Group Seminar

Arithmetic Coding

WANG Yilei
Sep 14 (2018)

Outline

- Background
  - Source Coding
  - Huffman Coding
  - Limitation of Symbol Code
  - Beyond Symbol Code
- Arithmetic Coding
  - Introduction
  - Representation of Interval
  - Upper Bound
  - Integer Implementation
  - Rescaling while encoding
- Adaptive Arithmetic Coding
- Applications
  - Constant Weight Binary Code
  - Constant Weight Integer Code

Background

- Source Coding

  Source alphabet $S = \{s_1, \cdots, s_m\}$ and code alphabet $A = \{a_1, \cdots, a_n\}$.

  We would like to transmit text written in the source alphabet, but our channel accepts only code alphabetic characters.

  Common case in computer science: $A = \{0, 1\}$. *We will always assume* $A = \{0, 1\}$ *in the remaining discussion.*

Background

- Source Coding

  Source alphabet $S = \{s_1, \cdots, s_m\}$ and code alphabet $A = \{a_1, \cdots, a_n\}$.

  Encoding function: A one-to-one function $\phi: S^+ \to A^+$.

  Encoding scheme: A list of productions $\{s_i \to w_i\}(w_i \in A^+, i = 1, \cdots, m)$.

  Each encoding scheme corresponds to an encoding function by concatenation.

  Example: $S = \{a, b, c\}$, $A = \{0,1\}$, *and the scheme is*

  $$a \to 0$$
  $$b \to 10$$
  $$c \to 11,$$

  then $\phi(acbb) = 0111010$.

  Symbol code: Source code with encoding function constructed by encoding scheme.

Background

- Source Coding

Source alphabet $S = \{s_1, \cdots, s_m\}$ and code alphabet $A = \{a_1, \cdots, a_n\}$.

Encoding function: A one-to-one function $\phi: S^+ \to A^+$.

Encoding scheme: A list of productions $\{s_i \to w_i\}(w_i \in A^+, i = 1, \cdots, m)$.

Each encoding scheme corresponds to an encoding function by concatenation.

Symbol code: Source code with encoding function constructed by encoding scheme.

Prefix code: An encoding scheme that there are no $i \neq j$ such that w_i is an initial segment, or prefix (from left to right) of w_j.

Prefix code:	*Non-prefix code:*
$a \to 0$	$a \to 0$
$b \to 10$	$b \to 10$
$c \to 11$	$c \to 01$

Prefix code is quite easy to be decoded.

Background

Source Coding

Source alphabet $S = \{s_1, \cdots, s_m\}$ and code alphabet $A = \{a_1, \cdots, a_n\}$.

Encoding function: A one-to-one function $\phi: S^+ \to A^+$.

Encoding scheme: A list of productions $\{s_i \to w_i\}(w_i \in A^+, i = 1, \cdots, m)$.

Each encoding scheme corresponds to an encoding function by concatenation.

Symbol code: Source code with encoding function constructed by encoding scheme.

Prefix code: An encoding scheme that there are no $i \neq j$ such that w_i is an initial segment, or prefix (from left to right) of w_j.

Huffman Coding

Suppose each source alphabet X is generated i.i.d. with $\Pr(X = s_i) = f_i$.

Average code word length: $\bar{\ell} = \sum_{i=1}^{m} f_i \cdot |w_i|$, where $|w_i|$ is the length of w_i.

Entropy: $h = H(X) = -\sum_{i=1}^{m} f_i \log f_i$. Shannon's source coding theorem: $h \leq \bar{\ell}$.

Huffman coding: Optimal symbol code (also prefix code) with $h \leq \bar{\ell} < h + 1$.

Background

- Limitation of Symbol Code

  Consider a random source string with length k: $\boldsymbol{X} = X_1 X_2 \cdots X_k$.

  The entropy of string $\boldsymbol{X}$: $H(\boldsymbol{X}) = \sum_{i=1}^{k} H(X_i) = kh$.

  The average length of code string using Huffman coding:

$$\mathrm{E}[|\phi(\boldsymbol{X})|] = \mathrm{E}[|\phi(X_1)\cdots\phi(X_k)|] = \mathrm{E}\left(\sum_{i=1}^{K}|\phi(X_i)|\right) = \sum_{i=1}^{k}\mathrm{E}[\phi(X_i)] = k\bar{\ell}$$

  The number of overhead bits is $k(\bar{\ell} - h)$, which is linear to the length of source string when $\bar{\ell} \neq h$.

  Example: $S = \{a, b, c\}$ *with frequencies* $\{0.7, 0.2, 0.1\}$, $A = \{0,1\}$.

  Then $h = -(0.7\log 0.7 + 0.2\log 0.2 + 0.1\log 0.1) \approx 1.16$.

  The encoding scheme given by Huffman coding is $\{a \to 0, b \to 10, c \to 11\}$.

  Then $\bar{\ell} = 0.7 \times 1 + 0.2 \times 2 + 0.1 \times 2 = 1.40$.

  The number of overhead bits is about 0.24 *per source alphabet.*

Background

- Limitation of Symbol Code

  Consider a random source string with length k: $\boldsymbol{X} = X_1 X_2 \cdots X_k$.

  The entropy of string $\boldsymbol{X}$: $H(\boldsymbol{X}) = \sum_{i=1}^{k} H(X_i) = kh$.

  The average length of code string using Huffman coding:

$$\mathrm{E}[|\phi(\boldsymbol{X})|] = \mathrm{E}[|\phi(X_1) \cdots \phi(X_k)|] = \mathrm{E}\left(\sum_{i=1}^{K} |\phi(X_i)|\right) = \sum_{i=1}^{k} \mathrm{E}[\phi(X_i)] = k\bar{\ell}$$

  The number of overhead bits is $k(h - \bar{\ell})$, which is linear to the length of source string when $\bar{\ell} \neq h$.

- Beyond Symbol Code

  To reduce the number of overhead bits, we remove the restriction that encoding function ϕ is always constructed by concatenation, i.e. we don't ensure $\phi(\boldsymbol{xy}) = \phi(\boldsymbol{x})\phi(\boldsymbol{y})$ for any $\boldsymbol{x} \in S^+, \boldsymbol{y} \in S^+$ now.

  Arithmetic coding achieves totally less than 1 bit overhead for any finite source string.

Arithmetic Coding

- Introduction

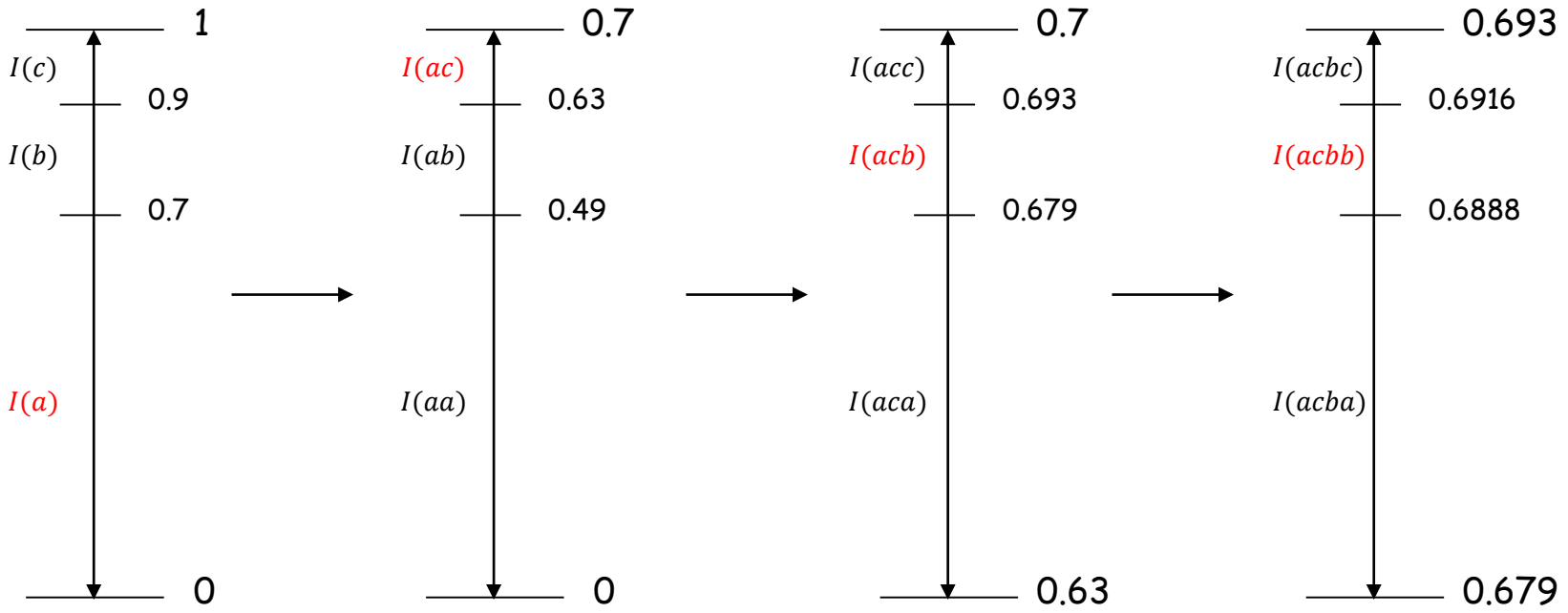
  Each source string $\boldsymbol{s} \in S^*$ corresponds to an interval $I(\boldsymbol{s}) = [\alpha(\boldsymbol{s}), \beta(\boldsymbol{s}))$.

  Initial case: For empty string ϵ, $I(\epsilon) = [0, 1)$.

  Recursion:

$$I(\boldsymbol{s}s_i) = [\alpha(\boldsymbol{s}) + \sum_{j=1}^{i-1} f_j\,(\beta(\boldsymbol{s}) - \alpha(\boldsymbol{s})), \alpha(\boldsymbol{s}) + \sum_{j=1}^{i} f_j\,(\beta(\boldsymbol{s}) - \alpha(\boldsymbol{s})))$$

  Example: $S = \{a, b, c\}$ *with frequencies* $\{0.7, 0.2, 0.1\}$, $A = \{0,1\}$. $I(acbb) = ?$



Arithmetic Coding

Introduction

Each source string $\boldsymbol{s} \in S^*$ corresponds to an interval $I(\boldsymbol{s}) = [\alpha(\boldsymbol{s}), \beta(\boldsymbol{s}))$.

Initial case: For empty string ϵ, $I(\epsilon) = [0, 1)$.

Recursion:

$$I(\boldsymbol{s}s_i) = [\alpha(\boldsymbol{s}) + \sum_{j=1}^{i-1} f_j\,(\beta(\boldsymbol{s}) - \alpha(\boldsymbol{s})), \alpha(\boldsymbol{s}) + \sum_{j=1}^{i} f_j\,(\beta(\boldsymbol{s}) - \alpha(\boldsymbol{s})))$$

Observation: Suppose $\boldsymbol{X}$ is a random source string with length k, and $\boldsymbol{x}$ is a fixed source string with length k. Then

- $\Pr(\boldsymbol{X} = \boldsymbol{x}) = \beta(\boldsymbol{x}) - \alpha(\boldsymbol{x})$,
- $\alpha(\boldsymbol{s}s_i) \geq \alpha(\boldsymbol{s})$ and $\beta(\boldsymbol{s}s_i) \leq \beta(\boldsymbol{s})$.

Representation of Interval

Let t be the integer such that $2^{-t} \leq \beta - \alpha < 2^{-t+1}$, i.e. $t = \lceil -\log(\beta - \alpha) \rceil$.

Let $c = \lceil 2^t \alpha \rceil$, then $\alpha \leq c/2^t < \beta$. We claim that the binary representation of c is the code string we need.

Example: $\alpha = 0.6888, \beta = 0.6916, \beta - \alpha = 0.0028, t = 9, c = 353$, $c/2^t = (0.101100001)_2$, *and the code string is* 101100001.

Arithmetic Coding

- Upper Bound

The average code length of length k random source string $\boldsymbol{X}$ is

$$\begin{aligned}
\mathrm{E}[|\phi(\boldsymbol{X})|] &= \sum_{\boldsymbol{s}\in S^k} \Pr(\boldsymbol{X}=\boldsymbol{s})\, t(\boldsymbol{s}) \\
&= \sum_{\boldsymbol{s}\in S^k} \Pr(\boldsymbol{X}=\boldsymbol{s})\, [-\log(\beta(\boldsymbol{s})-\alpha(\boldsymbol{s})] \\
&< -\sum_{\boldsymbol{s}\in S^k} \Pr(\boldsymbol{X}=\boldsymbol{s}) \log(\beta(\boldsymbol{s})-\alpha(\boldsymbol{s})) + \sum_{\boldsymbol{s}\in S^k} \Pr(\boldsymbol{X}=\boldsymbol{s}) \\
&< -\sum_{\boldsymbol{s}\in S^k} \Pr(\boldsymbol{X}=\boldsymbol{s}) \log \Pr(\boldsymbol{X}=\boldsymbol{s}) + 1 = H(\boldsymbol{X}) + 1.
\end{aligned}$$

- Integer Implementation

Arithmetic coding requires $O(k)$ float number operations, and $\beta(s)-\alpha(s)$ becomes smaller while the length of s grows, which needs unlimited registers, or the error will accumulates step by step.

There is an integer implementation of arithmetic coding, which needs only $O(k)$ integer operations. With $w = O(\log(k/(f_{min}\,\epsilon))$ registers, it achieves

$$\mathrm{E}[|\phi(\boldsymbol{X})|] < H(X) + 1 + \epsilon.$$

Arithmetic Coding

- Rescaling while encoding

  It is not necessary to always wait until the entire source string is scanned and processed before you have a code for it. Instead, we are able to transmit bits while encoding.

  Note that α is non-decreasing while β is non-increasing while encoding. We keep a underflow count r with initial value 0. Each time α or β is updated, we check whether

  - $\alpha \geq 1/2$. If so, we apply transformation $x \to 2x - 1$ to α and β, and transmit 10^r (1 followed by r 0's) to the decoder. Set $r = 0$.
  - $\beta < 1/2$. If so, we apply transformation $x \to 2x$ to α and β, and transmit 01^r (0 followed by r 1's) to the decoder. Set $r = 0$.
  - $1/4 \leq \alpha < 1/2 \leq \beta < 3/4$. If so, we apply transformation $x \to 2x - 1/2$ to α and β, and set $r = r + 1$.

  Example: $S = \{a, b, c\}$ *with frequencies* $\{0.7, 0.2, 0.1\}$, $A = \{0,1\}$. $I(acbb) =?$

Adaptive Arithmetic Coding

Arithmetic coding is able to deal with the model without assumption that every source alphabet is generated independent and identically.

Recursion:

$$I(\boldsymbol{s}s_i) = [\alpha(\boldsymbol{s}) + \sum_{j=1}^{i-1} f_j\,(\beta(\boldsymbol{s}) - \alpha(\boldsymbol{s})), \alpha(\boldsymbol{s}) + \sum_{j=1}^{i} f_j\,(\beta(\boldsymbol{s}) - \alpha(\boldsymbol{s})))$$

Let $\boldsymbol{s} = x_1 x_2 \cdots x_k$, and define the upper cumulative probabilities as follows:

$$Q_k(i \mid x_1, \cdots, x_k) = \sum_{j=1}^{i} \Pr(X_{k+1} = s_i \mid X_1 = x_1, \cdots X_k = x_k)$$

where $Q_k(0 \mid x_1, \cdots, x_k) = 0$. Then the recursion becomes

$$I(\boldsymbol{s}s_i) = [\alpha(\boldsymbol{s}) + Q_k(i-1 \mid x_1, \cdots, x_k)(\beta(\boldsymbol{s}) - \alpha(\boldsymbol{s})), \alpha(\boldsymbol{s}) + Q_k(i \mid x_1, \cdots, x_k)(\beta(\boldsymbol{s}) - \alpha(\boldsymbol{s})))$$

Note that both the analysis of upper bound and the integer implementation still work in adaptive arithmetic coding.

Applications

- Constant Weight Binary Code

Let $S = \{0, 1\}$. Define constant weight binary code with length n and weight w as $B(n, w) = \{\boldsymbol{s} = s_1 s_2 \cdots s_n \in S^n \mid s_1 + s_2 + \cdots s_n = w\}$.

We want to encode B with least average code length.

Let $p(\boldsymbol{s}) = \Pr(\boldsymbol{X} = \boldsymbol{s}) = \Pr(X_1 = s_1, X_2 = s_2, \cdots X_n = s_n)$, then $\sum_{\boldsymbol{s} \in B} p(\boldsymbol{s}) = 1$. Therefore, $H(\boldsymbol{X}) = -\sum_{\boldsymbol{s} \in B} p(\boldsymbol{s}) \log p(\boldsymbol{s}) \leq \log \binom{n}{w}$, and two sides are equal if and only if $p(\boldsymbol{s}) = \binom{n}{w}^{-1}$ for all $\boldsymbol{s} \in B$. If the distribution of B is unknown, we assume $p(\boldsymbol{s}) = \binom{n}{w}^{-1}$ so as to achieve worst case optimal.

Example: $n = 5, w = 3$. $\phi(10011) =?$

It is easy to see that for any $\boldsymbol{s} \in B$, $|\phi(\boldsymbol{s})| = \lceil \log \binom{n}{w} \rceil$. Therefore, $\mathrm{E}[|\phi(\boldsymbol{X})|] = \lceil H(\boldsymbol{X}) \rceil$, which is optimal.

Application

- Constant Weight Integer Code

Let $S = \{0, 1, \cdots, N\}$, where N is a large positive integer. Define constant weight integer code with length n and weight $w(\leq N)$ as $C(n, w) = \{\boldsymbol{s} = s_1 s_2 \cdots s_n \in S^n \mid s_1 + s_2 + \cdots s_n = w\}$.

We want to encode C with least average code length.

There is a bijection g: $C(n, w) \to B(n + w - 1, w)$. Let ϕ' be the optimal encoding function from $B(n + w - 1, w)$ to $\{0,1\}^+$, then $\phi = g \circ \phi'$ is an optimal encoding function from $C(n, w)$ to $\{0,1\}^+$.

We can also apply arithmetic coding directly to C if we notice the following equation holds:

$$\begin{aligned} Q_k(i \mid x_1, \cdots, x_k) &= \sum_{j=1}^{i} \Pr(X_{k+1} = i \mid X_1 = x_1, \cdots X_k = x_k, X_1 + \cdots + X_n = w) \\ &= 1 - \Pr(X_{k+1} \geq i + 1 \mid X_1 = x_1, \cdots X_k = x_k, X_1 + \cdots + X_n = w) \\ &= 1 - \binom{w - x_1 - \cdots - x_k + n - k - i - 2}{n - k - 1} \Big/ \binom{w - x_1 - \cdots - x_k + n - k - 1}{n - k - 1} \end{aligned}$$

References

- Johnson Jr P D, Harris G A, Hankerson D C. Introduction to information theory and data compression[M]. Chapman and Hall/CRC, 2003.
- MacKay D J C, Mac Kay D J C. Information theory, inference and learning algorithms[M]. Cambridge university press, 2003.
- Wikipedia contributors. "Arithmetic coding." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 4 Sep. 2018. Web. 13 Sep. 2018.
- Jones C. An efficient coding system for long source sequences[J]. IEEE Transactions on Information Theory, 1981, 27(3): 280-291.
- Ramabadran T V. A coding scheme for m-out-of-n codes[J]. IEEE Transactions on Communications, 1990, 38(8): 1156-1163.