COMP2610/6261 - Information Theory

Lecture 16: Arithmetic Coding

Australian National University

25 September, 2018

1. From SFE to Arithmetic Coding

2. Arithmetic Coding: Encoder
   - Intervals for Sequences
   - Co
   - Pu

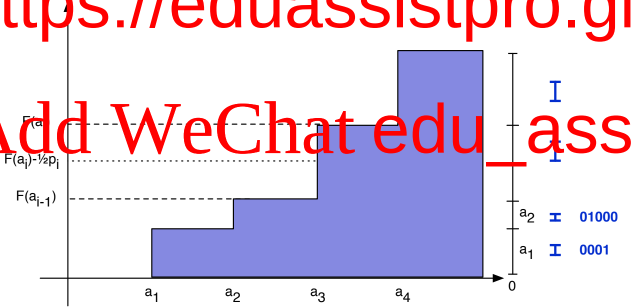
3. Arithmetic Coding: Decoder

4. Adapting Distributions On-The-Fly

Interval Codes (Recap)

Shannon-Fano-Elias Coding method:

- Order the alphabet $\mathcal{A}$.
- Represent distribution $\mathbf{p}$ by cumulative distribution F.
- Construct code by finding intervals of width $\frac{p_i}{2}$ that lie in each symbol interval $[F(a_{i-1}), F(a_i))$



Intervals and Prefix Codes (Recap)

The set of numbers in $[0, 1)$ that start with a given sequence of bits $\mathbf{b} = b_1 \dots b_n$ form the interval

$$\left[0.\mathtt{b} \quad \frac{1}{} \quad 0.\mathtt{0}\dots\mathtt{1}\right)$$

This inter $\mathtt{1}\ \mathtt{2}$ $\mathtt{n}$ refix

Prefix property (interval form): Once you pic $\mathtt{b_n}$, you cannot pick any codeword in the codeword inter

$$\left[0.\mathtt{b_1 b_2 \dots b_n}, 0.\mathtt{b_1 b_2 \dots b_n} + \frac{1}{2^n}\right)$$

1. From SFE to Arithmetic Coding

2. Arithmetic Coding: Encoder
   - Intervals for Sequences
   - Co
   - Pu

3. Arithmetic Coding: Decoder

4. Adapting Distributions On-The-Fly

Interval Coding Blocks

What if we apply SFE coding to blocks of an ensemble X?

Example: Let $\mathcal{A} = \{\texttt{aa}, \texttt{ab}, \texttt{ba}, \texttt{bb}\}$ with $\mathbf{p} = (0.2, 0.6, 0.1, 0.1)$.

Interval Coding Blocks

What if we apply SFE coding to blocks of an ensemble X?

Example: Let $\mathcal{A} = \{\mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}\}$ with $\mathbf{p} = (0.2, 0.6, 0.1, 0.1)$.

x	p	$\bar{F}$	$\bar{F}_2$	ℓ	Code
aa	0.2	0.1	$0.0\overline{0011}$	4	0001
ab					
ba					
bb					

Extend to longer sequences

Interval Coding Blocks

What if we apply SFE coding to blocks of an ensemble X?

Example: Let $\mathcal{A} = \{\texttt{aa}, \texttt{ab}, \texttt{ba}, \texttt{bb}\}$ with $\mathbf{p} = (0.2, 0.6, 0.1, 0.1)$.

$\mathbf{x}$	p	$\bar{F}$	$\bar{F}_2$	ℓ	Code
aa	0.2	0.1	$0.0\overline{0011}$	4	0001
ab					
ba					
bb					

Extend to longer sequences

This works but:

- Need $P(\mathbf{x})$ for all $\mathbf{x}$
- Total $|\mathcal{A}|^N$ values for length N
- Huffman has similar complexity but shorter codes.

Arithmetic Coding: A Bird's Eye View

Basic idea of arithmetic coding follows SFE coding

	SFE Coding	Arithmetic coding
Input	Single outcome x_i	Sequence of outcomes $x_1 x_2 \ldots x_N$
Key st		
Output	Binary string corresponding to chosen interval	ing to chosen interval

Arithmetic Coding: A Bird's Eye View

Basic idea of arithmetic coding follows SFE coding

	SFE Coding	Arithmetic coding
Input	Single outcome x_i	Sequence of outcomes $x_1 x_2 \dots x_N$
Key st[illegible]		
	Use $[F(x_{i-1}), F(x_i))$	
Output	Binary string corresponding to chosen interval	[illegible] ing to chosen interval
	Output first $\ell(x_i)$ bits of mid-point of interval	Output first $\ell(x_1 x_2 \dots x_N)$ bits of midpoint of interval

Arithmetic Coding: Summary

Arithmetic coding has some important properties:

- We do **not** compute a symbol coding for X and then concatenate
  -
- We d
  -
- We do not assume that each of the x_i
  - Not restricted to extended ensembles
  - Adapts to data distribution

1. From SFE to Arithmetic Coding

2. Arithmetic Coding: Encoder
   - Intervals for Sequences
   - Co
   - Pu

3. Arithmetic Coding: Decoder

4. Adapting Distributions On-The-Fly

Computing an Interval for Sequences

Say $N = 2$ and we want to code $x_1 x_2$

Ideally, we'd like to compute $p(\mathbf{x})$ for all possible $\mathbf{x}$ of length 2, and then find the interval for $p(x_1 x_2)$

Key ideas

- we d
  - decompose joint into conditional probabilities

Computing an Interval for Sequences

Say $N = 2$ and we want to code $x_1 x_2$

Ideally, we'd like to compute $p(\mathbf{x})$ for all possible $\mathbf{x}$ of length 2, and then find the interval for $p(x_1 x_2)$

Key ideas

- we d
  - decompose joint into conditional probabilities

- $p(\cdot|x_1)$ is just another probability distribution
  - so we can compute intervals as per SFE

Computing an Interval for Sequences

Say $N = 2$ and we want to code $x_1 x_2$

Ideally, we'd like to compute $p(\mathbf{x})$ for all possible $\mathbf{x}$ of length 2, and then find the interval for $p(x_1 x_2)$

Key ideas

- we c
  - decompose joint into conditional probabilities
- $p(\cdot|x_1)$ is just another probability distribution
  - so we can compute intervals as per SFE
- we can find an interval for $p(x_2|x_1)$ **within** the interval for x_1
  - normal SFE computes the interval within $[0, 1)$ by default

Computing an Interval for Sequences

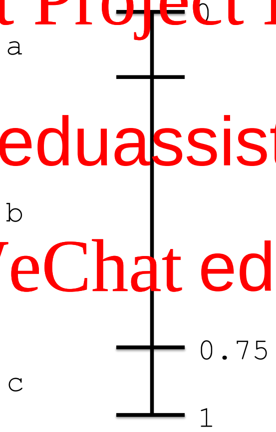
Example: Suppose $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $p(\mathtt{a}) = 0.25, p(\mathtt{b}) = 0.5, p(\mathtt{c}) = 0.25$

Like with SFE coding, we'd begin by slicing up $[0, 1)$ into three subintervals:

Computing an Interval for Sequences

Example: Suppose $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $p(\mathtt{a}) = 0.25, p(\mathtt{b}) = 0.5, p(\mathtt{c}) = 0.25$

Like with SFE coding, we'd begin by slicing up $[0, 1)$ into three subintervals:



So e.g. we treat $[0.25, 0.75)$ as the interval for $\mathtt{b}$

Computing an Interval for Sequences

Suppose the first symbol is $\texttt{b}$, and $p(\texttt{a}|\texttt{b}) = 0.25, p(\texttt{b}|\texttt{b}) = 0.5, p(\texttt{c}|\texttt{b}) = 0.25$

Computing an Interval for Sequences

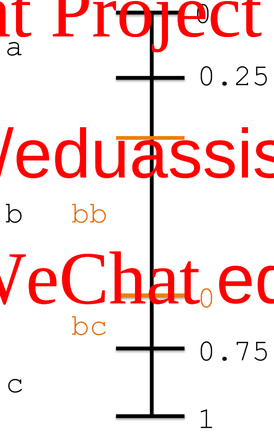
Suppose the first symbol is $\mathtt{b}$, and $p(\mathtt{a}|\mathtt{b}) = 0.25, p(\mathtt{b}|\mathtt{b}) = 0.5, p(\mathtt{c}|\mathtt{b}) = 0.25$

To code $\mathtt{ba}, \mathtt{bb}, \mathtt{bc}$, now slice up $[0.25, 0.75)$, the interval for $\mathtt{b}$ itself:

Computing an Interval for Sequences

Suppose the first symbol is $\texttt{b}$, and $p(\texttt{a}|\texttt{b}) = 0.25$, $p(\texttt{b}|\texttt{b}) = 0.5$, $p(\texttt{c}|\texttt{b}) = 0.25$

To code $\texttt{ba}$, $\texttt{bb}$, $\texttt{bc}$, now slice up $[0.25, 0.75)$, the interval for $\texttt{b}$ itself:



For $\texttt{ba}$ we choose the interval of length $p(\texttt{a}|\texttt{b}) = 0.25$ times the length of the enclosing interval $(0.75 - 0.25 = 0.5)$, i.e. $(0.25)(0.5) = 0.125$

Arithmetic Coding: End of Stream Symbol

It is convenient to explicitly have a special “end of stream” symbol, □

We add thi

- e.g. bability of see
- Implicitly we think of `ab` as actually being `ab□`

End of stream is by definition reached when we choo this special symbol

Arithmetic Coding: End of Stream Example

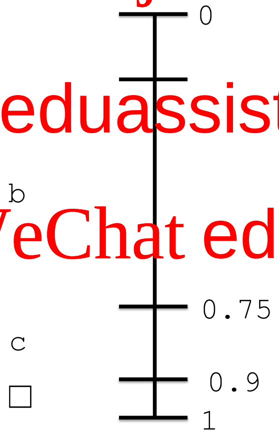
Example: Suppose $\mathcal{A} = \{\texttt{a}, \texttt{b}, \texttt{c}, \Box\}$ and
$p(\texttt{a}) = 0.25, p(\texttt{b}) = 0.5, p(\texttt{c}) = 0.15, p(\Box) = 0.1$

Like with SFE coding, we'd begin by slicing up $[0, 1)$ into three subintervals:

Arithmetic Coding: End of Stream Example

Example: Suppose $\mathcal{A} = \{\texttt{a}, \texttt{b}, \texttt{c}, \Box\}$ and
$p(\texttt{a}) = 0.25, p(\texttt{b}) = 0.5, p(\texttt{c}) = 0.15, p(\Box) = 0.1$

Like with SFE coding, we'd begin by slicing up $[0, 1)$ into three subintervals:



Arithmetic Coding: End of Stream Example

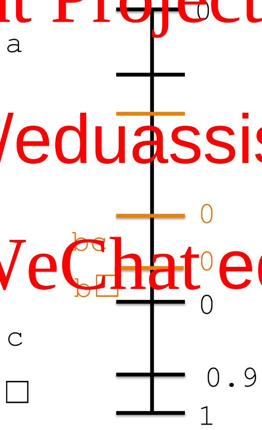
Now suppose that $p(\cdot|\texttt{b})$ stays the same as $p(\cdot)$

If the first symbol is b, we carve the interval for b into four pieces:

Arithmetic Coding: End of Stream Example

Now suppose that $p(\cdot|\texttt{b})$ stays the same as $p(\cdot)$

If the first symbol is `b`, we carve the interval for `b` into four pieces:



Exact same idea as before, just with special symbol □

Arithmetic Coding for Arbitrary Sequences

These ideas generalise to arbitrary length sequences

- We don't even need to know the sequence length beforehand!

As we see more symbols, we slice the appropriate sub-interval of $[0, 1)$ based on t

- Ter

1. From SFE to Arithmetic Coding

2. Arithmetic Coding: Encoder
   - Intervals for Sequences
   - Co
   - Pu

3. Arithmetic Coding: Decoder

4. Adapting Distributions On-The-Fly

Arithmetic Coding: Codeword Generation

Once we've seen the entire sequence, we end up with interval $[u, v)$

- How to output a codeword?

Arithmetic Coding: Codeword Generation

Once we've seen the entire sequence, we end up with interval $[u, v)$

- How to output a codeword?

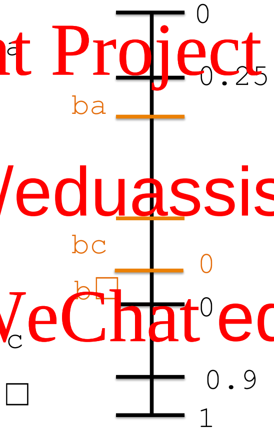
As per SF ... $+ v)/2$

- Her ...
- As b ...
  contained in the codeword interval

Generally, we can output some bits on the fly, rather t
process the entire sequence

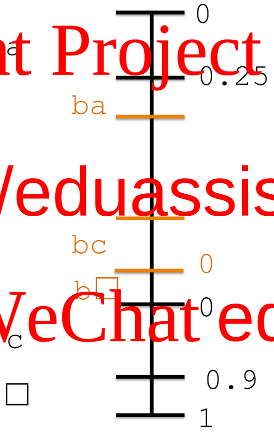
Arithmetic Coding: Codeword Generation Example

In previous example with input b, we'd stop in the interval for b□, i.e. $[0.7, 0.75)$



Arithmetic Coding: Codeword Generation Example

In previous example with input b, we'd stop in the interval for b□, i.e. $[0.7, 0.75)$



Midpoint is $0.725 = 10111\overline{0011}$, and $p(\texttt{b}\square) = (1/2) \cdot (0.1) = 0.05$

Output the first $\lceil \log_2 1/0.05 \rceil + 1 = 6$ bits, i.e. 101110

Arithmetic Coding: Formal Encoder

Formally, we compute the interval $[u, v)$ for a generic sequence as follows:

Arithmetic Coding of stream $x_1 x_2 \ldots$

$u \leftarrow 0.0$
$v \leftarrow 1.0$
$p \leftarrow v - u$

for $n = 1,$

- Com
- Compute $U_n(a_i|x_1, ..., x_{n-1}) = \quad_{i'=1} p(x_n = a_{i'} \; x_1, \ldots, x_n \;\; _1)$
- $v \leftarrow u + p \cdot U_n(x_n|x_1, ..., x_{n-1})$
- $u \leftarrow u + p \cdot L_n(x_n|x_1, ..., x_{n-1})$
- $p \leftarrow v - u$
- **if** $x_n = \square$, terminate

Output first $\ell(x_1 x_2 \ldots x_N) = \lceil \log 1/p \rceil + 1$ bits of $(u + v)/2$

Here, L_n, U_n just compute the appropriate lower and upper bounds, as per SFE coding

- We rescale these based on the current interval length

Decoding

How do we decode a sequence of bits?

Rough Sketch

- Carve out $[0, 1)$ based on initial distribution
- Ke

- Out
- Carve out appropriate interval based on prob

- $\vdots$

We can stop once we have containment in interval for □

Decoding: Example

Suppose $p(\mathtt{a}) = 0.5, p(\mathtt{b}) = 0.125, p(\mathtt{c}) = 0.25, p(\Box) = 0.125$ for every outcome in sequence

Decode `0110111`:

Seque	
0	t symbol a

Decoding: Example

Suppose $p(\texttt{a}) = 0.5, p(\texttt{b}) = 0.125, p(\texttt{c}) = 0.25, p(\Box) = 0.125$ for every outcome in sequence

Decode $\texttt{0110111}$:

Seque			
0			t symbol a
01	2	10	

Decoding: Example

Suppose $p(\mathtt{a}) = 0.5, p(\mathtt{b}) = 0.125, p(\mathtt{c}) = 0.25, p(\Box) = 0.125$ for every outcome in sequence

Decode `0110111`:

Seque			
0			t symbol a
01	$_2$	$_{10}$	
011	$[0.011, 0.100)_2$	[0.37	bol c

Decoding: Example

Suppose $p(\mathtt{a}) = 0.5, p(\mathtt{b}) = 0.125, p(\mathtt{c}) = 0.25, p(\Box) = 0.125$ for every outcome in sequence

Decode `0110111`:

Seque			
`0`			t symbol a
`01`	2	10	
`011`	$[0.011, 0.100)_2$	[0.37	bol c
`0110`	$[0.0110, 0.0111)_2$	[0.37	

Decoding: Example

Suppose $p(\texttt{a}) = 0.5, p(\texttt{b}) = 0.125, p(\texttt{c}) = 0.25, p(\Box) = 0.125$ for every outcome in sequence

Decode `0110111`:

Seque			
`0`			t symbol a
`01`	$_2$	$_{10}$	
`011`	$[0.011, 0.100)_2$	[0.37	bol c
`0110`	$[0.0110, 0.0111)_2$	[0.37	
`01101`	$[0.01101, 0.01110)_2$	[0.40	

Decoding: Example

Suppose $p(\mathtt{a}) = 0.5, p(\mathtt{b}) = 0.125, p(\mathtt{c}) = 0.25, p(\Box) = 0.125$ for every outcome in sequence

Decode `0110111`:

Seque [illegible]	[illegible]	[illegible]	[illegible]
`0`	[illegible]	[illegible]	t symbol a
`01`	[illegible] $_2$	[illegible] $_{10}$	[illegible]
`011`	$[0.011, 0.100)_2$	[0.37 [illegible]	bol c
`0110`	$[0.0110, 0.0111)_2$	[0.37 [illegible]	[illegible]
`01101`	$[0.01101, 0.01110)_2$	[0.40 [illegible]	[illegible]
`011011`	$[0.011011, 0.011100)_2$	[0.42 [illegible]	bol $\Box$

The last bit here is actually redundant (inherited from $+1$ bit in midpoint representation)

1. From SFE to Arithmetic Coding

2. Arithmetic Coding: Encoder
   - Intervals for Sequences
   - Co
   - Pu

3. Arithmetic Coding: Decoder

4. Adapting Distributions On-The-Fly

Adaptive Probabilities

So far we assume the sequence of probabilities are given in advance

In **Lecture 5**, you saw Bernoulli distribution for two outcomes

Adaptive Probabilities

So far we assume the sequence of probabilities are given in advance

In **Lecture 5**, you saw Bernoulli distribution for two outcomes

- Bet

- The
  Bet $(\theta| \quad_h + \quad_h, \quad_t + \quad_t)$

- The expected value of θ under the post

$$p(x = \mathtt{h} | n_h, n_t, m_h, m_t) = \frac{m_h + n_h}{m_h + n_h + m_t + n_t}$$

Dirichlet Model

A **Dirichlet distribution** is a generalisation of the Beta distribution to more than two outcomes. Its parameter is a vector $\mathbf{m} = (m_1, \ldots, m_K)$ can be viewed as “virtual counts” for each symbol $a_1, \ldots, a_K$:

Can impl .

Dirichlet Model

A **Dirichlet distribution** is a generalisation of the Beta distribution to more than two outcomes. Its parameter is a vector $\mathbf{m} = (m_1, \ldots, m_K)$ can be viewed as "virtual counts" for each symbol $a_1, \ldots, a_K$:

Can impl .

Flexible

- e.g., Choose $\mathbf{m}$ to be frequency of English l
- $\sum_k m_k$ Large = Stable; Small = Responsive

title

Example: Start with $m_h = m_t = 1$ and observe sequence $\texttt{hht}$.

$$p(\cdot|\epsilon) = (\tfrac{1}{2}, \tfrac{1}{2}), p(\cdot|\mathrm{h}) = (\tfrac{2}{3}, \tfrac{1}{3}), p(\cdot|\mathrm{hh}) = (\tfrac{3}{4}, \tfrac{1}{4}) \; p(\cdot|\mathrm{hht}) = (\tfrac{3}{5}, \tfrac{2}{5})$$

viz. Laplace's Rule, where ϵ means empty string

Why? Be

$$p(\mathrm{h}|\mathrm{h}) = \frac{1 + 1}{1 + 0 + 1}$$

$$p(\mathrm{t}|\mathrm{h}) = \frac{0 + 1}{1 + 0 + 1 + 1} = 1/3$$

We'll assume this learning is only for non □ symbols

- assume □ occurs with fixed probability each time

Adaptive Probabilities: Example

Possible outcomes $\mathtt{a}, \mathtt{b}, \Box$

Sequenc

Probabil $|\epsilon$

Encoder: ϵ

We start off with virtual counts $m_{\mathtt{a}} = m_{\mathtt{b}} = 1$

Adaptive Probabilities: Example

Possible outcomes $\texttt{a}, \texttt{b}, \square$

Observa

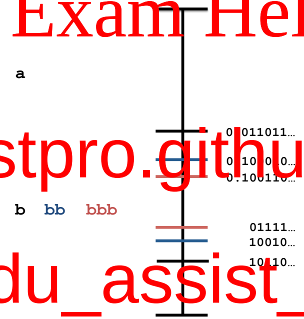
Probabil

Encoder Output: ϵ

Seeing $\texttt{b}$ makes us update $p(\texttt{a}|\texttt{b}) = (0.85) \cdot (1/3) \approx 0.28$, and $p(\texttt{b}|\texttt{b}) = (0.85) \cdot (2/3) \approx 0.57$. We keep $p(\square|\texttt{b}) = p(\square)$.

Adaptive Probabilities: Example

Possible outcomes $\mathtt{a}, \mathtt{b}, \Box$



Observa

Probabil

Encoder Output: $\mathtt{1}$

Seeing $\mathtt{bb}$ makes us update $p(\mathtt{a}|\mathtt{bb}) = (0.85) \cdot (1/4) \approx 0.21$, and
$p(\mathtt{b}|\mathtt{bb}) = (0.85) \cdot (3/4) \approx 0.64$
Now the first bit is unambiguously $\mathtt{1}$

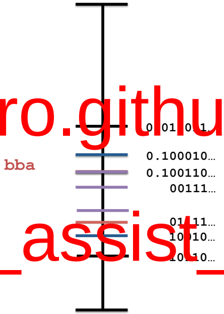
Adaptive Probabilities: Example

Possible outcomes $\mathtt{a}, \mathtt{b}, \Box$

Observa

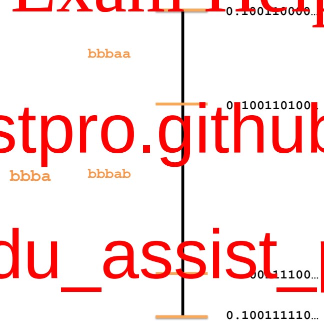
Probabil $|\mathtt{bbb} \approx (. \; , . \; , . \;)$

Encoder Output: 1



Adaptive Probabilities: Example

Possible outcomes a, b, □



Observa

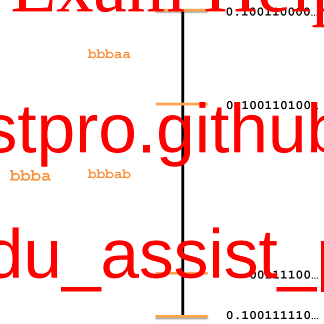
Probabil |bbba

Encoder Output: 1001

On seeing a, we can fill in three further bits unambiguously

Adaptive Probabilities: Example

Possible outcomes a, b, □



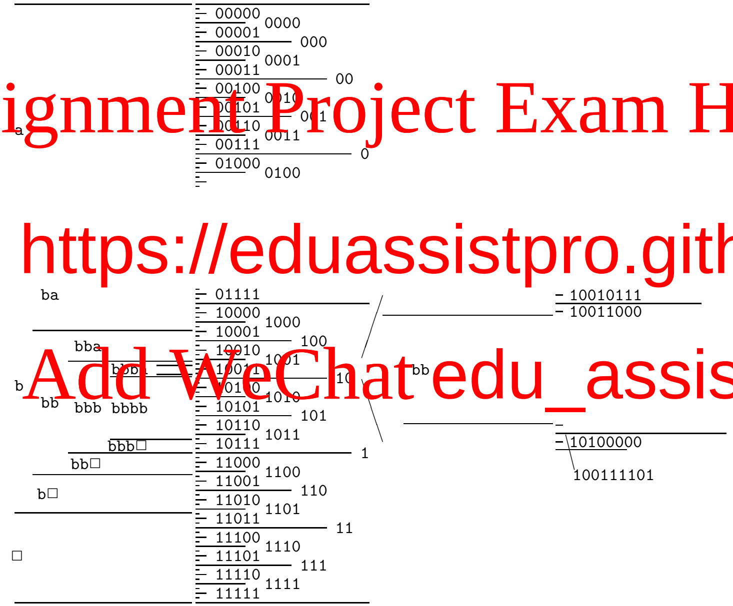
Observa

Probabil

Encoder Output: 100111101

To terminate, we find midpoint of 0.100111100 . . . and 0.100111110...

Arithmetic Coding: Example (MacKay, Figure 6.4)



Summary and Reading

Main Points

- Arithmetic Coding:
  - Uses interval coding and conditional probability
  -
  -
- Predictive distributions:
  - Update distribution after each symbol
  - Beta and Dirichlet priors = virtual counts

Reading

- Section 6.2 of MacKay