Lecture 8

Compression beyond iid data

Announcements

- HW1 feedback, solutions
- HW2 released
  - Ed clarification thread (pinned)
- Project Proposal
- IT-forum

Recap - entropy coders

- Huffman Coding
    - very fast
    - optimal symbol coder
    - achieves entropy when working with larger blocks - exponential complexity

Recap - entropy coders

- Arithmetic Coding
    - achieves entropy efficiently by treating entire input as a block
    - division operations are expensive
    - easily extends to complex and adaptive probability models

Recap - entropy coders

- ANS (Asymmetric Numeral Systems)
    - achieves entropy efficiently by treating entire input as a block
    - small compression overhead over arithmetic coding
    - two variants: rANS and tANS
    - faster than arithmetic coding
    - can modify base algorithm to use modern instructions like SIMD

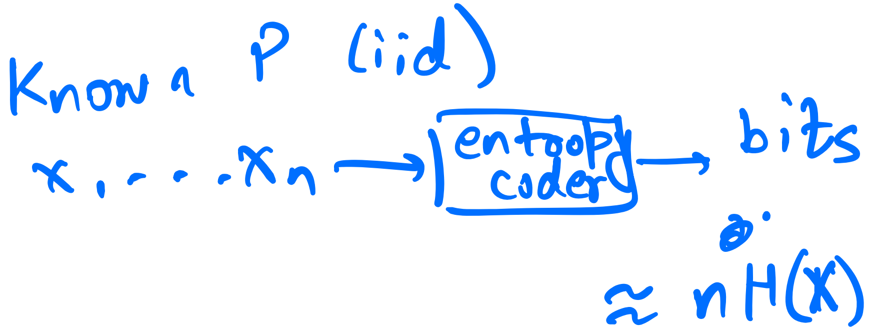
Recap - entropy coders

- Further reading on entropy coders, particularly ANS
  - Kedar's lecture notes
  - Paper by inventor (Jarek Duda): "Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding"
  - Yann Collet's blog - creator of FSE (tANS), LZ4 and zstd compression libraries
  - Charles Bloom's blog
  - Fabian Giesen's blog and paper on rANS with SIMD
  - https://encode.su/forums/2-Data-Compression - compression forum

Recap - entropy coders

Codec	Encode speed	Decode speed	compression
Huffman coding (1950s)	252 Mb/s	300 Mb/s	1.66
Arithmetic coding (1970s)	120 Mb/s	69 Mb/s	1.24
rANS (2010s)	76 Mb/s	140 Mb/s	1.24
tANS (2010s)	163 Mb/s	284 Mb/s	1.25

Source: Charles Bloom's blog



Recap - entropy coders

- All of these entropy coders are used in practice due to their unique strengths (and sometimes legacy reasons)
- We will keep revisiting these as components of various compression methods

Quiz - rANS recap

$x/p(x)$

rans_encode_step: `x_next = (x//freq[s])*M + cumul[s] +  x%freq[s]`

rans_decode_step:

```
    # Step I: find block_id, slot
    block_id = x//M
    slot = x%M

    # Step II: Find symbol s
    s = find_bin(cumul_array, slot)

    # Step III: retrieve x_prev
    x_prev = block_id*freq[s] + slot - cumul[s]
```

$x * p(x)$

Quiz

$M=8$

s	freq[s]	cumul[s]
0	3	0
1	3	3
2	2	6

$$x\_next = \lfloor x/\text{freq}[s] \rfloor \times M + \text{cumul}[s] + x \% \text{freq}[s]$$

Q1 rANS encoding

Say $\mathcal{X} = \{0, 1, 2\}$ be our symbols with probabilities $\{3/8, 3/8, 2/8\}$ respectively. You want to encode stream of symbols `2,0,1,0` using rANS.

1. What's the state value (`x`) in rANS after symbol `2` ? $\lfloor 0/2 \rfloor \times 8 + 6 + 0\%2 = 6$
2. What's the state value (`x`) in rANS after encoding symbols `2,0` ? $\lfloor 6/3 \rfloor \times 8 + 0 + \underbrace{6\%3}_{0} = 16$
3. What's the final state value (`x`) at the end of encoding stream `2,0,1,0` ? 114

Quiz - Q2 rANS decoding

Say $\mathcal{X} = \{0, 1, 2\}$ be our symbols with probabilities $\{3/8, 3/8, 2/8\}$ respectively. Now your decoder knows that the number of symbols are `4` and the final state your decoder received is `117`.

1. What is the value of `block_id` after running the `decode_block` for first time?
2. What is the value of `slot` after running the `decode_block` for first time?
3. What is the first decoded symbol? Note that this corresponds to the last encoded symbol since rANS decoding proceeds in reverse.
4. What is the updated state value (`x`) after first step?

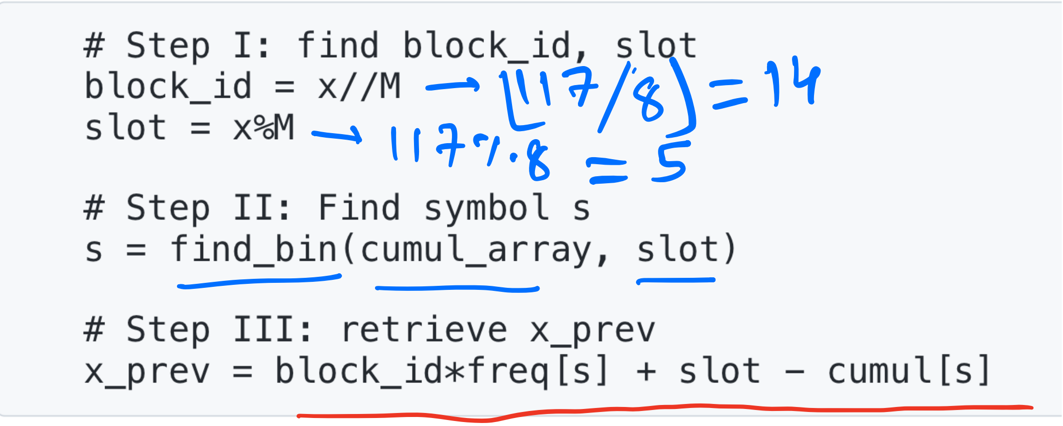
rANS decoding

$M=8$

s	freq[s]	cumul[s]
0	3	0
1	3	3
2	2	6

$X\_final = 117$

rans_decode_step:



```
# Step I: find block_id, slot
block_id = x//M
slot = x%M

# Step II: Find symbol s
s = find_bin(cumul_array, slot)

# Step III: retrieve x_prev
x_prev = block_id*freq[s] + slot - cumul[s]
```

$\rightarrow \lfloor 117/8 \rfloor = 14$

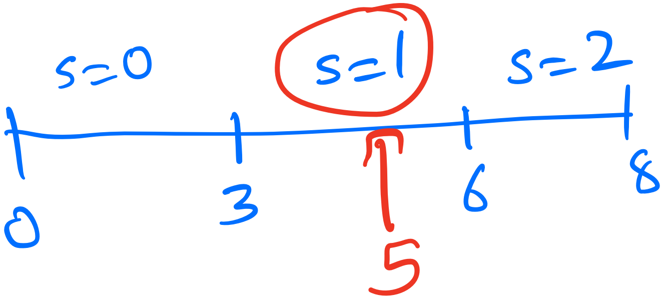
$\rightarrow 117\%8 = 5$

$block\_id = 14$

$slot = 5$

$s = 1$

$$x\_prev = 14 \times 3 + 5 - 3$$
$$= 42 - 3 + 5$$
$$= 44$$



Quiz - Q3 When to use ANS?

In which of the following scenarios would you consider using ANS as opposed to Huffman or Arithmetic coding? Select all that apply.

□ Your application requires the best compression ratio and you are willing to sacrifice encoding/decoding speeds. Arithmetic

□ Your application requires extremely fast encoding/decoding and you are willing to sacrifice compression ratio. Huffman

□ Your application requires adaptive decoding (i.e., the encoder and decoder need to build a model as they go through the data). Arithmetic

☑ You care about achieving close-to-optimal compression but also want good speed.

☑ You are working with a modern processor and want to exploit parallel processing to get higher speeds, while still achieving close-to-optimal compression. (SIMD)

```
$ cat sherlock.txt
    ...
      In mere size and strength it was a terrible creature which was
      lying stretched before us. It was not a pure bloodhound and it
      was not a pure mastiff; but it appeared to be a combination of
      the two—gaunt, savage, and as large as a small lioness. Even now
      in the stillness of death, the huge jaws seemed to be dripping
      with a bluish flame and the small, deep-set, cruel eyes were
      ringed with fire. I placed my hand upon the glowing muzzle, and
      as I held them up my own fingers smouldered and gleamed in the
      darkness.

      “Phosphorus,” I said.

      “A cunning preparation of it,” said Holmes, sniffing at the dead
    ...
```

Let's try and compress this 387 KB book.

```
>>> from core.data_block import DataBlock
>>>
>>> with open("sherlock.txt") as f:
>>>     data = f.read()
>>>
>>> print(DataBlock(data).get_entropy()*len(data)/8, "bytes")

199833 bytes
```

```
$ gzip < sherlock.txt | wc -c
134718

$ bzip2 < sherlock.txt | wc -c
99679
```

What's up? What are we missing here? Any suggestions?

1. Data is not iid.
2. Maybe the entire file doesn't have the same distribution (think concatenating an English novel with a Hindi novel).

In the next few lectures, we will discuss methods to compress real-life data, attempting to handle non-iid data whose distribution we do not know a priori.

Beyond iid data

- text
- images
- video
- tables
- basically anything in real life

Outline

- Lecture 8 (today): non-iid probability distributions, entropy rate, Markov sources
- Lecture 9: context-based arithmetic coding, introduction to LZ77 universal compression
- Lecture 10: lossless compression in practice

Probability recap

Recall for $U^n = (U_1, \ldots, U_n)$:

for iid

$P(U^n) = \Pi_{i=1}^n P(U_i)$

in general

$P(U^n) = \Pi_{i=1}^n P(U_i|U^{i-1}) = \Pi_{i=1}^n P(U_i|U_1, \ldots, U_{i-1})$

$P(U_1, U_2) = P(U_1) P(U_2|U_1)$

$\vdots$

Stochastic process (aka random process)

Given alphabet $\mathcal{U}$, a *stochastic process* $(U_1, U_2, \ldots)$ can have arbitrary dependence across the elements and is characterized by:

$P((U_1, U_2, \ldots, U_n) = (u_1, u_2, \ldots, u_n))$ for $n = 1, 2, \ldots$ and $(u_1, u_2, \ldots, u_n) \in \mathcal{U}^n$.

Need to define joint distribution for every n - way too general to be of much use.

$P(U_1, U_2)$

$P(U_{100}, U_{101})$

same distribution

Stationary stochastic process

> ### Definition: Stationary Process
>
> A stationary process is a stochastic process that is time-invariant, i.e., the probability distribution doesn't change with time (here time refers to the index in the sequence). More precisely, we have
>
> $$P(U_1 = u_1, U_2 = u_2, \ldots, U_n = u_n) = P(U_{l+1} = u_1, U_{l+2} = u_2, \ldots, U_{l+n} = u_n)$$
>
> for every n, every shift l and all $(u_1, u_2, \ldots, u_n) \in \mathcal{U}^n$.

- Mean, variance, entropy, etc. do not change with n.
- Can still have arbitrary time dependence.

$\mathbb{E}U_1 = \mathbb{E}U_{100}$

$H(U_1) = H(U_{100})$

$H(U_1, U_3) = H(U_{1000}, U_{1002})$

U_1 & $U_{1000000}$ can be dependent.

iid ⇒ stationary

$P(H) = P(T) = 1/2$

Examples

IID sequences: e.g., sequence of fair iid coin tosses

$$\rightarrow P((U_1, \dots, U_4) = (H, T, T, H)) = \frac{1}{2^4} = P(U_1 = H) P(U_2 = T) \cdots$$

$$P((U_{l+1}, U_{l+2}, \dots, U_{l+4}) = (H, T, T, H)) = \frac{1}{2^4}$$

U_n is independent of $U_{n-2}, \ldots U_1$ given U_{n-1}

Examples: Stationary time-invariant Markov processes

$0 \to 0, 1$

$1 \to 1, 2$

$2 \to 2, 0$

$$U_1 \sim Unif(\{0, 1, 2\})$$

$$U_{i+1} = (U_i + Z_i) \bmod 3$$

$$Z_i \sim Ber\left(\frac{1}{2}\right)$$

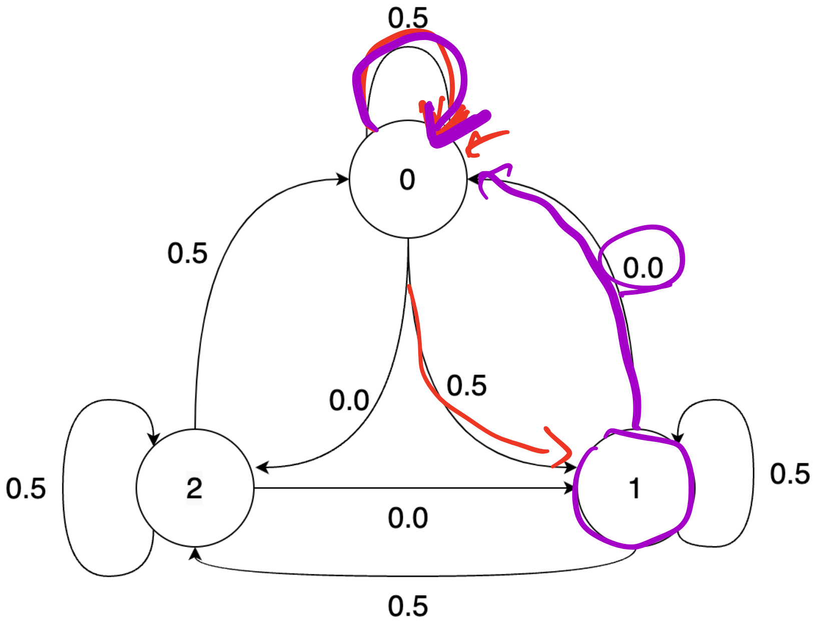
iid

$0/1$

```
Transition matrix
   U_{i+1} 0   1   2
U_i
0          0.5 0.5 0.0
1          0.0 0.5 0.5
2          0.5 0.0 0.5
```

Examples: Stationary time-invariant Markov processes

$$U_1 \sim Unif(\{0,1,2\})$$



marginalization

$$P(U_2=0) = \sum_{u_1=0}^{2} P(U_1=u_1, U_2=0)$$

chain rule

$$= \sum_{u_1=0}^{2} P(U_1=u_1)\, P(U_2=0|U_1=u_1)$$

$$= \underset{1/3}{P(U_1=0)} \times \underset{1/2}{P(U_2=0|U_1=0)}$$

$$+ \underset{1/3}{P(U_1=1)} \times \underset{0}{P(U_2=0|U_1=1)}$$

$$+ \underset{1/3}{P(U_1=2)} \times \underset{1/2}{P(U_2=0|U_1=2)}$$

$$= 1/3$$

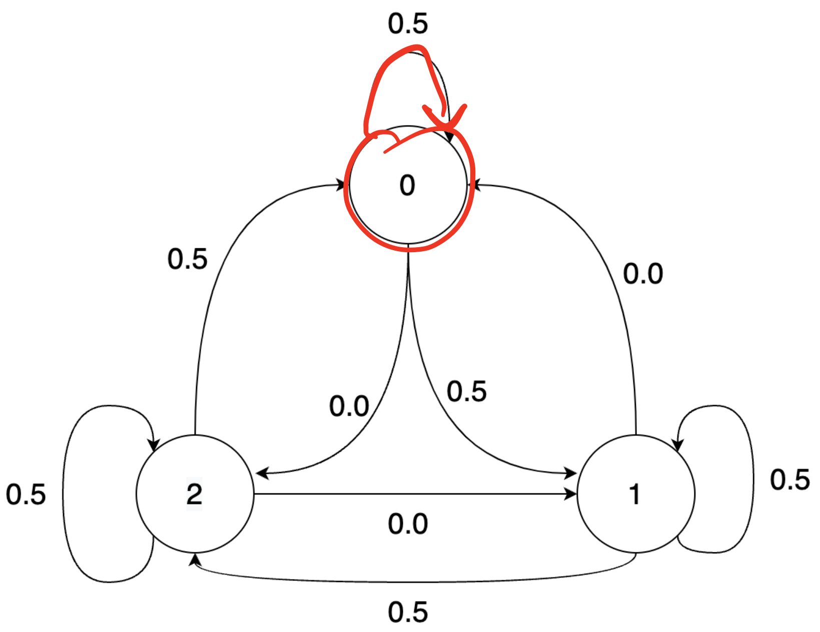
Markov chain

→ U_2 & U_1 have same distribution

$$\Downarrow$$

stationary

Examples: Stationary time-invariant Markov processes



$P(U_1 = 0) = 1$

$P(U_2 = 0) = 1/2$

Not stationary

What if $U_1 = 0$? Is the process still stationary?

$$U_1 \to U_1, \overset{U_2}{U_1 + Z_1}, \overset{U_3}{U_2 + Z_2}, \overset{U_4}{U_3 + Z_3}, \ldots \ldots$$

$$Z_i \sim \text{iid Ber}(1/2)$$

Examples: Stationary time-invariant Markov processes

$$U_1 \sim Unif(\{0, 1, 2\})$$

$$U_{i+1} = (U_i + Z_i) \bmod 3$$

$$Z_i \sim Ber\left(\frac{1}{2}\right) \quad \text{iid}$$

Question: Can you convert this to an iid sequence?

> All the iid compression work still useful!

$$\underset{\underline{Z_1}}{\underset{\downarrow}{U_2 - U_1}}, \underset{\underline{Z_2}}{\underset{\downarrow}{U_3 - U_2}}, \ldots$$

kth order Markov source

$U_{n-1}, \ldots U_{n-k}$
enough to predict U_n

> **Definition: kth order Markov source**
>
> A kth order Markov source is defined by the condition
>
> $$P(U_n | U_{n-1} U_{n-2} \ldots) = P(U_n | U_{n-1} U_{n-2} \ldots U_{n-k})$$
>
> for every n. In words, the conditional probability of U_n given the entire past depends only on the past k symbols. Or more precisely, U_n is independent of the past older than k symbols given the last k symbols.

Most practical stationary sources can be approximated well with a finite memory kth order Markov source with higher values of k typically providing a better approximation (with diminishing returns).

Markov - 1st order
iid → 0th order

Non-example

Arrival times for buses at a bus stop: $U_1, U_2, U_3, U_4, \ldots$

4:16 pm, 4:28 pm, 4:46 pm, 5:02 pm

Question 1: Is this stationary? → No, mean increases

Question 2: Can you convert this to a stationary (in fact iid) process?

delta/diff

4:28 − 4:16 → 12 minutes

"interarrival time"

Information-theoretic quantities for non-iid random variables

Conditional entropy

$$H(U) = \sum_{u} P(u)\log\frac{1}{P(u)}$$

$$= \mathbb{E}\left[\log\frac{1}{P(U)}\right]$$

The conditional entropy of U given V is defined as

$$H(U|V) \triangleq \underline{E\left[\log\frac{1}{P(U|V)}\right]}$$

Can also write this as

$$\begin{aligned}
H(U|V) &= \sum_{u\in\mathcal{U},v\in\mathcal{V}} \underline{P(u,v)}\log\frac{1}{P(u|v)} \\
&= \sum_{v\in\mathcal{V}} \underline{P(v)} \sum_{u\in\mathcal{U}} \underline{P(u|v)}\log\frac{1}{P(u|v)} \\
&= \sum_{v\in\mathcal{V}} P(v) H(U|V=v)
\end{aligned}$$

$H(U|V=v)$ → "uncertainty in U given V=v"

"conditional entropy of U given V=v"

$H(P(U|V=v))$

eg. $V=v \Rightarrow U \sim Ber(1/2) \Rightarrow H(U|V=v) = 1$

$H(U|V)$ → average uncertainty in U given V

"conditional entropy of U given V"

average of $H(U|V=v)$ $\quad \sum_v P(v)\, H(U|V=v)$

Conditional entropy - example

$P(U=0, V=0)$

U\V	0	1
0	1/4	1/4
1	0	1/2

$P(U=1, V=1)$

$U \sim \text{Ber}(1/2)$

$U=0 \Rightarrow V \sim \text{Ber}(1/2)$

$U=1 \Rightarrow V=1$

$$P(V=0 \mid U=0) = \frac{P(U=0, V=0)}{P(U=0)} = 1/4 / 1/2 = 1/2$$

$$H(V \mid U=0) = 1$$

$$H(V \mid U=1) = 0$$

$$H(V \mid U) = P(U=0)\,H(V \mid U=0) + P(U=1)\,H(V \mid U=1) = \tfrac{1}{2} \times 1 + \tfrac{1}{2} \times 0 = \boxed{\tfrac{1}{2}}$$

$$H(U) = \tfrac{1}{2}\log_2 \frac{1}{1/2} + \tfrac{1}{2}\log_2 \frac{1}{1/2} = 1$$

$$H(V) = \tfrac{1}{4}\log_2 \frac{1}{1/4} + \tfrac{3}{4}\log_2 \frac{1}{3/4} = 0.811 \geq H(V \mid U)$$

$$H(U,V) = \tfrac{1}{4} \times \log_2 \frac{1}{1/4} + \tfrac{1}{4} \times \log_2 \frac{1}{1/4} + \tfrac{1}{2} \times \log_2 \frac{1}{1/2}$$

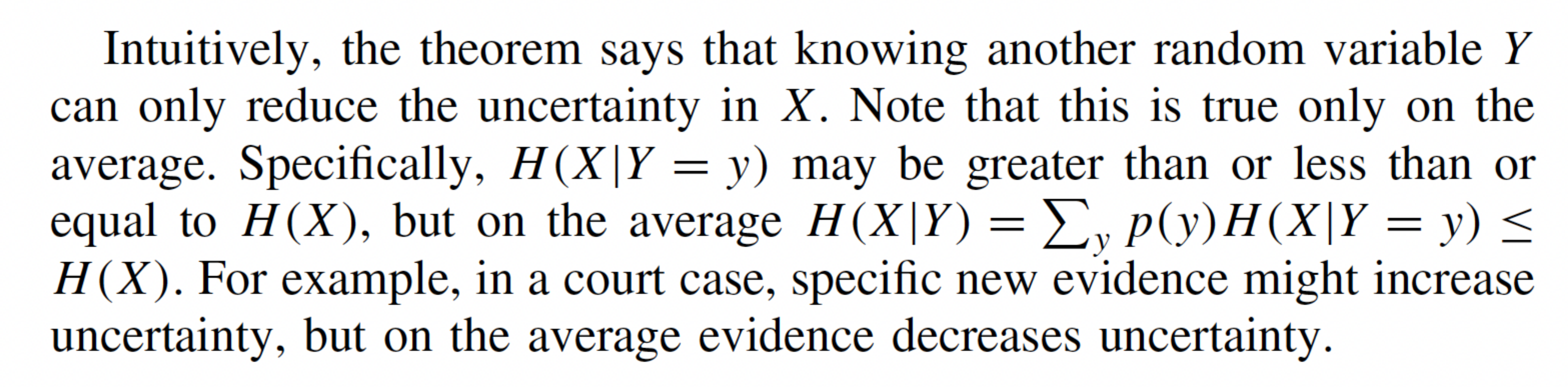
$$= 3/2 = H(U) + H(V \mid U)$$

Properties of conditional entropy

1. Conditioning reduces entropy: $H(U|V) \leq H(U)$ with equality iff U and V are independent.

Properties of conditional entropy

1. Conditioning reduces entropy: $H(U|V) \leq H(U)$ with equality iff U and V are independent.



Intuitively, the theorem says that knowing another random variable Y can only reduce the uncertainty in X. Note that this is true only on the average. Specifically, $H(X|Y = y)$ may be greater than or less than or equal to $H(X)$, but on the average $H(X|Y) = \sum_y p(y)H(X|Y = y) \leq H(X)$. For example, in a court case, specific new evidence might increase uncertainty, but on the average evidence decreases uncertainty.

(source: Cover & Thomas chapter 2)

Properties of conditional entropy

1. Conditioning reduces entropy: $H(U|V) \leq H(U)$ with equality iff U and V are independent.
2. Chain rule of entropy:

$$H(U,V) = H(U) + H(V|U) = H(V) + H(U|V)$$

For ~~ind~~ indep.: $H(U,V) = H(U)+H(V)$

Properties of conditional entropy

1. Conditioning reduces entropy: $H(U|V) \leq H(U)$ with equality iff U and V are independent.
2. Chain rule of entropy:

$$H(U,V) = H(U) + H(V|U) = H(V) + H(U|V)$$

3. Joint entropy vs. sum of entropies:

$$H(U,V) \leq H(U) + H(V)$$

with equality holding iff U and V are independent.

eg. $U \sim Ber(1/2)$
$V = 1-U$
$H(U) = 1 = H(V)$
$H(U,V) = 1 < H(U)+H(V)$

Properties of conditional entropy

1. Conditioning reduces entropy: $H(U|V) \leq H(U)$ with equality iff U and V are independent.
2. Chain rule of entropy:

$$H(U,V) = H(U) + H(V|U) = H(V) + H(U|V)$$

3. Joint entropy vs. sum of entropies:

$$H(U,V) \leq H(U) + H(V)$$

with equality holding iff U and V are independent.

Can generalize to conditioning U_{n+1} on $(U_1, U_2, \ldots, U_n)$:

$$H(U_{n+1}|U_1, U_2, \ldots, U_n)$$

Entropy rate

Before we look at examples, let's think about how we can generalize entropy for stationary processes. Some desired criteria:

- works for arbitrarily long dependency so $H(U_{n+1}|U_1, U_2, \ldots, U_n)$ for any finite n won't do
- has *operational* meaning in compression just like entropy
- is well-defined for any stationary process

Entropy rate

Not only one, but two equivalent ways of defining it!

Entropy rate

additional entropy per symbol given past

$$H_1(\mathbf{U}) = \lim_{n\to\infty} H(U_{n+1}|U_1, U_2, \ldots, U_n)$$

$$H_2(\mathbf{U}) = \lim_{n\to\infty} \frac{H(U_1, U_2, \ldots, U_n)}{n}$$

or avg. bits/symbol for large blocks

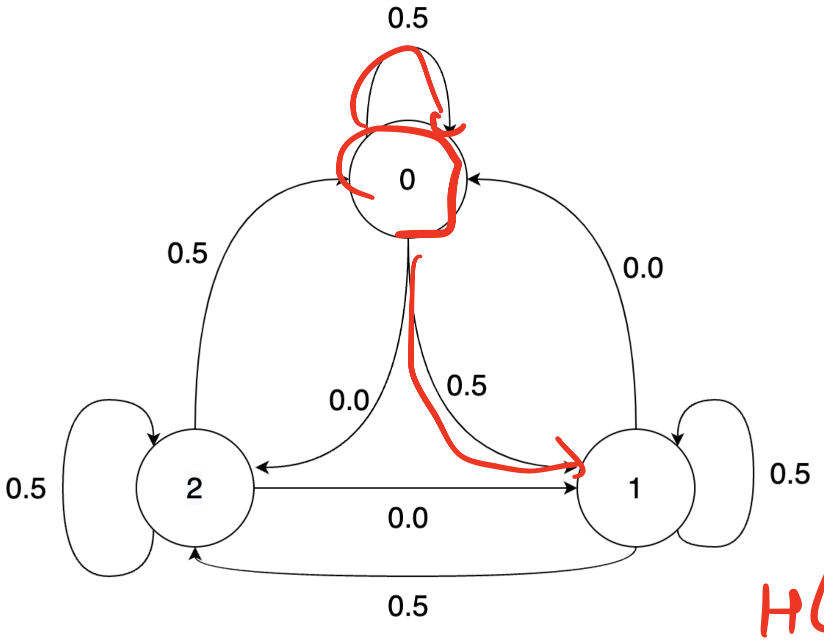
> ### C&T Thm 4.2.1
>
> For a stationary stochastic process, the two limits above are equal. We represent the limit as $H(\mathbf{U})$ (entropy rate of the process, also denoted as $H(\mathcal{U})$).

Examples

- Fair coin toss → $\frac{H(U_1, \dots U_n)}{n} = \frac{nH(U)}{n} = H(U) = 1$ bit

  iid ⇒ entropy rate = entropy

- Markov example



$H(U_2|U_1)$

given U_1 (say $U_1=0$)

U_2 takes 2 values with equal prob.

$H(U_2|U_1) = H(\text{Ber}(1/2)) = 1$ bit

$H(U_n|U_{n-1} \dots U_1) = \underline{\underline{H(U_n|U_{n-1})}} = 1$ bit iid $\Rightarrow$ $H(\mathcal{U}) = H(U_1)$

k^{th} order Markov: $H(\mathcal{U}) = H(U_{k+1} | U_k, \ldots, U_1)$

Example: entropy rate of English text

- Models (estimate probabilities from text):

  (a) 0th-order Markov chain (iid):

  $$H(\mathcal{X}) \approx 4.76 \qquad \text{bits per letter}$$

  (b) 1st order Markov chain:

  $$H(\mathcal{X}) \approx 4.03 \qquad \text{bits per letter}$$

  (c) 4th order Markov chain:

  $$H(\mathcal{X}) \approx 2.8 \qquad \text{bits per letter}$$

- Estimate by asking people to guess the next letter until they get it correct. The *order* of their guesses reflects their estimate of the *order* of their conditional probabilities for the next letter. (Shannon 1952).

  $$H(\mathcal{X}) \approx 1.3 \qquad \text{bits per letter}$$

$P(U)$ →

$P(U_1, U_2)$

$P(U_1, U_2, U_3)$

$\vdots$

$$P(U_1 = A, U_2 = C) = \frac{\#\{x_n = A, x_{n+1} = C\}}{\text{Total}}$$

Source: http://reeves.ee.duke.edu/information_theory/lecture4-Entropy_Rates.pdf

k^{th} order Markov source → $H(U_k | U_{k-1}, \dots U_1)$

AEP again!

> **Shannon–McMillan–Breiman theorem**
>
> $$-\frac{1}{n}\log_2 P(U_1, U_2, \dots, U_n) \to H(\mathbf{U}) \text{ a.s.}$$
>
> under technical conditions (ergodicity).

Takeaway: entropy rate is the best compression you can hope to achieve.

You can rewrite the above as $P(U_1, U_2, \dots, U_n) \approx 2^{-nH(\mathbf{U})}$ which should be familiar from Lecture 5.

How to achieve the entropy rate?

- Today: we start small, try to achieve 1st order entropy $H(U_{k+1}|U_k)$.
- Next lecture: achieving entropy rate for higher order processes and then arbitrary stationary distributions (in theory) and a really performant scheme (in practice). 🤯

Working with known 1st order Markov source

Suppose we know $P(U_2|U_1)$.

How would you go about compressing a block of length n using

$$E\left[\log_2 \frac{1}{P(U_1,\ldots,U_n)}\right] \approx nH(U_2|U_1)$$

bits?

Working with known 1st order Markov source

Idea 1: Use Huffman on blocks of length n.

- Usual concerns: big block size, complexity, etc.
- For non-iid sources, working on independent symbols is just plain suboptimal even discounting the effects of non-dyadic distributions.

Exercise: Compute $H(U_1)$ and $H(U_1, U_2)$ for

$$U_1 \sim Unif(\{0,1,2\})$$

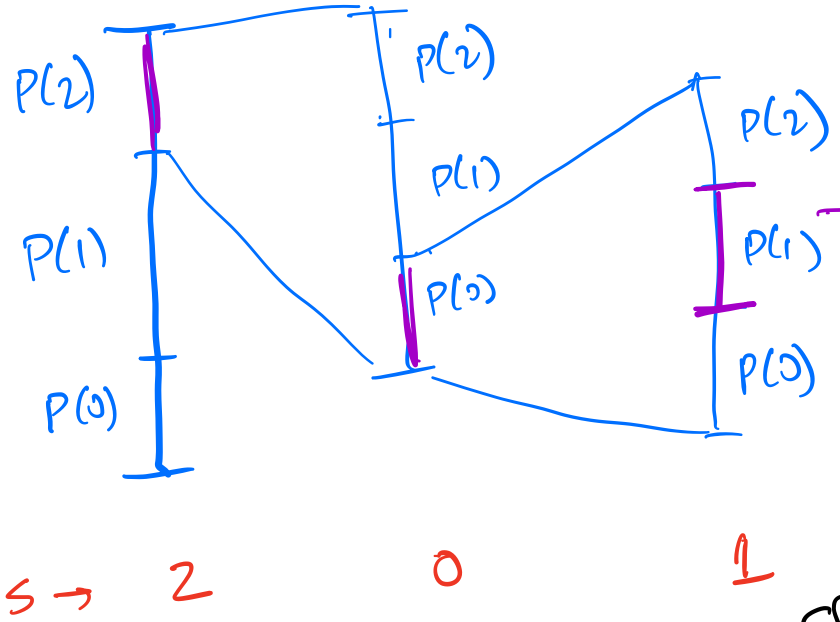
$$U_{i+1} = (U_i + Z_i) \bmod 3$$

$$Z_i \sim Ber\left(\frac{1}{2}\right)$$

and compare to $H(\mathbf{U})$.

Quick recap: Arithmetic coding

$X_1, X_2 \ldots \sim \text{iid } X$



interval length $= P(2) \times P(0) \times P(1)$

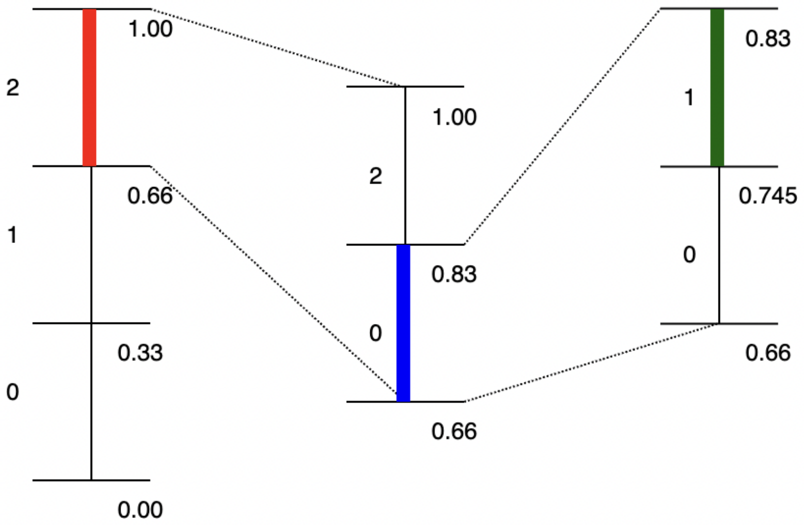
$$\text{Code length} \approx \log_2 \frac{1}{\text{interval length}}$$

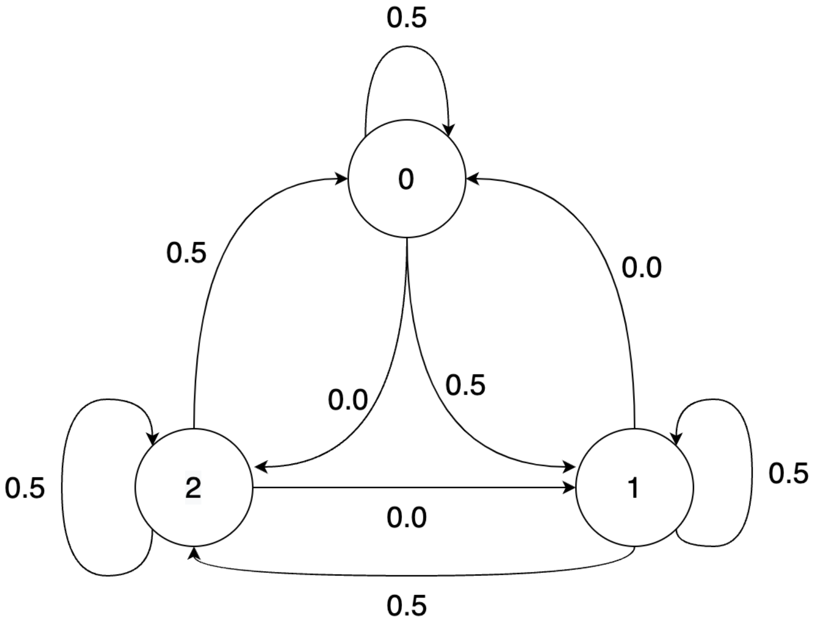
$$\approx \log_2 \frac{1}{P(x_1)P(x_2)\cdots P(x_n)}$$

$$E[\text{Code length}] \approx nH(X)$$

Working with known 1st order Markov source

Encoding 2, 0, 1

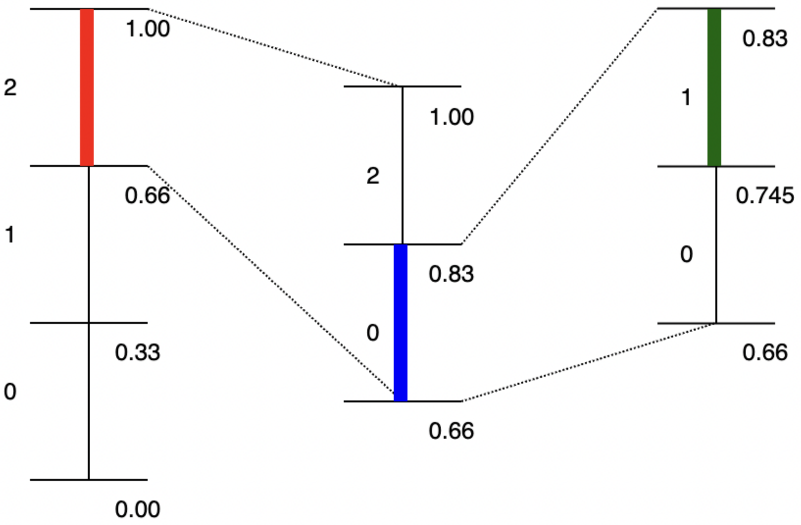


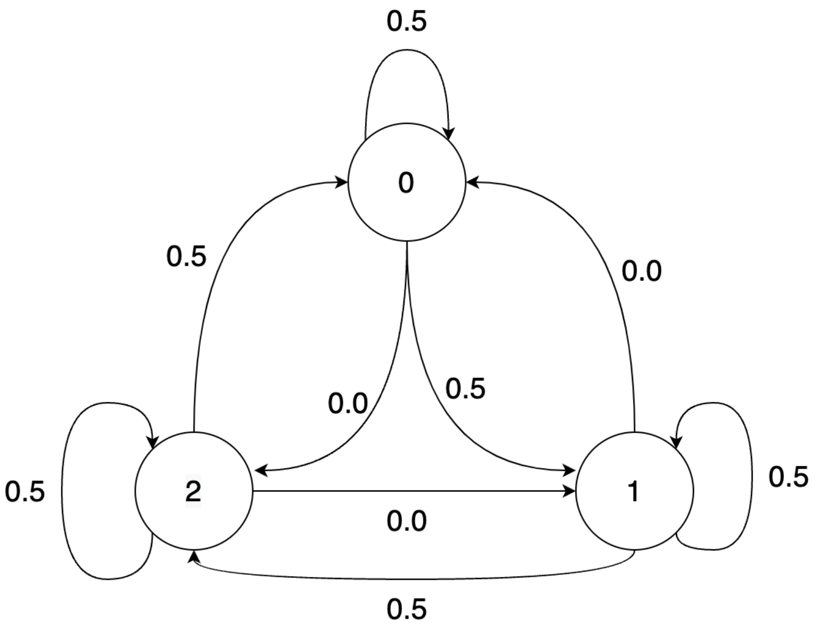


Question: Can you explain the general idea?

Working with known 1st order Markov source

Encoding 2, 0, 1





Question: Can you explain the general idea?

Answer: At every step, split interval by $P(-|u_{i-1})$ [more generally by $P(-|\text{entire past})$].

Arithmetic coding for known 1st order Markov source

Length of interval after encoding $u_1, u_2, u_3, \ldots, u_n = P(u_1)P(u_2|u_1)\ldots P(u_n|u_{n-1})$

Bits for encoding ~ $\log_2 \frac{1}{P(u_1)P(u_2|u_1)\ldots P(u_n|u_{n-1})}$

Expected bits per symbol

$$
\begin{aligned}
&\sim \frac{1}{n}E\left[\log_2 \frac{1}{P(U_1)P(U_2|U_1)\ldots P(U_n|U_{n-1})}\right] \\
&= \frac{1}{n}E\left[\log_2 \frac{1}{P(U_1)}\right] + \frac{1}{n}\sum_{i=2}^{n} E\left[\log_2 \frac{1}{P(U_i|U_{i-1})}\right] \\
&= \frac{1}{n}H(U_1) + \frac{n-1}{n}H(U_2|U_1) \\
&\sim H(U_2|U_1)
\end{aligned}
$$

Next time: more on compressing non-iid data with arithmetic coding and then we move on to LZ77!

Thank you!