An introduction to Information Theory

Adrish Banerjee

Department of Electrical Engineering
Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh
India

Aug. 8, 2016

Lecture #7A: Universal Source Coding-I: Lempel-Ziv Algorithm-LZ77

Outline of the lecture

- Introduction

Outline of the lecture

- Introduction
- LZ-77 encoding

Outline of the lecture

- Introduction
- LZ-77 encoding
- LZ-77 decoding

Introduction

- Source coding techniques that we have studied so far in the course, needs the knowledge of statistical model of the source.

Introduction

- Source coding techniques that we have studied so far in the course, needs the knowledge of statistical model of the source.
- Example: Huffman coding needs the probability of occurrence of source symbols.

Introduction

- Source coding techniques that we have studied so far in the course, needs the knowledge of statistical model of the source.
- Example: Huffman coding needs the probability of occurrence of source symbols.
- If the statistical knowledge of the source is not given, one has two options:

Introduction

- Source coding techniques that we have studied so far in the course, needs the knowledge of statistical model of the source.
- Example: Huffman coding needs the probability of occurrence of source symbols.
- If the statistical knowledge of the source is not given, one has two options:
  - Create a statistical model for the source.

Introduction

- Source coding techniques that we have studied so far in the course, needs the knowledge of statistical model of the source.
- Example: Huffman coding needs the probability of occurrence of source symbols.
- If the statistical knowledge of the source is not given, one has two options:
  - Create a statistical model for the source.
  - Use source coding techniques that do not need this information.

Introduction

- Source coding techniques that we have studied so far in the course, needs the knowledge of statistical model of the source.
- Example: Huffman coding needs the probability of occurrence of source symbols.
- If the statistical knowledge of the source is not given, one has two options:
  - Create a statistical model for the source.
  - Use source coding techniques that do not need this information.
- Universal source coding techniques are source coding procedures that do not depend on knowledge of statistical model of the source.

Introduction

- Variants of LZ77 algorithm used in gzip, png.

Introduction

- Variants of LZ77 algorithm used in gzip, png.
- Variants of LZW algorithm used in zip, pkzip, compress, gif, tiff.

Notations

- $\{0, 1, \cdots, q-1\}$: source alphabet, where q denotes cardinality of the source alphabet.

Notations

- $\{0, 1, \cdots, q-1\}$: source alphabet, where q denotes cardinality of the source alphabet.
- w_k: source output at time k.

Notations

- $\{0, 1, \cdots, q-1\}$: source alphabet, where q denotes cardinality of the source alphabet.
- w_k: source output at time k.
- $\{w_k\}_{k=-\infty}^{\infty}$: source output over all time.

Notations

- $\{0, 1, \cdots, q-1\}$: source alphabet, where q denotes cardinality of the source alphabet.
- w_k: source output at time k.
- $\{w_k\}_{k=-\infty}^{\infty}$: source output over all time.
- $W_1^N = \{w_k\}_{k=1}^{N}$: string to be encoded

Notations

- $\{0, 1, \cdots, q-1\}$: source alphabet, where q denotes cardinality of the source alphabet.
- w_k: source output at time k.
- $\{w_k\}_{k=-\infty}^{\infty}$: source output over all time.
- $W_1^N = \{w_k\}_{k=1}^{N}$: string to be encoded
- $W_i^j = \{w_k\}_{k=i}^{j}$: substring starting at w_i, and ending at w_j.

Notations

- $\{0, 1, \cdots, q-1\}$: source alphabet, where q denotes cardinality of the source alphabet.
- w_k: source output at time k.
- $\{w_k\}_{k=-\infty}^{\infty}$: source output over all time.
- $W_1^N = \{w_k\}_{k=1}^{N}$: string to be encoded
- $W_i^j = \{w_k\}_{k=i}^{j}$: substring starting at w_i, and ending at w_j.
- $W_i^j = W_i^{j-1} w_j$: substring partitioning; W_i^{j-1} is the prefix and w_j is the innovation symbol.

Notations

- $\{0, 1, \cdots, q-1\}$: source alphabet, where q denotes cardinality of the source alphabet.
- w_k: source output at time k.
- $\{w_k\}_{k=-\infty}^{\infty}$: source output over all time.
- $W_1^N = \{w_k\}_{k=1}^{N}$: string to be encoded
- $W_i^j = \{w_k\}_{k=i}^{j}$: substring starting at w_i, and ending at w_j.
- $W_i^j = W_i^{j-1} w_j$: substring partitioning; W_i^{j-1} is the prefix and w_j is the innovation symbol.
- $\{C_l\}_{l=0}^{L}$: string of codewords that uniquely defines a substring Y_l of W_1^N.

LZ-77 Encoding

- This procedure first described by Jacob Ziv and Abraham Lempel in 1977.
- This is also known as sliding-window LZ algorithm.

Algorithm Objective:

- String to be coded is given by W_1^N.

LZ-77 Encoding

- This procedure first described by Jacob Ziv and Abraham Lempel in 1977.
- This is also known as sliding-window LZ algorithm.

Algorithm Objective:

- String to be coded is given by W_1^N.
- Encoded sequence given by $\{C_l\}_{l=0}^{L}$ that uniquely define W_1^N.

LZ-77 Encoding

- This procedure first described by Jacob Ziv and Abraham Lempel in 1977.
- This is also known as sliding-window LZ algorithm.

Algorithm Objective:

- String to be coded is given by W_1^N.
- Encoded sequence given by $\{C_l\}_{l=0}^{L}$ that uniquely define W_1^N.
- Each codeword C_l uniquely defines a substring Y_l of W_1^N.

LZ-77 Encoding

- This procedure first described by Jacob Ziv and Abraham Lempel in 1977.
- This is also known as sliding-window LZ algorithm.

Algorithm Objective:

- String to be coded is given by W_1^N.
- Encoded sequence given by $\{C_l\}_{l=0}^{L}$ that uniquely define W_1^N.
- Each codeword C_l uniquely defines a substring Y_l of W_1^N.
- The length of substring Y_l varies with l.

LZ-77 Encoding

- This procedure first described by Jacob Ziv and Abraham Lempel in 1977.
- This is also known as sliding-window LZ algorithm.

Algorithm Objective:

- String to be coded is given by W_1^N.
- Encoded sequence given by $\{C_l\}_{l=0}^{L}$ that uniquely define W_1^N.
- Each codeword C_l uniquely defines a substring Y_l of W_1^N.
- The length of substring Y_l varies with l.
- Objective is to minimize the number of binary symbols used to represent $\{C_l\}_{l=0}^{L}$.

LZ-77 Encoding

Algorithm Details:

- Dictionary based source coding technique.

LZ-77 Encoding

Algorithm Details:

- Dictionary based source coding technique.
- A substring in the source is replaced by a codeword that identifies the substring in a dictionary.

LZ-77 Encoding

Algorithm Details:

- Dictionary based source coding technique.
- A substring in the source is replaced by a codeword that identifies the substring in a dictionary.
- Each codeword C_l except the zeroth codeword is the concatenation of two parameters L_l and m_l.

LZ-77 Encoding

Algorithm Details:

- Dictionary based source coding technique.
- A substring in the source is replaced by a codeword that identifies the substring in a dictionary.
- Each codeword C_l except the zeroth codeword is the concatenation of two parameters L_l and m_l.
- The codewords specify two types of information:

LZ-77 Encoding

Algorithm Details:

- Dictionary based source coding technique.
- A substring in the source is replaced by a codeword that identifies the substring in a dictionary.
- Each codeword C_l except the zeroth codeword is the concatenation of two parameters L_l and m_l.
- The codewords specify two types of information:
  - Pointers to previous positions in the dictionary. m_l is related to the position in the dictionary where the substring match starts.

LZ-77 Encoding

Algorithm Details:

- Dictionary based source coding technique.
- A substring in the source is replaced by a codeword that identifies the substring in a dictionary.
- Each codeword C_l except the zeroth codeword is the concatenation of two parameters L_l and m_l.
- The codewords specify two types of information:
  - Pointers to previous positions in the dictionary. m_l is related to the position in the dictionary where the substring match starts.
  - Length of the text to be copied from the past given by L_l.

LZ-77 Encoding

Algorithm Details:

- Dictionary based source coding technique.
- A substring in the source is replaced by a codeword that identifies the substring in a dictionary.
- Each codeword C_l except the zeroth codeword is the concatenation of two parameters L_l and m_l.
- The codewords specify two types of information:
  - Pointers to previous positions in the dictionary. m_l is related to the position in the dictionary where the substring match starts.
  - Length of the text to be copied from the past given by L_l.
- The string coded is used as a dictionary.

LZ-77 Encoding

Algorithm Details:

- Dictionary based source coding technique.
- A substring in the source is replaced by a codeword that identifies the substring in a dictionary.
- Each codeword C_l except the zeroth codeword is the concatenation of two parameters L_l and m_l.
- The codewords specify two types of information:
  - Pointers to previous positions in the dictionary. m_l is related to the position in the dictionary where the substring match starts.
  - Length of the text to be copied from the past given by L_l.
- The string coded is used as a dictionary.
- The parameter n_w denotes the window size of the procedure.

LZ-77 Encoding

Algorithm Details:

- Dictionary based source coding technique.
- A substring in the source is replaced by a codeword that identifies the substring in a dictionary.
- Each codeword C_l except the zeroth codeword is the concatenation of two parameters L_l and m_l.
- The codewords specify two types of information:
  - Pointers to previous positions in the dictionary. m_l is related to the position in the dictionary where the substring match starts.
  - Length of the text to be copied from the past given by L_l.
- The string coded is used as a dictionary.
- The parameter n_w denotes the window size of the procedure.
- n_w is typically a power of 2.

LZ-77 Encoding

- Both encoder and decoder maintain a dictionary $D_1^{n_w}$ containing the most recent n_w encoded source symbols.

LZ-77 Encoding

- Both encoder and decoder maintain a dictionary $D_1^{n_w}$ containing the most recent n_w encoded source symbols.
- When encoding a substring starting with source symbol w_k, the dictionary contains the source symbols $W_{k-n_w}^{k-1}$.

LZ-77 Encoding

- Both encoder and decoder maintain a dictionary $D_1^{n_w}$ containing the most recent n_w encoded source symbols.
- When encoding a substring starting with source symbol w_k, the dictionary contains the source symbols $W_{k-n_w}^{k-1}$.
- Initially dictionary contains the first n_w symbols of W_1^N.

LZ-77 Encoding

- Both encoder and decoder maintain a dictionary $D_1^{n_w}$ containing the most recent n_w encoded source symbols.
- When encoding a substring starting with source symbol w_k, the dictionary contains the source symbols $W_{k-n_w}^{k-1}$.
- Initially dictionary contains the first n_w symbols of W_1^N.
- First codeword C_0 is the initial dictionary, $W_1^{n_w}$.

LZ-77 Encoding

- Both encoder and decoder maintain a dictionary $D_1^{n_w}$ containing the most recent n_w encoded source symbols.
- When encoding a substring starting with source symbol w_k, the dictionary contains the source symbols $W_{k-n_w}^{k-1}$.
- Initially dictionary contains the first n_w symbols of W_1^N.
- First codeword C_0 is the initial dictionary, $W_1^{n_w}$.
- Second codeword C_1 is found by searching for largest substring Y_1, beginning with w_{n_w+1} such that Y_1 starts in the dictionary and ends $L_1 \geq 1$ time units later.

LZ-77 Encoding

- Both encoder and decoder maintain a dictionary $D_1^{n_w}$ containing the most recent n_w encoded source symbols.
- When encoding a substring starting with source symbol w_k, the dictionary contains the source symbols $W_{k-n_w}^{k-1}$.
- Initially dictionary contains the first n_w symbols of W_1^N.
- First codeword C_0 is the initial dictionary, $W_1^{n_w}$.
- Second codeword C_1 is found by searching for largest substring Y_1, beginning with w_{n_w+1} such that Y_1 starts in the dictionary and ends $L_1 \geq 1$ time units later.
- The index of the dictionary symbol where the copy of Y_1 begins is denoted by p.

LZ-77 Encoding

- $m_1 = n_w - p$.

LZ-77 Encoding

- $m_1 = n_w - p$.
- When there are multiple choices for m_1, usually smallest m_1 is chosen.

LZ-77 Encoding

- $m_1 = n_w - p$.
- When there are multiple choices for m_1, usually smallest m_1 is chosen.
- After determining C_1, the dictionary is updated by deleting the L_1 oldest entries and adding the L_1 most recently encoded symbols.

LZ-77 Encoding

- $m_1 = n_w - p$.
- When there are multiple choices for m_1, usually smallest m_1 is chosen.
- After determining C_1, the dictionary is updated by deleting the L_1 oldest entries and adding the L_1 most recently encoded symbols.
- This is similar to sliding window of the dictionary to the right by L_1 shifts.

LZ-77 Encoding

- $m_1 = n_w - p$.
- When there are multiple choices for m_1, usually smallest m_1 is chosen.
- After determining C_1, the dictionary is updated by deleting the L_1 oldest entries and adding the L_1 most recently encoded symbols.
- This is similar to sliding window of the dictionary to the right by L_1 shifts.
- Third codeword C_2 is found by searching for largest substring Y_2, beginning with $w_{n_w+L_1+1}$ such that Y_2 starts in the dictionary and ends $L_2 \geq 1$ time units later. m_2 is calculated.

LZ-77 Encoding

- $m_1 = n_w - p$.
- When there are multiple choices for m_1, usually smallest m_1 is chosen.
- After determining C_1, the dictionary is updated by deleting the L_1 oldest entries and adding the L_1 most recently encoded symbols.
- This is similar to sliding window of the dictionary to the right by L_1 shifts.
- Third codeword C_2 is found by searching for largest substring Y_2, beginning with $w_{n_w+L_1+1}$ such that Y_2 starts in the dictionary and ends $L_2 \geq 1$ time units later. m_2 is calculated.
- Dictionary is then updated.

LZ-77 Encoding

- $m_1 = n_w - p$.
- When there are multiple choices for m_1, usually smallest m_1 is chosen.
- After determining C_1, the dictionary is updated by deleting the L_1 oldest entries and adding the L_1 most recently encoded symbols.
- This is similar to sliding window of the dictionary to the right by L_1 shifts.
- Third codeword C_2 is found by searching for largest substring Y_2, beginning with $w_{n_w+L_1+1}$ such that Y_2 starts in the dictionary and ends $L_2 \geq 1$ time units later. m_2 is calculated.
- Dictionary is then updated.
- This process is repeated until we have encoded the full source sequence W_1^N.

LZ-77 Encoding

Problem: To encode the following sequence

$$W_1^N = \{00101001000000110110110000\}$$

using LZ-77 algorithm, $n_w = 8$.

Solution: Given below is the encoded coded output $\{C_l\}_{l=0}^{6}$.

l	$D_l^{n_w}$	Y_l	L_l	m_l	C_l	Dictionary and phase locations
0		{00101001}			{00101001}	
1	{00101001}	{00}	2	2	[2,2]	{:00101001:0000011011011000}
2	{10100100}	{000}	3	0	[3,0]	{00:10100100 : 00011011011000}
3	{00100000}	1	1	5	[1,5]	{00101:00100000:11011011000}
4	{01000001}	10	2	6	[2,6]	{001010:01000001:11011011000}
5	{00000110}	{110110}	6	2	[6,2]	{00101001:00000110 : 11011000}
6	{10110110}	{00}	2	0	[2,0]	{00101001000001:10110110 : 00}

LZ-77 Encoding

Encoding of m_l and L_l.

- A comma free encoding is a binary encoding method such that decoder can directly identify the beginning of a new codeword from the binary sequence.

LZ-77 Encoding

Encoding of m_l and L_l.

- A comma free encoding is a binary encoding method such that decoder can directly identify the beginning of a new codeword from the binary sequence.
- Wyner-Ziv Comma free encoding: For any positive integer k, we denote the binary representation of k by $b(k)$ and $|b(k)| = \lceil \log_2(k+1) \rceil$ to be the number of bits in this binary representation.

LZ-77 Encoding

Encoding of m_l and L_l.

- A comma free encoding is a binary encoding method such that decoder can directly identify the beginning of a new codeword from the binary sequence.
- Wyner-Ziv Comma free encoding: For any positive integer k, we denote the binary representation of k by $b(k)$ and $|b(k)| = \lceil \log_2(k+1) \rceil$ to be the number of bits in this binary representation.
- Also, we define a binary sequence of $k-1$ zeros followed by a 1 by $u(k) = 0^{k-1}1$.

LZ-77 Encoding

Encoding of m_l and L_l.

- A comma free encoding is a binary encoding method such that decoder can directly identify the beginning of a new codeword from the binary sequence.
- Wyner-Ziv Comma free encoding: For any positive integer k, we denote the binary representation of k by $b(k)$ and $|b(k)| = \lceil \log_2(k+1) \rceil$ to be the number of bits in this binary representation.
- Also, we define a binary sequence of $k-1$ zeros followed by a 1 by $u(k) = 0^{k-1}1$.
- Then, comma free coding of L_l is given by

$$e(L_l) = \hat{e}(|b(L_l)|) \cdot b(L_l)$$

where

$$\hat{e}(p_l) = u(|b(p_l)|) \cdot b(p_l)$$

LZ-77 Encoding

Encoding of m_l and L_l.

- A comma free encoding is a binary encoding method such that decoder can directly identify the beginning of a new codeword from the binary sequence.
- Wyner-Ziv Comma free encoding: For any positive integer k, we denote the binary representation of k by $b(k)$ and $|b(k)| = \lceil \log_2(k+1) \rceil$ to be the number of bits in this binary representation.
- Also, we define a binary sequence of $k-1$ zeros followed by a 1 by $u(k) = 0^{k-1}1$.
- Then, comma free coding of L_l is given by

$$e(L_l) = \hat{e}(|b(L_l)|) \cdot b(L_l)$$

where

$$\hat{e}(p_l) = u(|b(p_l)|) \cdot b(p_l)$$

- m_l is represented using $\log_2 n_w$ binary bits.

LZ-77 Encoding

Comma free coding of the codewords for the previous example.

l	L_l	$b(L_l)$	$\lvert b(L_l)\rvert = k$	$b(k)$	$\lvert b(k)\rvert$	$u(\lvert b(k)\rvert)$	$e(L_l)$	m_l	$b(m_l)$	C_l
0							{00101001}			{00101001}
1	2	{10}	2	{10}	2	{01}	{011010}	2	{010}	{011010010}
2	3	{11}	2	{10}	2	{01}	{011011}	0	{000}	{011011000}
3	1	{1}	1	{1}	1	{1}	{111}	5	{101}	{111101}
4	2	{10}	2	{10}	2	{01}	{011010}	6	{110}	{011010110}
5	6	{110}	3	{11}	2	{01}	{0111110}	2	{010}	{0111110010}
6	2	{10}	2	{10}	2	{01}	{011010}	0	{000}	{011010000}

LZ-77 Decoding

- Decoder receives the encoded sequence and knows the window size n_w.

LZ-77 Decoding

- Decoder receives the encoded sequence and knows the window size n_w.
- First n_w bits are directly stored in the dictionary.

LZ-77 Decoding

- Decoder receives the encoded sequence and knows the window size n_w.
- First n_w bits are directly stored in the dictionary.
- To determine, L_1, the decoder reads a sequence of 0's followed by the first 1 to determine the length of encoding of p_1, which in term can be used to determine the number of bits used to represent the length of L_1 and subsequently the L_1.

LZ-77 Decoding

- Decoder receives the encoded sequence and knows the window size n_w.
- First n_w bits are directly stored in the dictionary.
- To determine, L_1, the decoder reads a sequence of 0's followed by the first 1 to determine the length of encoding of p_1, which in term can be used to determine the number of bits used to represent the length of L_1 and subsequently the L_1.
- Having determined L_1, the decoder knows that next $\log_2 n_w$ binary bits are for representing m_1.

LZ-77 Decoding

- Once L_1 and m_1 are known, the decoder can find out the phrase Y_1 from the dictionary.

LZ-77 Decoding

- Once L_1 and m_1 are known, the decoder can find out the phrase Y_1 from the dictionary.
- The phrase Y_1 is appended to the dictionary, and that is the decoded sequence so far.

LZ-77 Decoding

- Once L_1 and m_1 are known, the decoder can find out the phrase Y_1 from the dictionary.
- The phrase Y_1 is appended to the dictionary, and that is the decoded sequence so far.
- Dictionary is updated by deleting the oldest L_1 entries and appending the phrase Y_1 to the dictionary.

LZ-77 Decoding

- Once L_1 and m_1 are known, the decoder can find out the phrase Y_1 from the dictionary.
- The phrase Y_1 is appended to the dictionary, and that is the decoded sequence so far.
- Dictionary is updated by deleting the oldest L_1 entries and appending the phrase Y_1 to the dictionary.
- This procedure is repeated until the original sequence is completely reconstructed.

LZ-77 Decoding

- Received Code Sequence

  0010100101101001001101100011110101101011001111100100110100000

LZ-77 Decoding

- Received Code Sequence

  0010100101101001001101100011110101101011001111100100110100000

- After decoding of comma free codes, we get the initial string as $\{00101001\}$ and $[L_i, m_i]$ as

$$[2,2],[3,0],[1,5],[2,6],[6,2],[2,0]$$

LZ-77 Decoding

- Received Code Sequence

  00101001011010010011011000111101011010110011111001001101000 0

- After decoding of comma free codes, we get the initial string as $\{00101001\}$ and $[L_i, m_i]$ as

$$[2,2],[3,0],[1,5],[2,6],[6,2],[2,0]$$

- After decoding, we get the decoded sequence as

$$001010010000011011011000$$