Arithmetic coding

- Elias coding
- Arithmetic coding as finite-precision Elias coding
- Redundancy due to finite precision
- Multiplication-free arithmetic coding
- Context-adaptive coding
Elias coding

- Entropy coding algorithm for sequences of symbols \( x \) with general (conditional) probabilities
- Representation of \( x \) by a subinterval of the unit interval \([0,1)\)
- Width of the subinterval is approximately equal to the probability \( f_x(x) \)
- Subinterval for \( x \) can be determined by recursive subdivision algorithm
- Represent \( x \) by shortest binary fraction in the subinterval
- Subinterval of width \( f_x(x) \) is guaranteed to contain one number that can be represented by \( L \) binary digits, with

\[
L \approx -\log_2 f_x(x)
\]
Example: Elias coding of memoryless binary source

\[
\begin{align*}
    f_x(0) &= \frac{1}{4} \\
    f_x(1) &= \frac{3}{4}
\end{align*}
\]
Example: Elias coding of memoryless ternary source

\[
f_X(a) = f_X(b) = \frac{1}{4}, \quad f_X(c) = \frac{1}{2}
\]
Elias coding: choose binary fraction in subinterval

- Uniquely identify interval \([c_n, c_n + a_n]\) by a binary fraction

\[
0.bbbbb\ldots b \in [c_n, c_n + a_n]
\]

- Length of bit-string

\[
L_n = \left\lfloor \log_2 \frac{1}{a_n} \right\rfloor + 1 = \left\lfloor h_{x_{0:n-1}} \left( x_{0:n-1} \right) \right\rfloor + 1
\]

- Bit-string

\[
\hat{c}_n = 2^{-L_n} \left\lfloor 2^{L_n} c_n + 1 \right\rfloor > c_n
\]
Successive decoding

- Elias bit string can be decoded symbol by symbol, starting with the first symbol.
- For each symbol \( n=0, 1, 2, \ldots \):
  - Calculate the intervals \([c_n, c_n + a_n]\) corresponding to all possible \( x_n \).
  - Determine the interval for which \( \hat{c}_n \in [c_n, c_n + a_n] \).
  - Emit the corresponding \( x_n \).
Elias coding for memoryless source

Elias coding algorithm

Initialize $c_0 = 0$ and $a_0 = 1$.

For each $n = 0, 1, \ldots$

Update $a_{n+1} \leftarrow a_n f_X (x_n)$

Update $c_{n+1} \leftarrow c_n + a_n F_X (x_n)$

Start with unit interval

Interval width

Interval lower limit

$n$-th symbol to be encoded

Cumulative distribution (excluding current symbol)

$$F_X (\alpha_i) = \sum_{j=0}^{i-1} f_X (\alpha_j)$$

where $A_X = \{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{K-1}\}$
Elias coding for changing probabilities

Elias coding algorithm

Initialize $c_0 = 0$ and $a_0 = 1$.

For each $n = 0, 1, \ldots$

Update $a_{n+1} \leftarrow a_n f_{X_n}(x_n)$

Update $c_{n+1} \leftarrow c_n + a_n F_{X_n}(x_n)$

Interval width

Interval lower limit

Start with unit interval

$n$-th symbol to be encoded

Cumulative distribution (excluding current symbol)

$$F_{X_n}(\alpha_i) = \sum_{j=0}^{i-1} f_{X_n}(\alpha_j)$$

where $A_{X_n} = \{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{K-1}\}$
Elias coding for Markov random processes

Elias coding algorithm for Markov-p sources

Initialize $c_0 = 0$ and $a_0 = 1$.
For each $n = 0, 1, \ldots$

Set $p' = \min \{p, n\}$

Update $a_{n+1} \leftarrow a_n f_{x_n|x_{n-p':n-1}}(x_n, x_{n-p':n-1})$

Update $c_{n+1} \leftarrow c_n + a_n F_{x_n|x_{n-p':n-1}}(x_n, x_{n-p':n-1})$

Interval width

Interval lower limit

Start with unit interval

Up to $p$ previous symbols

Cumulative conditional distribution (excluding current symbol)

$n$-th symbol to be encoded
Arithmetic coding

- Elias coding not practical for long symbol strings: required arithmetic precision grows with string length
- Finite precision implementations: “arithmetic coding”
- Widely used in modern image and video compression algorithms: JBIG, JPEG, JPEG-2000, H.263, H.264/AVC

\[ x_n \]

Arithmetic coder

\[ f_{X_n}(x_n) \]

bit stream \( c \)

Arithmetic decoder

\[ f_{X_n}(x_n) \]
Finite precision for arithmetic coding

Initialize $c_0 = 0$ and $a_0 = 1$.
For each $n = 0, 1, \ldots$

Update $a_{n+1} \leftarrow a_n f_X (x_n)$

Update $c_{n+1} \leftarrow c_n + a_n F_X (x_n)$

$N$-bit precision is permissible, if rounding down

At most $N+P$ LSBs change, except for a possible carry affecting MSBs. Each bit can be affected by at most one carry.

$P$-bit approximation of probabilities
Finite precision for arithmetic coding (cont.)

- Output code string

\[ c_n = \underbrace{xxxxxxx...x}_{\text{MSBs that will no longer change}} \underbrace{01111111...1}_{r_n \text{ bits that might be affected by a carry}} \underbrace{cccccccc...c}_{N+P \text{ LSBs that represent lower interval boundary for further computation}} \]

Send

Represent by state variable \( r_n \)

Represent by state variable \( c_n \)

- Maximum value of \( r_n \) can be limited by bit stuffing
Inefficiency due to interval rounding

- **Recall:** Subinterval of width \( f_X(x) \) is guaranteed to contain one number that can be represented by \( L_n \) binary digits, with

\[
L_n \approx -\log_2 f_{X_{0:n-1}} (x_{0:n-1})
\]

- Hence, rounding one interval value \( a_{n+1} \) increases bit string by

\[
\log_2 \frac{a_n f_X(x_n)}{a_{n+1}} < \log_2 \frac{2^{N-1} + 1}{2^{N-1}} = \log_2 \left( 1 + 2^{1-N} \right) \approx \frac{1}{\ln 2} 2^{1-N}
\]
Limited precision probabilities

- Efficiency loss

\[ E \left[ -\log_2 \left( f_X(x) \left(1 + \frac{\Delta f_X(x)}{f_X(x)} \right) \right) \right] = E \left[ -\log_2 \left( f_X(x) \right) \right] - E \left[ \log_2 \left( 1 + \frac{\Delta f_X(x)}{f_X(x)} \right) \right] \]

\[ = H(X) \]

\[ \approx \frac{1}{\ln 2} E \left[ \frac{\Delta f_X(x)}{f_X(x)} \right] \]

\[ E \left[ \frac{\Delta f_X(x)}{f_X(x)} \right] = \sum_x f_X(x) \cdot \frac{\Delta f_X(x)}{f_X(x)} = \sum_x \Delta f_X(x) = 0 \]

provided that rounded probabilities still add to 1
Example: inefficiency for binary source due to representing smaller probability by $2^{-p}$
Interval width is normalized to $1/2 \leq A < 1$ by binary shift after each symbol.

Approximation, with $p(L) \leq p(M)$:

- $Ap(L) \approx \alpha \cdot p(L) = p'(L)$
- $Ap(M) = A(1 - p(L)) \approx A - \alpha \cdot p(L) = A - p'(L)$

Corresponds to approximating of $p(L)$ within a factor of 2:

$$p(L) \approx \frac{\alpha}{A} p(L)$$

Factor $\alpha$ can be lumped into external probabilities.

Optimum $\alpha$ depends somewhat on data, typically $\alpha = \frac{2}{3}$ or $\alpha = 0.708$ or $\alpha = \frac{3}{4}$.
MPS-LPS switch

- **Problem**: interval width for symbol $M$ can become negative
  \[ A - p'(L) < 0 \quad \text{if} \quad p'(L) > A \in \left[ \frac{1}{2}, 1 \right] \]

- **Solution**: switch the role of $M$ and $L$, if $p'(L) > \frac{1}{2}$

- **Problem**: interval for symbol $M$ can be smaller than interval for $L$ if
  \[ p'(L) > A - p'(L) \iff p'(L) > \frac{1}{2} A \in \left[ \frac{1}{4}, \frac{1}{2} \right] \]

- **Solution**: “Conditional exchange” of symbols, when problem occurs

- Improves compression efficiency, implemented in JPEG and JBIG
Sufficiency of binary coders

- Consider r.v. $X \in \{0,1,\ldots,2^K-1\}$ equivalent to a r.v. $B = \begin{pmatrix} B_0 \\ B_1 \\ \vdots \\ B_{K-1} \end{pmatrix}$

- No loss by conditionally encoding bits (in any order)
  \[
  H(X) = H(B) = H(B_0) + H(B_1|B_0) + \ldots + H(B_{K-1}|B_0,B_1,\ldots,B_{K-2})
  \]

- Supply arithmetic coder with symbol/probability pairs
  \[
  (b_0, p_0), (b_1, p_1), \ldots, (b_{K-1}, p_{K-1}) \quad \text{with} \quad p_k = f_{B_k|B_0:k-1}(0,b_{0:k-1})
  \]

- Total number of (conditional) probabilities to be estimated per symbol $x$
  \[
  1 + 2 + \ldots + 2^{K-1} = 2^K - 1
  \]

  \[
  \ldots \text{same as with } K\text{-ary encoding of } x
  \]
Adaptive probability estimation

- Consider stationary binary Markov-$k$ process $\{X_n\}$ with probabilities

$$f_{X_k|X_{0:k-1}}(0, x_{0:k-1}) = f_{X_n|X_{(n-k):n-1}}(0, x_{(n-k):n-1})$$

"Context vector," $2^k$ possible combinations

- Context labeling function, enumerating all possible context vectors

$$\lambda(x) : x \in \{0, 1\}^k \rightarrow \{0, 1, \ldots, 2^k - 1\}$$

- Backward probability estimation separately for each context

\[
p_0[n] = \frac{C_0\left[n, \lambda(x_{(n-k):n-1})\right] + \Delta}{\left(C_0\left[n, \lambda(x_{(n-k):n-1})\right] + \Delta\right) + \left(C_1\left[n, \lambda(x_{(n-k):n-1})\right] + \Delta\right)}
\]

Probability of symbol $x[n]=0$

Count of zeroes that have previously occurred in that context

Bias for low counts, typically $\Delta=1$
Adaptive probability estimation (cont.)

- Tracking changing statistics, compromise between
  - Rapid adaptation (small past sample counts)
  - Accurate estimates (large past sample counts)
- Scaled count estimation algorithm

Initialize $C_0 = C_1 = 0$

For $n = 0,1,\ldots$

If $x_n = 1$
  
  $C_1 \leftarrow C_1 + 1$

else

  $C_0 \leftarrow C_0 + 1$

If $\min\{C_0, C_1\} > C_{\min}$ or $\max\{C_0, C_1\} > C_{\max}$

  $C_0 \leftarrow \lfloor \frac{C_0}{2} \rfloor$;  
  $C_1 \leftarrow \lfloor \frac{C_1}{2} \rfloor$

Estimate $p_0[n] = \frac{C_0 + \Delta}{C_0 + C_1 + 2\Delta}$
Context adaptive coding

- 2-d extension of Markov model

```
\[
f_{X[n]|X_{N+n}}(X[n], X_{N+n}) = f_{X[n]|X_{N+n}}(X[n], X_{N+n})
\]
```

- Markov conditional independence property

```
\[
\text{“causal half-plane” } N^\infty + n
\]
```

“Coding context” \( N + n \)
Context adaptive coding (cont.)

- Coding context vector can be mapped into context label directly without loss
  - Feasible for binary images
  - Example: JBIG uses 10 binary pixels as context label
- For 8-bit images, number of different contexts $2^{56}$
  - Context might have to be clustered
  - Combine with prediction
Reading

