# Information Theory and Networks

## Lecture 14: Practical Compression

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
http://www.maths.adelaide.edu.au/matthew.roughan/Lecture_notes/InformationTheory/

School of Mathematical Sciences,
University of Adelaide

September 18, 2013

# Part I

# Practical Compression



> Baseball is 90 percent mental and the other half is physical.
> *Yogi Berra*



## Section 1

## Asymptotic Equipartition Property (AEP)

# Weak Law of Large Numbers

For independent, identically distributed (IID) RVs $X_i$, then as $n \to \infty$

$$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{p} E[X_i]$$

where convergence is in probability.





Weak Law of Large Numbers
For independent, identically distributed (IID) RVs $X_i$, then as $n \to \infty$
$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{p} E[X_i]$
where convergence is in probability.

Convergence in probability means

$$\lim_{n\to\infty} P\left(\left|\bar{X} - E[X_i]\right| > \epsilon\right) = 0.$$

where $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, and any $\epsilon > 0$.

Strong Law of Large Numbers says the same thing, but convergence is almost sure, i.e.,

$$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{a.s.} E[X_i],$$

where almost sure convergence means

$$P\left(\lim_{n\to\infty} \bar{X} = E[X_i]\right) = 1.$$

A more general result is the Central Limit Theorem.

# AEP

- Uses the Law of Large Numbers to find an approximation for entropy in terms we can realize from observed sequences
- Flipping it around, probabilities of observed sequences of $n$ symbols will be close to $2^{-nH}$
  - almost all events are equally surprising
- Allows division of possible sequences into
  - typical
  - non-typical

  Properties proved for typical set will be true with high probability.





AEP
- Uses the Law of Large Numbers to find an approximation for entropy in terms we can realize from observed sequences
- Flipping it around, probabilities of observed sequences of $n$ symbols will be close to $2^{-nH}$
  - almost all events are equally surprising
- Allows division of possible sequences into
  - typical
  - non-typical

Properties proved for typical set will be true with high probability.

## AEP formalized

### Theorem (AEP)

*If $X_1, X_2, \ldots$ are IID with PMF $p(x)$, then*

$$-\frac{1}{n}\log P(x_1, x_2, \ldots, x_n) \xrightarrow{p} H(X)$$

### Proof.

Functions of independent RVs are also independent RVs, so the $P(X_i)$ and $\log P(X_i)$ are IID RVs, so

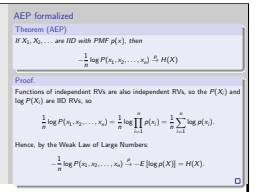
$$\frac{1}{n}\log P(x_1, x_2, \ldots, x_n) = \frac{1}{n}\log\prod_{i=1}^{n} p(x_i) = \frac{1}{n}\sum_{i=1}^{n}\log p(x_i).$$

Hence, by the Weak Law of Large Numbers:

$$-\frac{1}{n}\log P(x_1, x_2, \ldots, x_n) \xrightarrow{p} -E\left[\log p(X)\right] = H(X).$$







Again, notice that it is convergence in probability. This is sometimes called the weak AEP (similarly to the weak Law of Large Numbers). There is an equivalent strong AEP, with almost sure convergence (as in the Strong Law of Large Numbers).

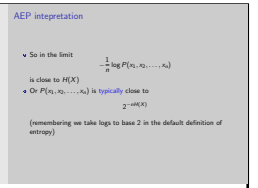
The AEP can also be extended to deal with more general stationary, ergodic stochastic processed, where the convergence is to the Entropy Rate.

## AEP intepretation

- So in the limit
$$-\frac{1}{n}\log P(x_1, x_2, \ldots, x_n)$$
is close to $H(X)$
- Or $P(x_1, x_2, \ldots, x_n)$ is typically close to
$$2^{-nH(X)}$$
(remembering we take logs to base 2 in the default definition of entropy)







## Typical Sequences

### Definition (typical)

The typical set $A_\epsilon^{(n)}$ with respect to the PMF $p(x)$ is the set of sequences $(x_1, x_2, \ldots, x_n) \in \Omega^n$ with the property

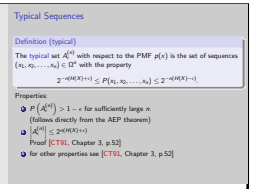
$$2^{-n(H(X)+\epsilon)} \leq P(x_1, x_2, \ldots, x_n) \leq 2^{-n(H(X)-\epsilon)}.$$

Properties:

1. $P\left(A_\epsilon^{(n)}\right) > 1 - \epsilon$ for sufficiently large $n$.
   (follows directly from the AEP theorem)
2. $\left|A_\epsilon^{(n)}\right| \leq 2^{n(H(X)+\epsilon)}$
   Proof [CT91, Chapter 3, p.52]
3. for other properties see [CT91, Chapter 3, p.52]





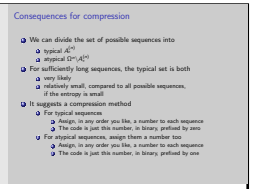



## Consequences for compression

1. We can divide the set of possible sequences into
   1. typical $A_\epsilon^{(n)}$
   2. atypical $\Omega^n \backslash A_\epsilon^{(n)}$
2. For sufficiently long sequences, the typical set is both
   1. very likely
   2. relatively small, compared to all possible sequences, if the entropy is small
3. It suggests a compression method
   1. For typical sequences
      1. Assign, in any order you like, a number to each sequence
      2. The code is just this number, in binary, prefixed by zero
   2. For atypical sequences, assign them a number too
      1. Assign, in any order you like, a number to each sequence
      2. The code is just this number, in binary, prefixed by one







## Consequences for compression

1. It suggests a compression method
   1. For typical sequences the code is has binary length, at most

$$\ell = n(H + \epsilon) + 1 + 1$$

      1. There are less than $2^{n(H+\epsilon)}$ sequences, so we need numbers with $n(H + \epsilon)$ bits.
      2. The first $+1$ arise from prefixing with a zero
      3. The second $+1$ arise because $n(H + \epsilon)$ might not be an integer
   2. For atypical sequences the code is has binary length, at most

$$\ell = n \log_2 |\Omega| + 1 + 1$$

      1. The first $+1$ arise from prefixing with a one
      2. The second $+1$ arise because $n \log_2 |\Omega|$ might not be an integer





Consequences for compression
1. It suggests a compression method
   1. For typical sequences the code is has binary length, at most
$\ell = n(H+\epsilon)+1+1$
      1. There are less than $2^{n(H+\epsilon)}$ sequences, so we need numbers with $n(H+\epsilon)$ bits.
      2. The first $+1$ arise from prefixing with a zero
      3. The second $+1$ arise because $n(H+\epsilon)$ might not be an integer
   2. For atypical sequences the code is has binary length, at most
$\ell = n\log_2|\Omega|+1+1$
      1. The first $+1$ arise from prefixing with a one
      2. The second $+1$ arise because $n\log_2|\Omega|$ might not be an integer

## Consequences for compression

**Theorem (Expected Message Length)**

*If $X_1, X_2, \ldots$ are IID with PMF $p(x)$, then for any $\epsilon' > 0$, there exists a code which maps sequences of length $n$ into binary strings such that the mapping is one-to-one) and therefore invertible and*

$$E\left[\frac{1}{n}\ell(X_1, X_2, \ldots, X_n)\right] \leq H(X) + \epsilon',$$

*for $n$ sufficiently large.*





Consequences for compression
Theorem (Expected Message Length)
If $X_1, X_2, \ldots$ are IID with PMF $p(x)$, then for any $\epsilon' > 0$, there exists a code which maps sequences of length $n$ into binary strings such that the mapping is one-to-one) and therefore invertible and
$E\left[\frac{1}{n}\ell(X_1, X_2, \ldots, X_n)\right] \leq H(X) + \epsilon',$
for $n$ sufficiently large.

We already basically know this, but note that the coding method proposed above is MUCH simpler than Huffman coding. It doesn't require estimates of actual probabilities, just whether a sequence is typical or not. So we can now deal with much larger blocks if we like.

## Consequences for compression

**Proof.**

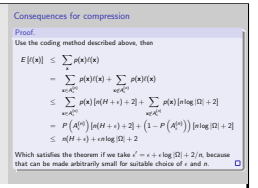
Use the coding method described above, then

$$\begin{aligned} E[\ell(\mathbf{x})] &\leq \sum_{\mathbf{x}} p(\mathbf{x})\ell(\mathbf{x}) \\ &= \sum_{\mathbf{x}\in A_\epsilon^{(n)}} p(\mathbf{x})\ell(\mathbf{x}) + \sum_{\mathbf{x}\notin A_\epsilon^{(n)}} p(\mathbf{x})\ell(\mathbf{x}) \\ &\leq \sum_{\mathbf{x}\in A_\epsilon^{(n)}} p(\mathbf{x})\left[n(H+\epsilon)+2\right] + \sum_{\mathbf{x}\notin A_\epsilon^{(n)}} p(\mathbf{x})\left[n\log|\Omega|+2\right] \\ &= P\left(A_\epsilon^{(n)}\right)\left[n(H+\epsilon)+2\right] + \left(1-P\left(A_\epsilon^{(n)}\right)\right)\left[n\log|\Omega|+2\right] \\ &\leq n(H+\epsilon)+\epsilon n\log|\Omega|+2 \end{aligned}$$

Which satisfies the theorem if we take $\epsilon' = \epsilon + \epsilon\log|\Omega| + 2/n$, because that can be made arbitrarily small for suitable choice of $\epsilon$ and $n$. □







## Consequences for compression

**Corollary**

*Don't code per symbol!*

- The above gives us a bound on coding of $H(X)$ bits per symbol in the original sequence.
- Simple counter example:
  - Sequence

    *aaaaaaaaaaaaaaaaaaaa*

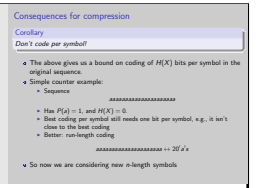
  - Has $P(a) = 1$, and $H(X) = 0$.
  - Best coding per symbol still needs one bit per symbol, e.g., it isn't close to the best coding
  - Better: run-length coding

    $$aaaaaaaaaaaaaaaaaaaa \leftrightarrow 20' a's$$

- So now we are considering new $n$-length symbols







Section 2

Some Compression algorithms





Section 2
Some Compression algorithms

## Run length encoding (RLE)

If our data has many sequences of the same symbol

- record the symbols, and how long each run is, so

  *aaaaabbbbbaaaaaaabbbbbaaaaaaaaabbbbb*

  becomes

  5*a*5*b*7*a*5*b*9*a*5*b*

- 36 symbols becomes 12
  - "alphabet" may be bigger though, as now we include numbers
- Compression factor depends on the data, a lot.



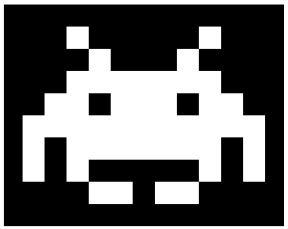


Run length encoding (RLE)
If our data has many sequences of the same symbol
- record the symbols, and how long each run is, so
aaaaabbbbbaaaaaaabbbbbaaaaaaaaabbbbb
becomes
5a5b7a5b9a5b
- 36 symbols becomes 12
  - "alphabet" may be bigger though, as now we include numbers
- Compression factor depends on the data, a lot.

We are starting here to see that sequences are important – not just frequencies.

## Run length encoding (RLE)

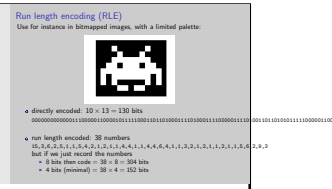
Use for instance in bitmapped images, with a limited palette:



- directly encoded: $10 \times 13 = 130$ bits
  0000000000000111000001100001011111000110110100011110100011110000011110100110110101011111000001100000000111000000000
- run length encoded: 38 numbers
  15,3,6,2,5,1,1,5,4,2,1,2,1,1,4,4,1,1,4,4,6,4,1,1,3,2,1,2,1,1,2,1,1,5,6,2,9,3
  but if we just record the numbers
  - 8 bits then code $= 38 \times 8 = 304$ bits
  - 4 bits (minimal) $= 38 \times 4 = 152$ bits







Another example is fax (mostly white space, with a few black dots).

Space invader string is in column ordering.

## Run length encoding (RLE)

- Run length encoded: 38 numbers
  15,3,6,2,5,1,1,5,4,2,1,2,1,1,4,4,1,1,4,4,6,4,1,1,3,2,1,2,1,1,2,1,1,5,6,2,9,3
  but if we just record the numbers
  - 8 bits then code $= 38 \times 8 = 304$ bits
  - 4 bits (minimal) $= 38 \times 4 = 152$ bits
- What if we Huffman encode the numbers?

$$H(X) \simeq 2.54$$

So the total number of bits (assuming efficient encoding) would be

$$38 \times 2.54 \simeq 97 \text{ bits}$$

which is slightly better than 130 bits for the raw file.

- Compare Huffman coding of original with blocks of 5 gives about 73 bits, so we may as well just do a raw Huffman code.



## Further reading I

Thomas M. Cover and Joy A. Thomas, *Elements of information theory*, John Wiley and Sons, 1991.

Raymond W. Yeung, *Information theory and network coding*, Springer, 2010.