Lossless Source Coding

Geometric distributions and Golomb codes – Part 2

Golomb “Power of 2” (PO2) codes

- The expected code length for G_m (unrestricted Golomb) is

$$L_{G_m} = \lceil \log m \rceil + 1 + \frac{\gamma^t}{1-\gamma^m} \qquad (t = 2^{\lfloor \log m \rfloor + 1} - m)$$

- For a *Golomb PO2 code* $G_k^* = G_{2^k}$, we have

$$L_k^* \triangleq L_{G_{2^k}} = k + 1 + \frac{\gamma^{2^k}}{1-\gamma^{2^k}}$$

- Since the PO2 codes are simpler to implement, we will attempt to use the best G_k^* code for γ, instead of the *optimal* G_m.
- Questions:
  - How costly is this sub-optimality?
  - Are there efficient ways to adapt the choice of parameter k to the data?

Golomb "Power of 2" (PO2) codes

$$L_k^*(\gamma) = k + 1 + \frac{\gamma^{2^k}}{1-\gamma^{2^k}} = k + 1 + \frac{z}{1-z} \qquad (z \triangleq \gamma^{2^k})$$

- We want to use the code G_k^* with parameter k that *minimizes* $L_k^*(\gamma)$

$$L_{\min}^*(\gamma) = \min_{k \geq 0} L_k^*(\gamma)$$

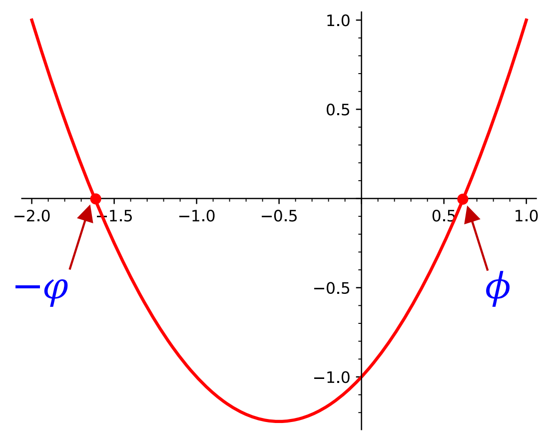
- Say for some value of γ we know the best choice of k. Start varying γ. When does k stop being the best?
- Transition point $k \to k+1$ is at γ such that $L_k^*(\gamma) = L_{k+1}^*(\gamma)$. Notice that $k \to k+1$ implies $z \to z^2$.

$$L_k^* - L_{k+1}^* = \frac{z}{1-z} - 1 - \frac{z^2}{1-z^2} = \frac{z^2+z-1}{1-z^2} = 0 \iff z = \frac{-1 \pm \sqrt{5}}{2}$$

Clearly, we must choose the positive root

$$z = \frac{\sqrt{5}-1}{2} \triangleq \phi \approx 0.618\ldots$$

- ϕ is the inverse of the *golden ratio* $\varphi = \frac{\sqrt{5}+1}{2} \approx 1.618$ (also $\phi = \varphi - 1$)



Golomb "Power of 2" (PO2) codes

- Transition $k \to k+1$ occurs at

$$z = \gamma^{2^k} = \phi, \quad k = 0, 1, 2, \ldots$$

- Transition points for γ:

$$\gamma_k^* = \phi^{2^{-k}}, \quad k = 0, 1, 2, \ldots$$

Transitions for Golomb PO2

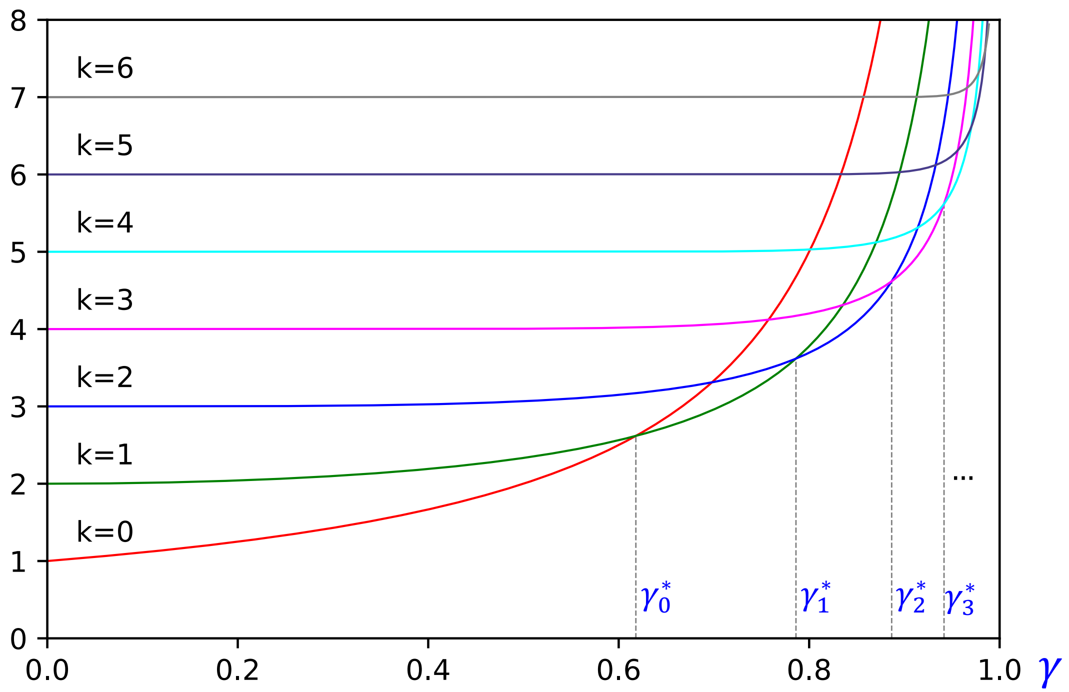
k	γ_k^*
0	0.61803399
1	0.78615138
2	0.88665178
3	0.94162189
4	0.97037204
5	0.98507463
6	0.99250926
7	0.99624759
8	0.99812203

Transitions for unrestricted Golomb

m	γ_m
1	0.61803399
2	0.75487766
3	0.81917251
4	0.85667488
5	0.88127146
6	0.89865371
7	0.91159235
8	0.92159931

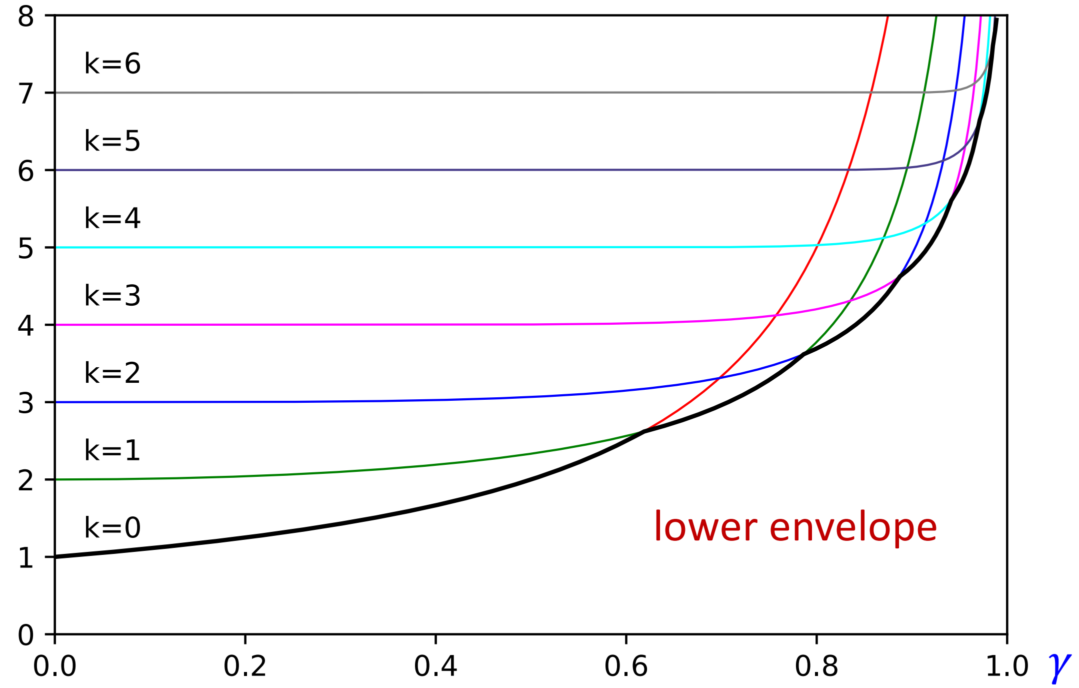
Average code length for Golomb PO2 codes

$$\boxed{L_k^*(\gamma) = k + 1 + \frac{\gamma^{2^k}}{1 - \gamma^{2^k}}}$$



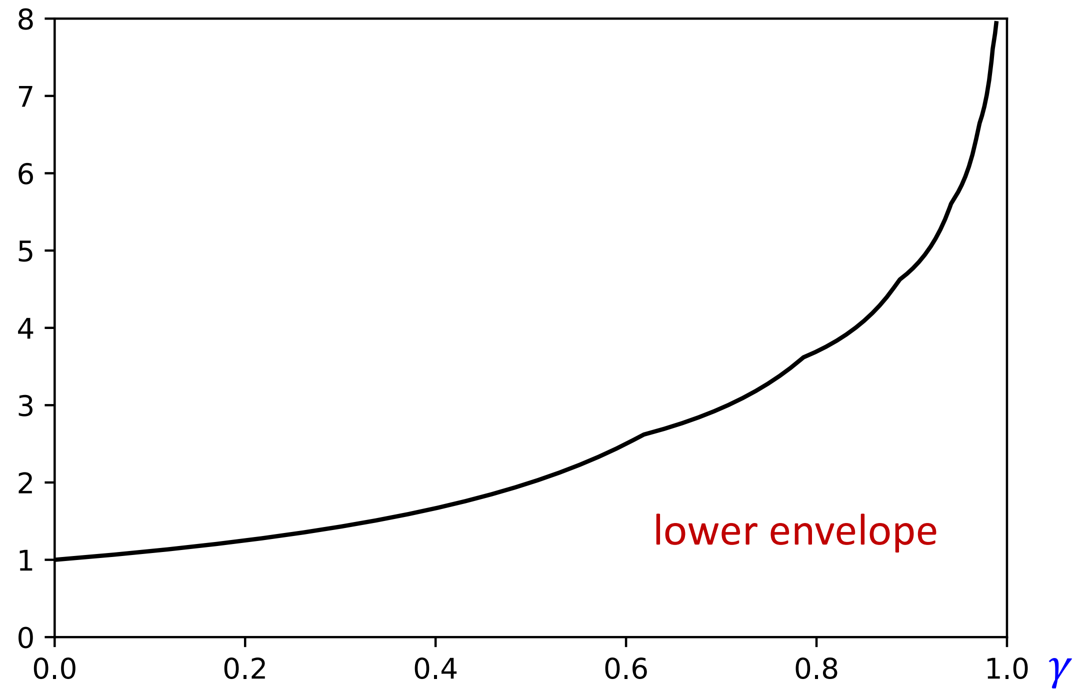
Average code length for Golomb PO2 codes

$$L^*_{\min}(\gamma) = \min_k L^*_k(\gamma)$$



Average code length for Golomb PO2 codes

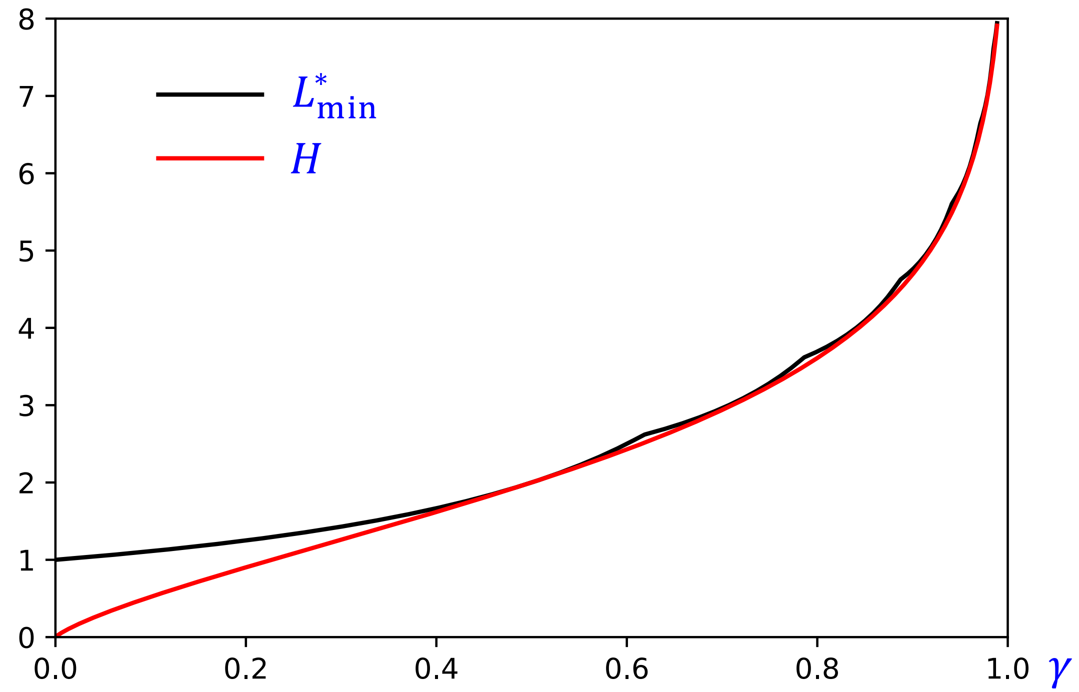
$$\boxed{L_{\min}^*(\gamma) = \min_k L_k^*(\gamma)}$$



Average code length vs entropy for Golomb PO2 codes

$$L^*_{\min}(\gamma) = \min_k L^*_k(\gamma)$$

$$H(\gamma) = \frac{h_2(\gamma)}{1-\gamma}$$

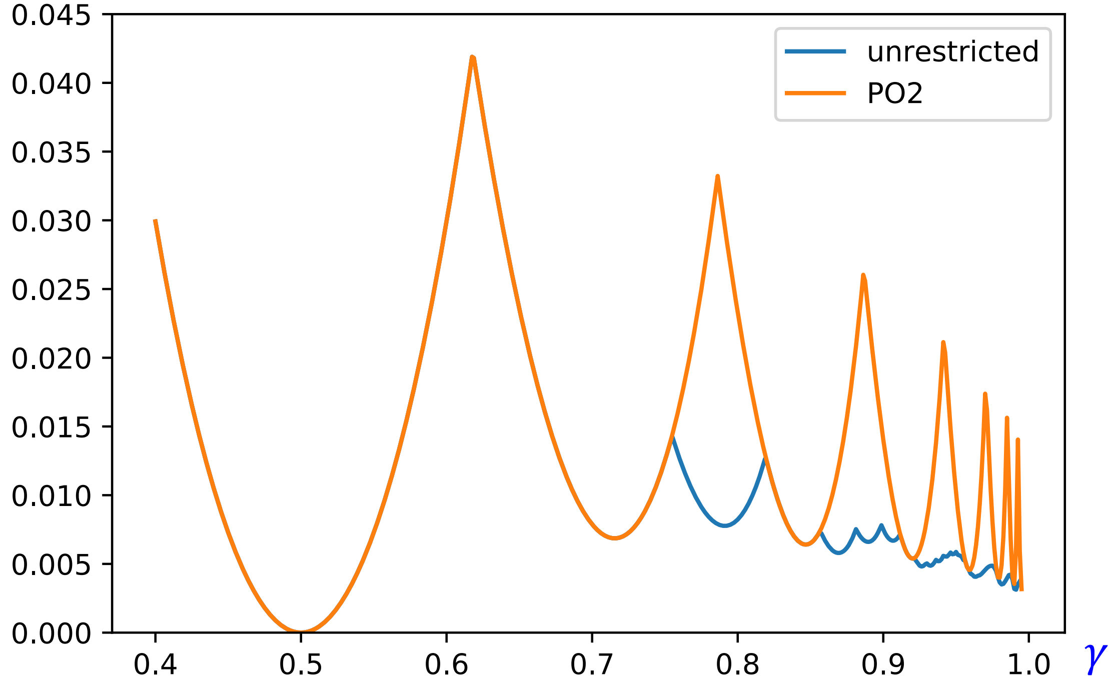


Relative redundancy: Golomb unrestricted and PO2 codes

$$R_{\mathrm{rel}}(\gamma) = \frac{L_{\min}(\gamma) - H(\gamma)}{H(\gamma)}$$

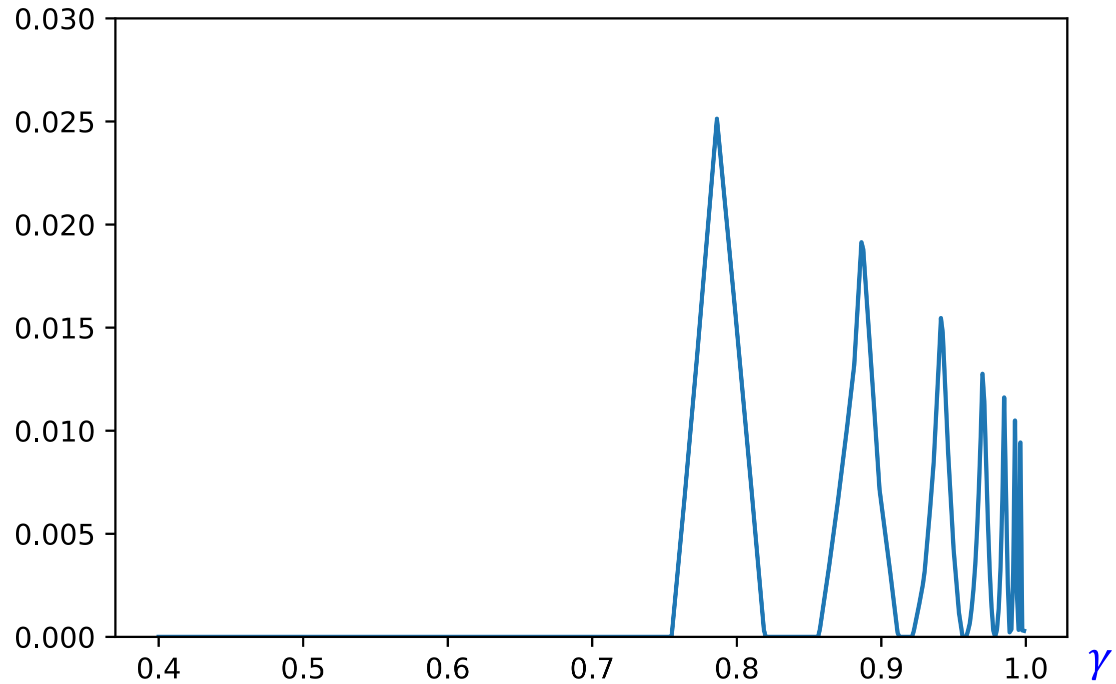
Here $L_{\min}$ is either $L^*_{\min}$ (PO2) or $L^G_{\min}$ (unrestricted)

$$L^G_{\min}(\gamma) = \min_m L_{G_m}(\gamma)$$



Relative code length penalty for PO2 vs unrestricted Golomb codes

$$\Delta_{\mathrm{rel}} = \frac{L^*_{\min} - L^{\mathrm{G}}_{\min}}{L^{\mathrm{G}}_{\min}}$$



Parameter estimation

- The situation where we know γ and can choose the best code accordingly is unrealistic.
- Given a sequence of i.i.d. samples x_1^n, presumably from a GD (with γ *unknown*), how do we choose the best code to use?
  - Option 1: try $m = 1, 2, 3, \ldots$ and find the minimum $L_{G_m}(x_1^n)$ (or, if using only PO2 codes, try $k = 0, 1, 2, \ldots$ and find the minimum $L_k^*(x_1^n)$).
  - Option 2: *estimate* γ, use the best code for γ (whether unrestricted or PO2).
- Option 1 would be ideal, but it implies encoding (or at least computing code length) multiple times.
  Option 2 is simpler and gets very close! (If we use the right estimate.)
  - in fact, for PO2 codes, we'll skip the estimation of γ and estimate the best code (best k) *directly*.

Maximum likelihood estimation of γ

- Maximum Likelihood (ML) estimation of γ for a sequence x_1^n
  - Let
$$S(x_1^n) \triangleq \sum_{i=1}^{n} x_i$$
  - We have
$$P_\gamma(x_1^n) = \prod_{i=1}^{n} P_\gamma(x_i) = \prod_{i=1}^{n} (1-\gamma)\gamma^{x_i} = (1-\gamma)^n \gamma^{\sum_{i=1}^{n} x_i} = (1-\gamma)^n \gamma^{S(x_1^n)}$$
    S is a *minimal sufficient statistic* for the GD. (We will omit the argument x_1^n.)
  - The ML estimate of γ for is the value that maximizes $P_\gamma(x_1^n)$:
$$\frac{dP_\gamma}{d\gamma} = -n(1-\gamma)^{n-1}\gamma^S + S(1-\gamma)^n\gamma^{S-1}$$
$$= (1-\gamma)^{n-1}\gamma^{S-1}\left[-n\gamma + S(1-\gamma)\right]$$

derivative vanishes at $\hat{\gamma} = \frac{S}{S+n}$ *ML estimate of γ*

Example: best code vs ML estimate

- Choosing the code from the ML estimate of γ does not necessarily yield the best code length for an individual sequence x_1^n.
  - The codes have positive redundancy, and only approximate an ideal code length $-\log P(x)$.
  - The ML estimate of γ will incur some error, which decreases with larger n.

- Consider $x_1^6 = 022222 \quad (n = 6, \;\; S = \sum_i x_i = 10)$.
  - We have $\hat{\gamma} = \frac{S}{S+n} = \frac{10}{16} = 0.625 \in (0.618, 0.786)$ $\Rightarrow$ choose G_1^*.
    $L_1(x_1^6) = 2 + 5 \cdot 3 = 17$ bits
  - However,
    $L_0(x_1^6) = 1 + 5 \cdot 3 = 16$ bits

- Nevertheless, we will see that, asymptotically, an ML-based strategy will incur a negligible code length penalty vs exhaustively searching and choosing the best code.

k	γ_k^*
0	0.61803399
1	0.78615138
2	0.88665178
3	0.94162189
4	0.97037204
5	0.98507463
6	0.99250926
7	0.99624759
8	0.99812203

Choosing a PO2 code using approximate ML estimation

$$\hat{\gamma} = \frac{S}{S+n} \quad \Rightarrow \quad S = \frac{n}{1-\hat{\gamma}}$$

At transition points $\gamma_k^* = \phi^{2^{-k}}$ we have

$$\frac{S_k}{n} = \frac{1}{1-\phi^{2^{-k}}}$$

k	S_k/n
0	2.62
1	4.68
2	8.82
3	17.13
4	33.75
5	67.00
6	133.50
7	266.50
8	532.49
9	1064.48
10	2128.46

Estimation procedure for k

Given x^n:

- Compute $\frac{S(x_1^n)}{n}$
- Compare to thresholds on left
- Choose k and G_k^* accordingly

Observation: Thresholds $S_k/\,n$ are close to powers of 2.

Choosing a PO2 code using approximate ML estimation

$$\frac{S_k}{n} = \frac{1}{1-\phi^{2^{-k}}} \qquad (\phi \approx 0.618 < 1)$$

Why are these values close to powers of 2?

Taylor $e^{-x} \approx 1 - x$

$$1-\phi^{2^{-k}} = 1 - e^{-2^{-k}\ln\phi^{-1}} \approx 2^{-k}\ln\phi^{-1} \approx 2^{-k}\cdot 0.48 \approx 2^{-k-1}$$

$$\Rightarrow \frac{S_k}{n} = \frac{1}{1-\phi^{2^{-k}}} \approx 2^{k+1}$$

k	$\frac{S_k}{n} - \frac{1}{2}$
0	2.12
1	4.18
2	8.32
3	16.63
4	33.25
5	66.50
6	133.00
7	266.00
8	531.99
9	1063.98
10	2127.96

For small values of k, $\frac{S_k}{n} - \frac{1}{2} \approx 2^{k+1}$ is a better approximation (the correction makes little difference for larger values of k).

Simplified estimation procedure for k

- Compute $S(x_1^n)$
- Find smallest $k \geq 0$ such that $S(x_1^n) - \frac{n}{2} \leq 2^{k+1}n$
  A one-liner in C/C++!

```
for (k=0; (2*n)<<k < (S-n/2); k++);
```

Sequential coding

- So far, we have chosen a parameter k appropriate to encode x_1^n, *after seeing the whole sequence* (two-pass coding).
- In many applications, the sequence can be read only once, and *sequential coding* is required.
- Adaptive "plug-in" approach:
  - For each $t = 1,2,3, \ldots, n$ do
    Estimate k for x_1^{t-1} (exact estimation, with some arbitrary convention for $t = 1$)
    Encode x_t using G_k^*.

Theorem [Merhav, Seroussi, Weinberger 2000a, proven for TSGDs]
Let X_1^n be a random sequence of i.i.d. drawings from a GD with unknown parameter γ , let $L_{\mathrm{seq}}(X_1^n)$ denote the code length produced by the sequential encoder above, and let L_γ^* denote the expected per-symbol (normalized) code length produced by the *best* code G_k^* for (each realization of) X_1^n , derived from observing the whole sequence. Then,

$$\frac{1}{n} E_\gamma\left[L_{\mathrm{seq}}(x_1^n)\right] \leq L_\gamma^* + O(1/n\,)$$

($\Longrightarrow$ penalty for sequentiality is asymptotically negligible)

Summary: a compression algorithm for the GD

Input: sequence x_1^n

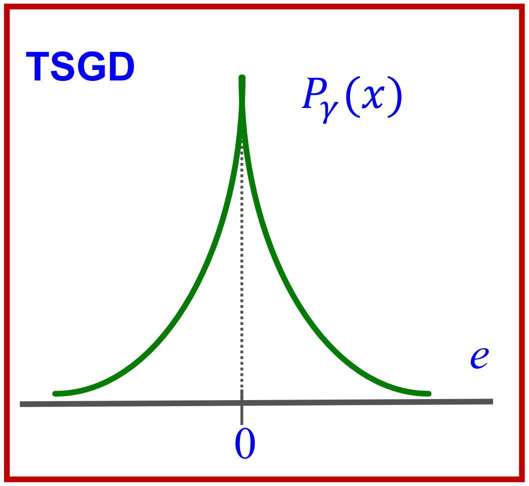
Output: binary compressed stream

```
1.  // Init
    t = 0;
    S = 0;
    k = 3;  // arbitrary, any reasonable value will do
2.  while ( t < n ) {
  a.  // Encode
      t = t + 1;
      mask = (1 << k) - 1;
      bin = x_t & mask;
      output(bin, k);          // output binary part in k bits
      ucount = (x_t >> k);     // count of zeros for unary
      output_zeros(ucount);    // output ucount zero bits
      output(1, 1);            // terminating 1 for unary
  b.  // Update stats
      S = S + x_t;
  c.  // Estimate next k
      for (k=0; (2*t)<<k < (S-t/2); k++);
    }
```

Coding two-sided geometric distributions

- We want to encode an integer-valued random variable with distribution

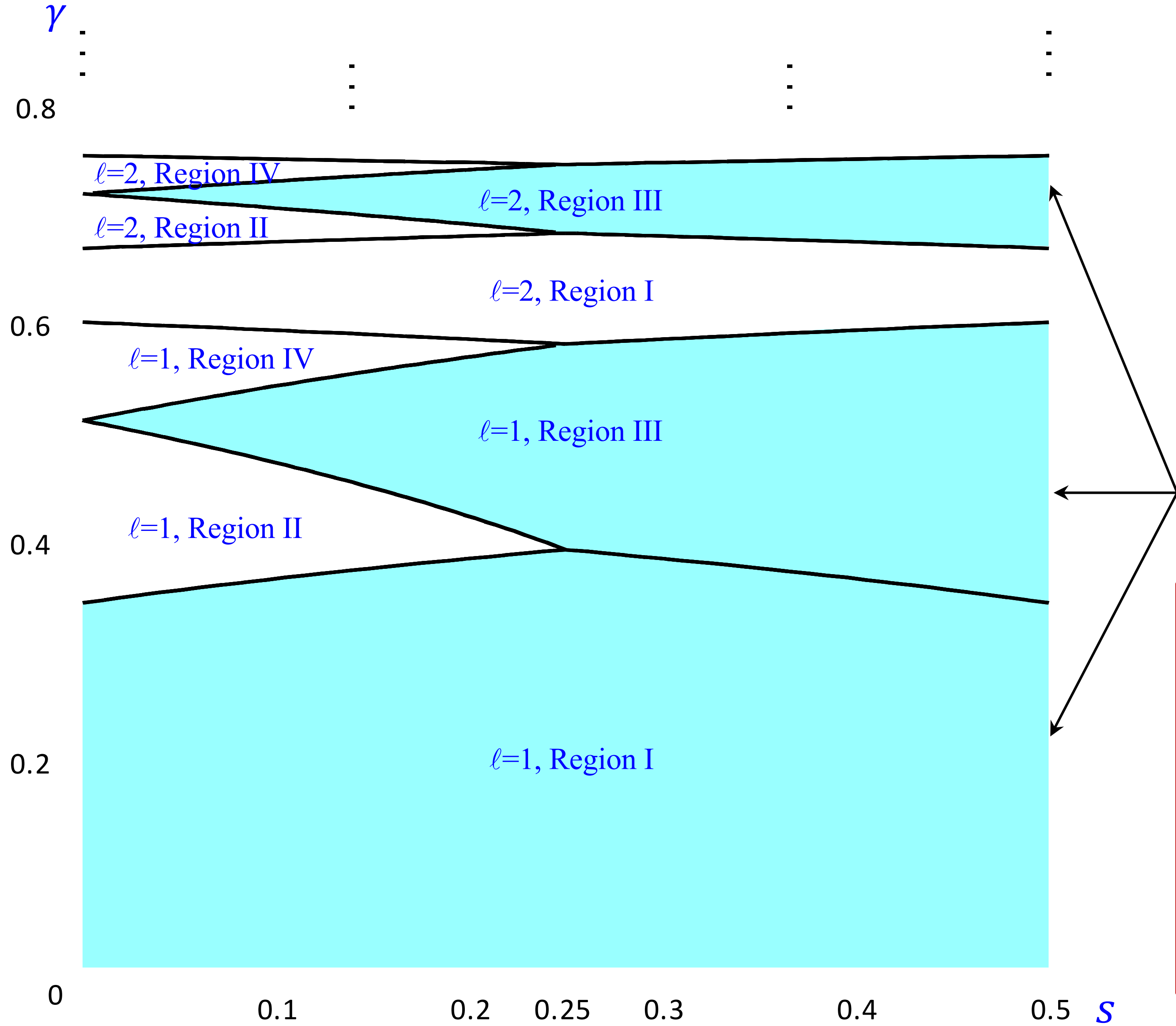
$$P_\gamma(x) = \frac{1+\gamma}{1-\gamma}\gamma^{|x|}$$



- Two "natural" approaches:
  1. Reorder the integers as $0, -1, 1, -2, 2, -3, 3, \ldots$ and encode the index with a Golomb code (even though it is *not* geometrically distributed)
  $$M(e) = 2|e| - (e < 0), \quad e \in \mathbb{Z}$$
  2. Encode $|x|$ (also not strictly GD) with a Golomb code and append the sign of x (1 bit) whenever $x \neq 0$.
- Neither is optimal for all γ, but both are optimal for *some* values of γ.
  - The $M(e)$ strategy works well with PO2 codes, and is used in many practical applications (e.g., the *JPEG-LS* lossless image compression standard).
- Optimal codes for TSGDs were fully characterized in [Merhav, Seroussi, Weinberger '2000b].
  - They use Golomb codes as building blocks, including the two schemes above, but also other less intuitive ones.

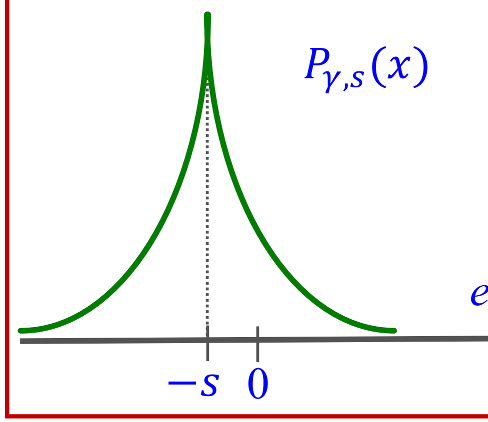
Optimal Code Regions for TSG Distributions

$$P_{\gamma,s}(e) = c_0 \gamma^{|e+s|}, \qquad 0 < \gamma < 1, \qquad 0 \le s \le 1/2.$$



Golomb-PO2 codes

TSGD with offset



Region I: $G_{2\ell-1}(M(x))$

Region III: $G_{2\ell}(M(x))$

Regions II, IV: *symmetric codes* (Region II is equiv. to sign+magnitude when ℓ is PO2).

THE END