Context-based Encoding
Prediction with Partial Matching (PPM)

Computer Science Department
CS4481b/9628b: Image Compression
Winter 2017
Instructor: Mahmoud R. El-Sakka
Office: MC-419
Email: elsakka@csd.uwo.ca
Phone: 519-661-2111 x86996

Probability Models

- Model 1:
  - Pixels are independent of each other
  - Pixels have identical probability

Ignorance model

- Model 2:
  - Pixels are independent of each other
  - Pixels have *none-identical* probabilities

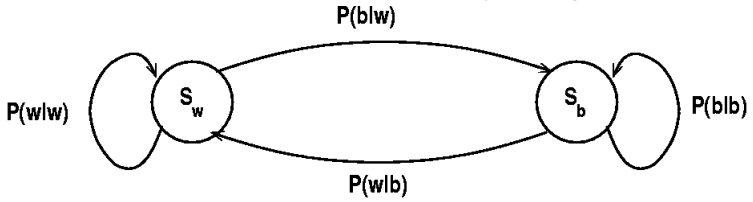
Probability model

- Model 3:
  - Pixels are *dependent* on each other
  - Pixels have *none-identical* probabilities

Markov model

Markov Models

- Two state Markov model (for binary image)



- This model can be generalize to any number of states
- Some compression schemes can easily accommodate more than one probability distribution at the same time (composite source)

Introduction

- So far, we learned that we get more compression when the message that is being encoded has a more skewed set of probability
  - "***Skewed***" here means that certain symbols occur with much higher probability than others in the sequence to be encoded
- Hence, it makes sense to look for ***ways*** to utilize various probability models that may result in greater skew
- One very effective way to do so is to look at the probability of occurrence of a pixel (or a letter in case of text files) in the ***context*** in which it occurs
  - We should not look at each symbol in a sequence as if it has just happened out of the blue
  - We should examine the history of the sequence before determining the likely probabilities of different values that the symbol can take
  - By knowing the context in which a symbol occurs, we can guess with a much greater likelihood of success what the value of the symbol would be

Introduction

- Consider the encoding of the word *probability* and suppose we have encoded the first four letters and we want to encode the fifth letter, *a*
- If we ignore the first four letters, the probability of the letter *a* is about 0.06
- If we use the information that the previous letter is *b*, this
  - reduces the probability of occurrence of several letters such as *q* and *z* and
  - boosts the probability of occurrence of several letters such as *a*
- In this example, *b* would be the *first-order context* for letter *a* and *ob* would be the *second-order context* for letter *a*, and so on
- Using high-order contexts in which *a* occurs will generally increase the probability of the occurrence of *a* in this example
- Considering all possible high-order contexts would be an overwhelming amount of information
  - If the alphabet size is M, the number of first-order contexts will be M, the number of second-order contexts will be M^2, and so on (too too much)
- ***Prediction with Partial Matching*** (PPM) is a set of compression algorithms that resolve this problem in a very simple way

Introduction

- ***Prediction with Partial Matching*** (***PPM***) is a set of compression algorithms that were proposed by Cleary and Witten, 1984, to be used with the arithmetic encoding method to create a single pass text compressor
- PPM is a context-based compression scheme where the symbol is encoded based on its context
  - The context of the symbol is taken into consideration when calculating its probability
  - As the size of the context increase, we can achieve more compression (*up to a certain limit*)
  - However large contexts would require us to estimate and store large number of conditional probabilities

Introduction

- The basic idea of PPM is to use large contexts and compute the probabilities on the fly (*a single pass text compressor*)
  - We only need to store *the probabilities and the contexts that* ***have occurred*** in the sequence being encoded

PPM Algorithm

- To encode a certain symbol we first start by looking for this symbol in the largest context
- If the symbol was not found, a *escape symbol* is sent and the size of the context is reduced
- This process continues until either
  - ***The symbol is found in a context***
    The symbol is encoded using the set of probabilities in this context
  - ***The symbol is not found in any context*** (called *-1 order context*)
    The symbol is encoded with probability of 1/M where M is the size of the source alphabet

Example: thisΔisΔtheΔtithe

- We are trying to encoding:

thisΔisΔtheΔtithe

- The maximum context size is 2
- Assume that we have already encoded the first 7 letters *"thisΔis"*
- We have the following counts

Example: thisΔisΔtheΔtithe

- -1 order context

Letter	Count	Cum_count
t	1	1
h	1	2
i	1	3
s	1	4
e	1	5
Δ	1	6
Total count		6

Example: thisΔisΔtheΔtithe

- Zero order context

Letter	Count	Cum_count
t	1	1
h	1	2
i	2	4
s	2	6
Δ	1	7
<Esc>	1	8
Total count		8

Example: thisΔisΔtheΔtithe

- 1st order context

Context	Letter	Count	Cum_count
t	h	1	1
	<Esc>	1	2
	Total Count		2
h	i	1	1
	<Esc>	1	2
	Total Count		2
i	s	2	2
	<Esc>	1	3
	Total Count		3
s	Δ	1	1
	<Esc>	1	2
	Total Count		2
Δ	i	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- 2nd order context

Context	Letter	Count	Cum_count
th	i	1	1
	<Esc>	1	2
	Total Count		2
hi	s	1	1
	<Esc>	1	2
	Total Count		2
is	Δ	1	1
	<Esc>	1	2
	Total Count		2
sΔ	i	1	1
	<Esc>	1	2
	Total Count		2
Δi	s	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- 2nd order context

Context	Letter	Count	Cum_count
th	i	1	1
	<Esc>	1	2
	Total Count		2
hi	s	1	1
	<Esc>	1	2
	Total Count		2
is	Δ	1	1
	<Esc>	1	2
	Total Count		2
sΔ	i	1	1
	<Esc>	1	2
	Total Count		2
Δi	s	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- 2nd order context

Context	Letter	Count	Cum_count
th	i	1	1
	<Esc>	1	2
	Total Count		2
hi	s	1	1
	<Esc>	1	2
	Total Count		2
is	Δ	1	1
	<Esc>	1	2
	Total Count		2
sΔ	i	1	1
	<Esc>	1	2
	Total Count		2
Δi	s	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- We use the arithmetic encoder to encode the letter
- Using the probabilities, the upper and lower limits of the encoder are calculated as follows

$$new\_L = current\_L + \left\lfloor (current\_U - current\_L + 1) \times \frac{cumulative\_count(symbol - 1)}{Total\_count} \right\rfloor$$

$$new\_U = current\_L + \left\lfloor (current\_U - current\_L + 1) \times \frac{cumulative\_count(symbol)}{Total\_count} \right\rfloor - 1$$

- Assume that length of word in our arithmetic encoder is 6
- Initially
- L = 000000 = 0
- U =111111 = 63

Example: thisΔisΔtheΔtithe

- to encode 'Δ'

$$L = 0 + \left\lfloor (63 - 0 + 1) \times \frac{0}{2} \right\rfloor = 0 = 000000$$

$$U = 0 + \left\lfloor (63 - 0 + 1) \times \frac{1}{2} \right\rfloor - 1 = 31 = 011111$$

- MSB in both upper and lower is 0
- send 0 to the decoder
- shift left and put 0 in LSB of L
- shift left and put 1 in LSB of U

Summary

- Transmitted sequence: 0
- L: 000000 = 0
- U: 111111 = 63

Example: thisΔisΔtheΔtithe

- All context tables will be updated accordingly

Example: thisΔisΔtheΔtithe

- zero order context

Letter	Count	Cum_count
t	1	1
h	1	2
i	2	4
s	2	6
Δ	*2*	*8*
<Esc>	1	*9*
Total count		*9*

Example: thisΔisΔtheΔtithe

- 1st order context

Context	Letter	Count	Cum_count
t	h	1	1
	<Esc>	1	2
	Total Count		2
h	i	1	1
	<Esc>	1	2
	Total Count		2
i	s	2	2
	<Esc>	1	3
	Total Count		3
s	Δ	*2*	*2*
	<Esc>	1	*3*
	Total Count		*3*
Δ	I	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- 2nd order context

Context	Letter	Count	Cum_count
th	i	1	1
	<Esc>	1	2
	Total Count		2
hi	s	1	1
	<Esc>	1	2
	Total Count		2
is	Δ	*2*	*2*
	<Esc>	1	*3*
	Total Count		*3*
sΔ	i	1	1
	<Esc>	1	2
	Total Count		2
Δi	s	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- The decoder will do the same thing, i.e.,
  - Identify that *'is'* is the 2nd order context which will be considered
  - When decoding the symbol Δ, the decoder updates ***ALL*** context tables accordingly

Example: thisΔisΔtheΔtithe

- Next symbol is *t*
- Both encoder and decoder consider the 2nd order context before *'t'* which is *'sΔ'*

Example: thisΔisΔtheΔtithe

- 2nd order context

Context	Letter	Count	Cum_count
th	i	1	1
	<Esc>	1	2
	Total Count		2
hi	s	1	1
	<Esc>	1	2
	Total Count		2
is	Δ	1	1
	<Esc>	1	2
	Total Count		2
sΔ	i	1	1
	<Esc>	1	2
	Total Count		2
Δi	s	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- 2nd order context

Context	Letter	Count	Cum_count
th	i	1	1
	<Esc>	1	2
	Total Count		2
hi	s	1	1
	<Esc>	1	2
	Total Count		2
is	Δ	1	1
	<Esc>	1	2
	Total Count		2
sΔ	i	1	1
	<Esc>	1	2
	Total Count		2
Δi	s	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- We adjust the boundaries of the interval to encode <*Esc*> in the context *'sΔ'*

$$L = 0 + \left\lfloor (63 - 0 + 1) \times \frac{1}{2} \right\rfloor = 32 = 100000$$

$$U = 0 + \left\lfloor (63 - 0 + 1) \times \frac{2}{2} \right\rfloor - 1 = 63 = 111111$$

- MSB in both upper and lower is 1
- send 1 to the decoder
- shift left and put 0 in LSB of L
- shift left and put 1 in LSB of U

Summary

- Transmitted sequence: 01
- L: 000000 = 0
- U: 111111 = 63

Example: thisΔisΔtheΔtithe

- Receiving *<Esc>*, the decoder knows that the next symbol is *not* found in the 2nd order context
- Both the encoder and the decoder now look in the 1st order context which is 'Δ'

Example: thisΔisΔtheΔtithe

- 1st order context

Context	Letter	Count	Cum_count
t	h	1	1
	<Esc>	1	2
	Total Count		2
h	i	1	1
	<Esc>	1	2
	Total Count		2
i	s	2	2
	<Esc>	1	3
	Total Count		3
s	Δ	2	2
	<Esc>	1	3
	Total Count		3
Δ	i	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- 1st order context

Context	Letter	Count	Cum_count
t	h	1	1
	<Esc>	1	2
	Total Count		2
h	i	1	1
	<Esc>	1	2
	Total Count		2
i	s	2	2
	<Esc>	1	3
	Total Count		3
s	Δ	2	2
	<Esc>	1	3
	Total Count		3
Δ	i	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- We adjust the boundaries of the interval to encode <*Esc*> in the context *'Δ'*

$$L = 0 + \left\lfloor (63 - 0 + 1) \times \frac{1}{2} \right\rfloor = 32 = 100000$$

$$U = 0 + \left\lfloor (63 - 0 + 1) \times \frac{2}{2} \right\rfloor - 1 = 63 = 111111$$

- MSB in both upper and lower is 1
- send 1 to the decoder
- shift left and put 0 in LSB of L
- shift left and put 1 in LSB of U

Summary

- Transmitted sequence: 011
- L: 000000 = 0
- U: 111111 = 63

Example: thisΔisΔtheΔtithe

- Receiving *<Esc>*, the decoder knows that the next symbol is *not* found in the 1st order context
- Both the encoder and the decoder now look in the zero order context

Example: thisΔisΔtheΔtithe

- zero order context

Letter	Count	Cum_count
t	1	1
h	1	2
i	2	4
s	2	6
Δ	2	8
<Esc>	1	9
Total count		9

Example: this∆is∆the∆tithe

- We adjust the boundaries of the interval to encode *'t'* in the zero order context

$$L = 0 + \left\lfloor (63 - 0 + 1) \times \frac{0}{9} \right\rfloor = 0 = 000000$$

$$U = 0 + \left\lfloor (63 - 0 + 1) \times \frac{1}{9} \right\rfloor - 1 = 6 = 000110$$

- MSB in both upper and lower is 000
- send 000 to the decoder
- shift left three times and put three 0s in LSB of L
- shift left three times and put three 1s in LSB of U

Summary

- Transmitted sequence: 011000
- L: 000000 = 0
- U: 110111 = 55

Example: this∆is∆the∆tithe

- ***All*** context tables will be updated accordingly

Example: this∆is∆the∆tithe

- zero order context

Letter	Count	Cum_count
t	2	2
h	1	3
i	2	5
s	2	7
∆	2	9
<Esc>	1	10
Total count		10

Example: this∆is∆the∆tithe

- 1st order context

Context	Letter	Count	Cum_count
t	h	1	1
	<Esc>	1	2
	Total Count		2
h	i	1	1
	<Esc>	1	2
	Total Count		2
i	s	2	2
	<Esc>	1	3
	Total Count		3
s	∆	2	2
	<Esc>	1	3
	Total Count		3
∆	i	1	1
	t	1	2
	<Esc>	1	3
	Total Count		3

Example: thisΔisΔtheΔtithe

- 2nd order context

Context	Letter	Count	Cum_count
th	i	1	1
	<Esc>	1	2
	Total Count		2
hi	s	1	1
	<Esc>	1	2
	Total Count		2
is	Δ	2	2
	<Esc>	1	3
	Total Count		3
sΔ	i	1	1
	t	*1*	*2*
	<Esc>	1	*3*
	Total Count		*3*
Δi	s	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- Next symbol is *h*
- Both encoder and decoder consider the 2nd order context that occurs before *'h'*, which is *'Δt'*
- The context *'Δt'* has not been encountered before
  - Both the encoder and the decoder
    - look in the 1st order context, ***without sending/receiving any <ESC>***
    - **add this *NEW* context to their tables (during the table update process)**

Example: this∆is∆the∆tithe

- 1st order context

Context	Letter	Count	Cum_count
t	h	1	1
	<Esc>	1	2
	Total Count		2
h	i	1	1
	<Esc>	1	2
	Total Count		2
i	s	2	2
	<Esc>	1	3
	Total Count		3
s	∆	2	2
	<Esc>	1	3
∆	i	1	1
	<Esc>	1	2
	Total Count		2

Example: this∆is∆the∆tithe

- 1st order context

Context	Letter	Count	Cum_count
t	h	1	1
	<Esc>	1	2
	Total Count		2
h	i	1	1
	<Esc>	1	2
	Total Count		2
i	s	2	2
	<Esc>	1	3
	Total Count		3
s	∆	2	2
	<Esc>	1	3
∆	i	1	1
	<Esc>	1	2
	Total Count		2

Example: thisΔisΔtheΔtithe

- **We adjust the boundaries of the interval to encode *'h'* in the context *'t'***

$$L = 0 + \left\lfloor (55 - 0 + 1) \times \frac{0}{2} \right\rfloor = 0 = 000000$$

$$U = 0 + \left\lfloor (55 - 0 + 1) \times \frac{1}{2} \right\rfloor - 1 = 27 = 011011$$

- MSB in both upper and lower is 0
- send 0 to the decoder
- shift left and put 0 in LSB of L
- shift left and put 1 in LSB of U

Summary

- Transmitted sequence: 0110000
- L: 000000 = 0
- U: 110111 = 55

Example: thisΔisΔtheΔtithe

- ***All*** context tables will be updated accordingly

Example: thisΔisΔtheΔtithe

- zero order context

Letter	Count	Cum_count
t	2	2
h	2	4
i	2	6
s	2	8
Δ	2	10
<Esc>	1	11
Total count		11

Example: thisΔisΔtheΔtithe

- 1st order context

Context	Letter	Count	Cum_count
t	h	2	2
	<Esc>	1	3
	Total Count		3
h	i	1	1
	<Esc>	1	2
	Total Count		2
i	s	2	2
	<Esc>	1	3
	Total Count		3
s	Δ	2	2
	<Esc>	1	3
	Total Count		3
Δ	i	1	1
	t	1	2
	<Esc>	1	3
	Total Count		3

Example: thisΔisΔtheΔtithe

- 2st order context

Context	Letter	Count	Cum_count
th	i	1	1
	<Esc>	1	2
	Total Count		2
hi	s	1	1
	<Esc>	1	2
	Total Count		2
is	Δ	2	2
	<Esc>	1	3
	Total Count		3
sΔ	i	1	1
	t	1	2
	<Esc>	1	3
	Total Count		3
Δi	s	1	1
	<Esc>	1	2
	Total Count		2
Δt	***h***	***1***	***1***
	<Esc>	***1***	***2***
	Total Count		***2***

Improvements

- The ***<Esc>*** symbol is always considered as having a count of 1
- This method is called ***ppma***
- ***ppmb*** is an improvement over ***ppma*** which reduces 1 from the count of each symbol and assigns these counts to the ***<Esc>*** symbol
- ***ppmc*** leaves the counts of symbols unchanged and the ***<Esc>*** symbol will be given a count equal to the number of symbols
- There are many other methods to estimate the count of the ***<Esc>*** symbol

Improvements

PPMa

Context	Symbol	count
prob	a	10
	l	9
	o	3
	<Esc>	1
Total Count		23

PPMb

Context	Symbol	count
prob	a	9
	l	8
	o	2
	<Esc>	3
Total Count		22

PPMc

Context	Symbol	count
prob	a	10
	l	9
	o	3
	<Esc>	3
Total Count		25

PPMc proved to be the most efficient

Deterministic Context

- Contexts that are always followed by the same symbol are called ***deterministic contexts***

Context	Symbol	count
prob	a	10
	<Esc>	1
Total Count		11

- ***PPM**** algorithm uses the fact that long contexts which give only a single prediction (i.e., deterministic context) are seldom followed by another symbol
  - If *prob* has always been followed by *a* in the past, it will probably not be followed by any other letter next time
- In ***PPM****,
  - the longest deterministic context is identified, if any, (*known to the decoder as well*)
  - ***If*** the symbol to be encoded occurs in that context, the symbol will be encoded, ***otherwise*** an escape symbol is encoded and the algorithm goes to the maximum context length
  - This strategy may save us some escape symbols from the context with maximum length to that deterministic context

The Exclusion Principle

- The basic idea behind arithmetic encoding is the division of the unit interval into subintervals, each of which represents a particular symbol
- The smaller the interval, the more bits are required to distinguish it from other subintervals
- If we can reduce the number of symbols to be represented, the number of subintervals goes down as well
- This in turn means that the sizes of the subintervals increase, leading to a reduction in the number of bits required for encoding
- The exclusion principle used in ***ppm*** provides that kind of reduction in rate

The Exclusion Principle (Example)

- Consider that we have been compressing a text sequence and came upon the sequence *probability*, where
  - We are trying to encode the letter *a*,
  - the tables of the two-letter context *ob* and the one-letter context *b* are as follow:

Context	Symbol	count
ob	l	10
	o	3
	<Esc>	2
Total Count		15

Context	Symbol	count
b	l	5
	o	3
	a	4
	r	2
	e	2
	<Esc>	5
Total Count		21

- Since *a* does not occur in the *ob* context, we issue an escape symbol and reduce the size of the context to one
- Looking at the one-letter context *b* table, we see that *a* does occur in this context with a count of 4 out of 21

The Exclusion Principle (Example)

- Note that
  - context *b* table also includes letters *l* and *o*
  - Yet by sending the escape symbol in the *ob* context, we have signaled the decoder that the symbol being encoded is not any of the letters that we have in table *ob*
  - Hence, we should *temporary* eliminate the *l* and *o* from the context *b* table
  - This *temporary* exclusion reduces the count of *a* in this context to 4 out of 11 (not 4 out of 21)

Context	Symbol	count
ob	l	10
	o	3
	<Esc>	2
Total Count		15

Context	Symbol	count
b	l	5
	o	3
	a	4
	r	2
	e	2
	<Esc>	5
Total Count		21

Context	Symbol	count
b	a	4
	r	2
	e	2
	<Esc>	3
Total Count		11