceil \log_D \frac{1}{p(x)} \rceil$ (Shannon code).

$\log \frac{1}{p(x)} \le l(x) < \log \frac{1}{p(x)} + 1$

Expected codeword length per (source) symbol L_n

$$L_n \triangleq \frac{1}{n} \sum_{x^n \in \mathcal{X}^n} p(x^n) l(x^n) = \frac{1}{n} \mathbb{E}[l(X_1, \ldots, X_n)]$$

Similarly, we can choose $l(x^n) = \lceil \log_D \frac{1}{p(x^n)} \rceil$.

Theorem

The minimum expected codeword length per symbol satisfies

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

Moreover, if $X_1, \ldots, X_n$ is a stationary process,

$$\lim_{n \to \infty} L_n^* = \lim_{n \to \infty} \frac{H(X_1, \ldots, X_n)}{n} = H(\mathcal{X}).$$

Huffman coding and Shannon-Fano-Elias coding

Huffman coding:

(i) achieve the minimum possible length for a given distribution $p(x)$ and $H(X) \le L < H(X) + 1$;

(ii) If $D = 2$, combining the two least likely symbols into one symbol until we are finally left with one symbol.

(iii) If $D \ge 3$, add "dummy" symbols with prob. 0 such that the number of symbols becomes $1 + k(D-1)$. Then we perform the Huffman coding.

(iv) Cons: does not support encoding/decoding on the fly; complexity increases exponentially in length n $|\mathcal{X}|^n$ $p(x^n)$

Shannon-Fano-Elias coding

(i) A precursor of the arithemetic coding which achieves $H(X) + 1 \le L < H(X) + 2$;

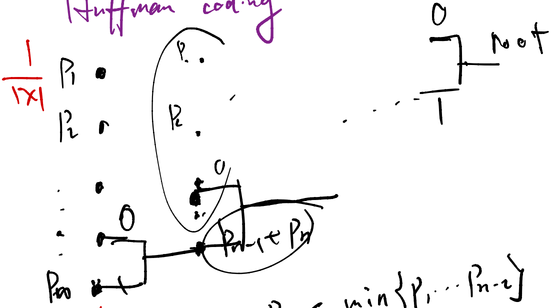
(ii) Constructs a prefix-free code $\lfloor \bar{F}(x) \rfloor_{l(x)}$ using the modified CDF $\bar{F}(x) = \sum_{a<x} p(a) + \frac{1}{2} p(x)$ and round-off length $l(x) = \lceil \log \frac{1}{p(x)} \rceil + 1$.

Remark:

(i) These schemes require the knowledge of source distribution $p(x), x \in \mathcal{X}$.

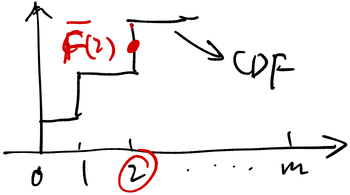
(ii) These schemes also assume all source symbols are i.i.d.

Huffman coding



Assumption: $p_{n-1}+p_n < \min\{p_1 \cdots p_{n-2}\}$

Shannon – Fano – Elias coding



Arithmetic coding and Lempel-Ziv coding

Arithmetic coding:

(i) Use a subinterval of the unit interval to represent a sequence of symbols.

(ii) Pros: Encoding is sequential; adaptive if the source model (i.e., source distribution) changes over time.

(iii) Cons: requires the knolwedge of source distribution $p_n(x)$ at each time n.

Lempel-Ziv (LZ) coding (Universal source coding schemes):

1. Sliding-window LZ (LZ77)

   flag position length of match

   (i) Algorithn: Parsing longest phrases to (F, P, L) or (F, C), then do encoding

   uncompressed uncompressed symbol

   (ii) Example: Let $W = 4$ and string: ABBABBABBBAABABA

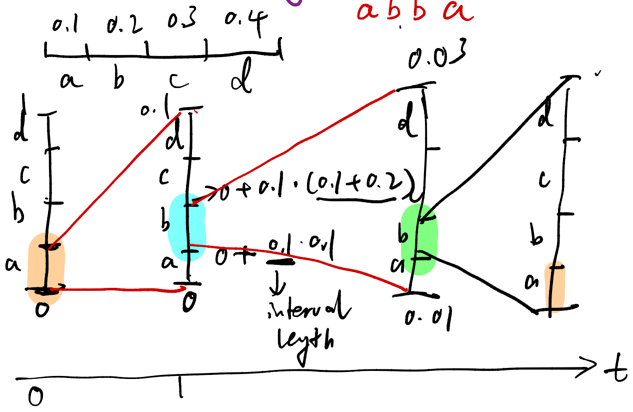
2. Tree-structured LZ (LZ78)

   (i) Algorithm: Parsing shortest phrases that have not occurred so far, then use a tree to map to (P, C)

   (ii) Example: string: ABBABBABBBAABABAA

3. Pros: LZ codes are universal codes that does not depend on the source distribution; can achieve asymptotic compression equal to $H(\mathcal{X})$.

Arithmetic coding



Sliding-window LZ

A B B A B B A B B B A A B A B A

p: the difference between x_j in sliding window and current symbol position x_i

(0, A)
(0, B)
(1, 1, 1)
(1, 3, 6)
(1, 4, 2)
(1, 1, 1)
(1, 3, 2)
(1, 2, 2)

Tree-structured LZ

A B B A B B A B B B A A B A B A A

A	(0, A)
B	(0, B)
BA	(2, A)
BB	(2, B)
AB	(1, B)
BBA	(4. A)
ABA	(5, A)
BAA	(3, A)