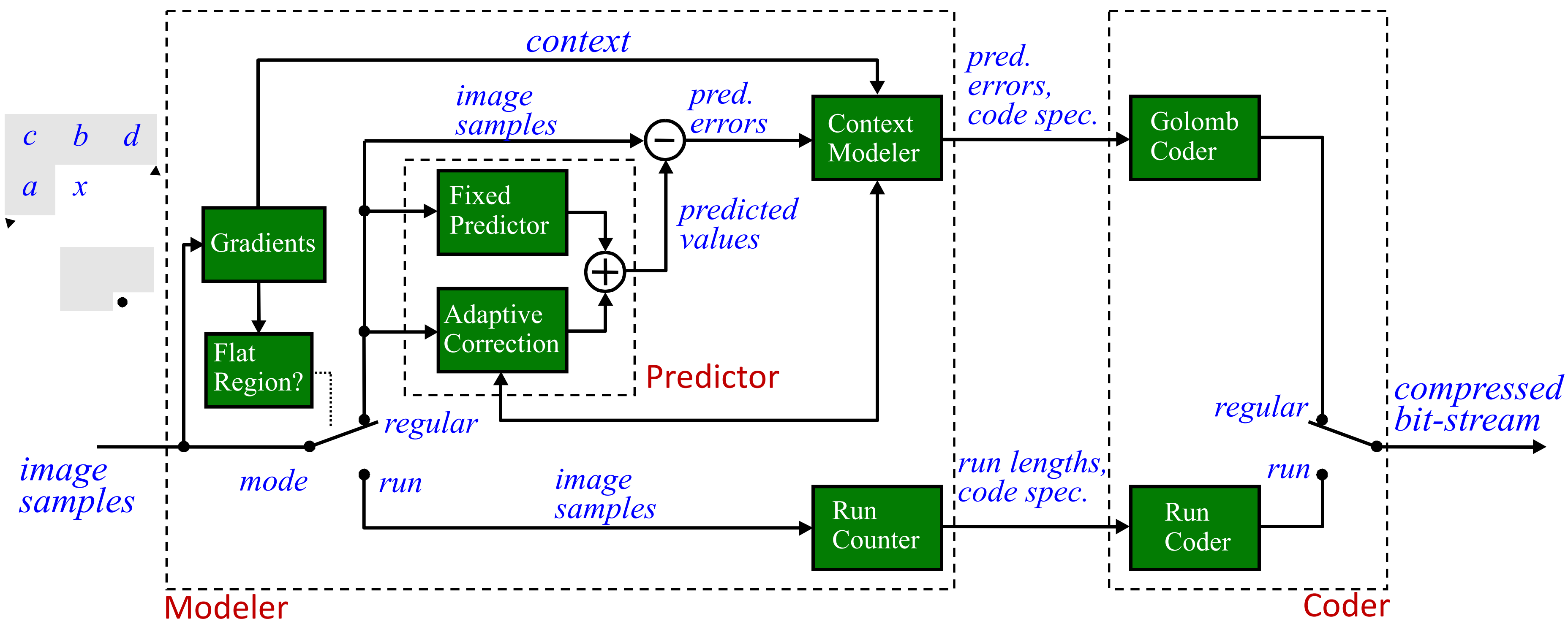
Applications of Information Theory in Image Processing

4. The LOCO-I lossless image compression algorithm and the JPEG-LS standard

The LOCO-I algorithm

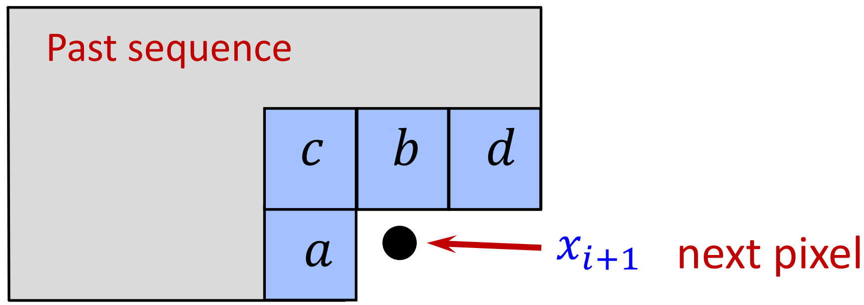
- *JPEG-LS* is based on the *LOCO-I* algorithm:
  LOw COmplexity LOssless COmpression of Images
- Guiding principles
  - Apply basic principles of continuous tone image modeling.
  - Approach state of the art performance at *lowest possible complexity.*
- Basic components:
  - Fixed + Adaptive prediction
  - Conditioning contexts based on quantized gradients
  - Two-parameter conditional probability model (TSGD)
  - Low complexity adaptive coding matched to the model (variants of Golomb codes)
  - Run length coding in flat areas to address drawback of symbol-by-symbol coding
- Reference:
  M. Weinberger, G. Seroussi, G. Sapiro, "The LOCO-I Image compression algorithm: principles and standardization into JPEG-LS", *IEEE Trans. Image Processing*, vol. 9, No. 8, August 2000.

JPEG-LS (LOCO-I Algorithm): Block Diagram



Fixed Predictor: MED

- *Causal template* for prediction and conditioning



Predictor:

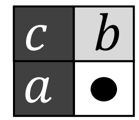
$$\hat{x}_{i+1} = \begin{cases} \min(a,b) & \text{if } c \geq \max(a,b) \\ \max(a,b) & \text{if } c \leq \min(a,b) \\ a+b-c & \text{otherwise} \end{cases}$$

- *Alternative interpretation: median* of three linear predictors (MED)
  - $f_1(a,b,c) = a$
  - $f_2(a,b,c) = b$
  - $f_3(a,b,c) = a+b-c$

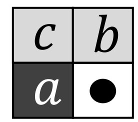
$$f_{\text{med}}(a,b,c) = \text{median}(f_1, f_2, f_3)$$

MED Predictor Properties

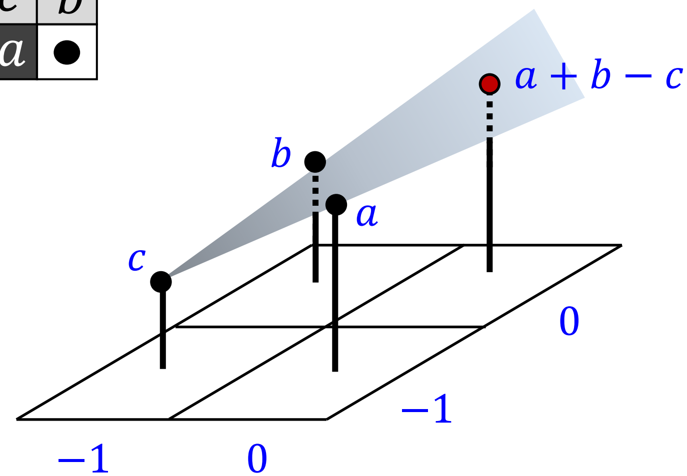
- Nonlinear, has some *edge detection* capability:
  - Predicts b in ‘‘vertical edge’’



  - Predicts a in ‘‘horizontal edge’’



  - Predicts $a + b - c$ in ‘‘smooth region’’



planar extrapolation

Parameter Reduction and Adaptivity

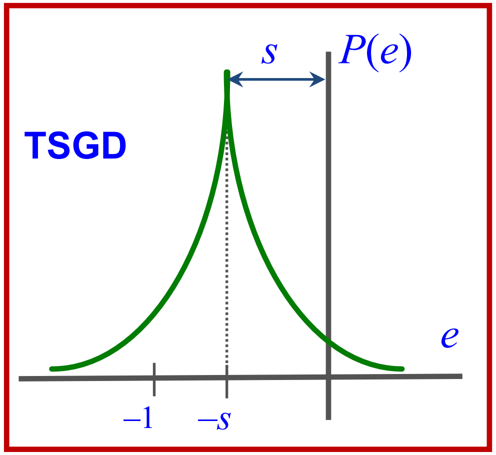
- Statistical model = context set + probability model for prediction residuals
  - each pixel classified into a *context class*
  - prediction error encoded based on a probability distribution conditioned on the context class
- Two factors determine the total number of statistical parameters:
  - number of context classes
  - parametrization of probability distribution for prediction residuals
- Goal: capture high order dependencies without excessive model cost
- Adaptive coding is desired, but arithmetic coding ruled out (if possible…) due to complexity constraints

Parameter Reduction and Adaptivity

- Probabilistic model: Model prediction residuals with a *two-sided geometric distribution (TSGD)*

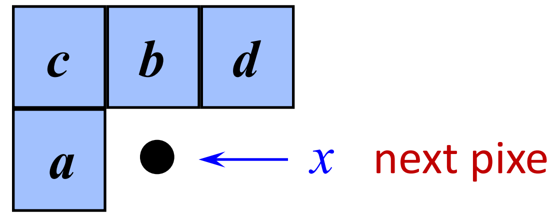
$$P(e) = c_0\theta^{|e+s|}, \qquad \theta \in (0,1), \qquad s \in [0,1), \quad c_0 = \frac{1-\theta}{\theta^s + \theta^{1-s}}$$

  - "discrete Laplacian"
  - only two parameters per context class
    - θ: *"sharpness"* (rate of decay, variance, etc.)
    - s: *"shift"* (often non-zero in a context-dependent scheme)
  - shift s constrained to $[0,1)$ by integer-valued adaptive correction (*bias cancellation*) on a fixed predictor
  - allows for relatively large number of context classes (365 in JPEG-LS)
  - *suited to low complexity adaptive coding*

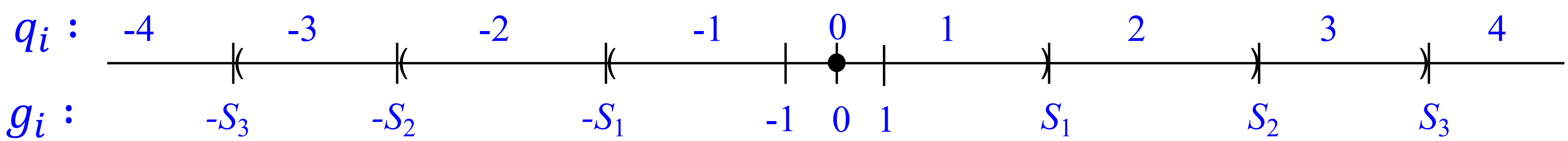


Context Class Determination

Causal template:



- **Gradients** $g_1 = (d - b),\ g_2 = (b - c),\ g_3 = (c - a)$
  - $[g_1, g_2, g_3]$: triplet of *raw gradients*
  - gradients capture the level of activity (smoothness, edginess) surrounding a pixel
  - each of g_1, g_2, g_3 quantized into 9 regions determined by 4 thresholds $1, S_1, S_2, S_3$



- Regions $0, 1, 2, 3, 4$ (resp.): $\{0\},\ \ 1 \le g < S_1,\ \ S_1 \le g < S_2,\ \ S_2 \le g < S_3,\ \ g \ge S_3$
  - Symmetrically on the negative side for regions $-1, -2, -3, -4$
- defaults in JPEG-LS: $S_1 = 3,\ S_2 = 7,\ S_3 = 21$
- $g_i \to q_i \in \{-4, -3, -2, -1,\ \ 0,\ \ 1,\ \ 2,\ \ 3,\ \ 4\}$
- $[g_1, g_2, g_3] \to [q_1, q_2, q_3]$ a triplet of *quantized gradients*

one more step ...

Context Class Determination

- $[g_1, g_2, g_3] \rightarrow [q_1, q_2, q_3]$ a triplet of quantized gradients
- Symmetric contexts (with respect to B/W) merged:

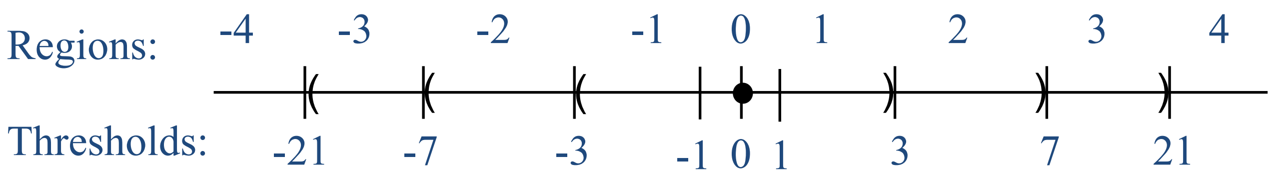
$$P(\, e \mid [q_1, q_2, q_3]\,) \;\leftrightarrow\; P(\, -e \mid [-q_1, -q_2, -q_3]\,)$$

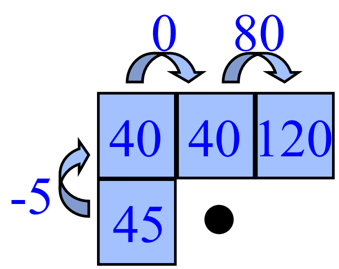
  - Canonical form: change the sign of the vector as needed so that the first nonzero in $[q_1, q_2, q_3]$ (if any) is *positive*. Remember if sign was flipped.
    Result of context class determination:

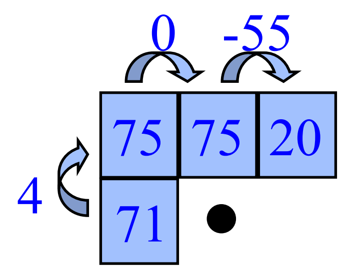
$$[q_1, q_2, q_3, \qquad \mathrm{sign}]$$

  - How is this used? Say the prediction error is $e = x_{i+1} - \hat{x}_{i+1}$.
    - If $\mathrm{sign} < 0$, add a count of $-e$ to the stats, else add a count of e. Encode the value that was added to the stats.
    - On the decoder size, after decoding e, add it to the stats. If $\mathrm{sign} < 0$, flip the sign of e, then reconstruct $x_{i+1} = \hat{x}_{i+1} + e$.
- Fixed number of contexts: $(9^3 + 1)/2 = 365$
  - addresses the problem of context dilution

Context Class Determination -- Examples







$[g_1, g_2, g_3] = [\,80, 0, -5]$

$[q_1, q_2, q_3] = [\,4, 0, -2]$

$[g_1, g_2, g_3] = [\,-55, 0, 4]$

$[q_1, q_2, q_3] = [\,-4, 0, 2]$

same context class (keeping track of sign)

Encoding of TSGDs: Golomb codes

- Optimal prefix codes for TSGDs are built out of the *Golomb codes*.
  - Family of prefix-free codes for *geometric distributions* over nonnegative integers described by Golomb in 1966 (motivated by sequences of Bernoulli trials)
- Consider an integer $m \geq 1$. The mth order *Golomb code* G_m encodes an integer $j \geq 0$ as follows:

$$G_m(j) = \mathrm{binary}_m(j \bmod m) \cdot \mathrm{unary}(j \operatorname{div} m)$$

$j \bmod m,\ j \operatorname{div} m =$ remainder and quotient in integer division j/m (resp.)

$\mathrm{binary}_m(i) =$ (shortest) binary representation of i, $\lfloor \log m \rfloor$ or $\lceil \log m \rceil$ bits

$\mathrm{unary}(i) = \overbrace{00\ldots0}^{i}1$ unary representation of i.

- Given m and $G_m(j)$, a decoder uniquely reconstructs j.

Golomb codes -- Examples

$m = 5$

i	$G_m(i)$	$\ell(i)$
0	00 1	3
1	01 1	3
2	10 1	3
3	110 1	4
4	111 1	4
5	00 01	4
6	01 01	4
7	10 01	4
8	110 01	5
9	111 01	5
10	00 001	5
11	01 001	5
12	10 001	5
13	110 001	6
14	111 001	6
⋮	⋮	⋮

3

5

5

⋮

$m = 2^k = 4, \quad k = 2$

i	i (binary)	$G_m(i)$	$\ell(i)$
0	00	00 1	3
1	01	01 1	3
2	10	10 1	3
3	11	11 1	3
4	1 00	00 01	4
5	1 01	01 01	4
6	1 10	10 01	4
7	1 11	11 01	4
8	10 00	00 001	5
9	10 01	01 001	5
10	10 10	10 001	5
11	10 11	11 001	5
12	11 00	00 0001	6
13	11 01	01 0001	6
14	11 10	10 0001	6
⋮		⋮	⋮

4

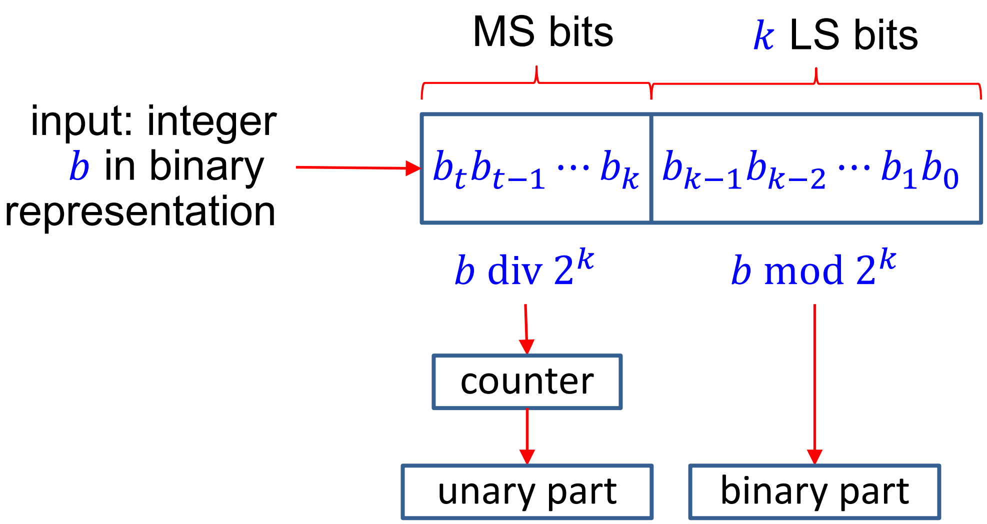
4

4

⋮

Golomb PO2 codes

- When $m = 2^k$, we call G_m a *Golomb power of two* (PO2) code and use k as the defining parameter: $G_k^* \triangleq G_{2^k}$.
- PO2 codes are especially simple to implement!
  Example: *Golomb PO2 encoder*



C/C++:

$b \bmod 2^k$: `b & ((1<<k)-1)`

$b \operatorname{div} 2^k$: `b >> k`

Optimality of Golomb codes for GDs

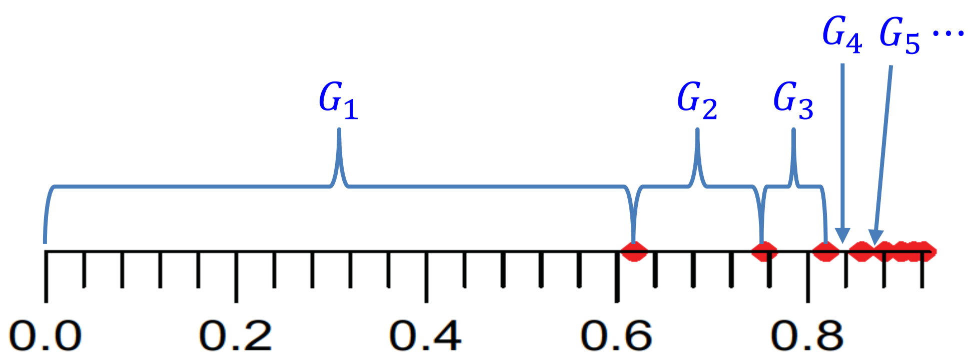
Theorem [Gallager, Van Voorhis 1975] Let $P_\theta(i) = (1-\theta)\theta^i$ be a geometric distribution on the nonnegative integers, with $0 < \theta < 1$, and let m be the unique (positive) integer satisfying

$$\theta^m + \theta^{m+1} \le 1 < \theta^m + \theta^{m-1}.$$

Then, the Golomb code G_m is optimal for P_θ.

Solution of $\theta^m + \theta^{m+1} = 1$

m	θ_m
1	0.6180339887
2	0.7548776662
3	0.8191725134
4	0.8566748839
5	0.8812714616
6	0.8986537126
7	0.9115923535
8	0.9215993196

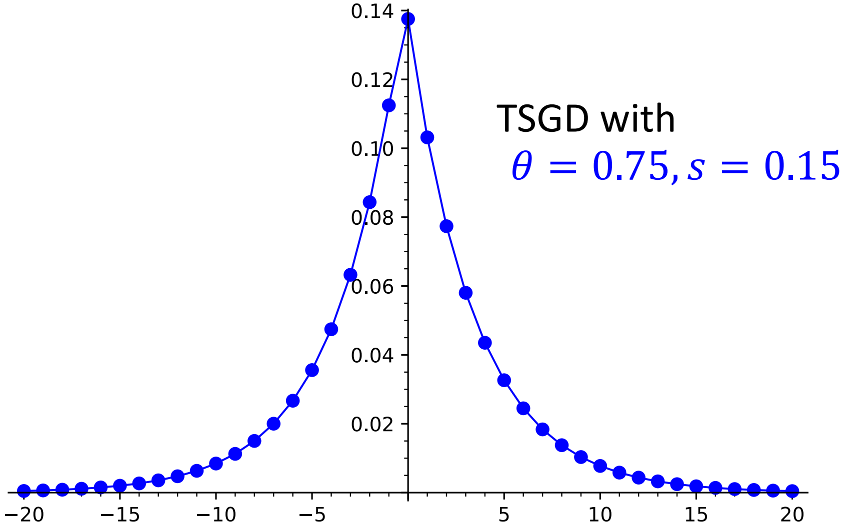


Golomb (1966) had proved optimality for $\theta = 2^{-\frac{1}{m}}$, i.e. $\theta^m = \frac{1}{2}$.

How about TSGDs?

$$P_{\theta,s}(e) = c_0\theta^{|e+s|}, \qquad 0 < \theta < 1, \qquad 0 \leq s < 1.$$

- This had been traditionally studied for $s = 0$.
  People used one of two approaches to encode e :
  1. Encode $|e|$ with the optimal G_m for θ and send a sign bit if $e \neq 0$ (symmetric).



How about TSGDs?

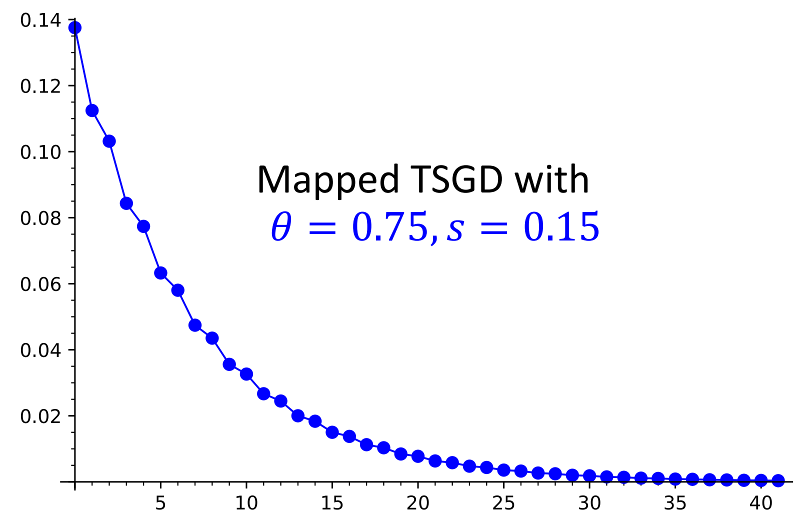
$$P_{\theta,s}(e) = c_0\theta^{|e+s|}, \qquad 0 < \theta < 1, \qquad 0 \le s < 1.$$

- This had been traditionally studied for $s = 0$.
  People used one of two approaches to encode e :
  1. Encode $|e|$ with the optimal G_m for θ and send a sign bit if $e \neq 0$ (symmetric).
  2. Define $M(e) \overset{\text{def}}{=} \begin{cases} 2e & e \geq 0 \\ 2|e|-1 & e < 0 \end{cases}$

e	$M(e)$
0	0
-1	1
1	2
-2	3
2	4
$\vdots$	$\vdots$

Rice mapping

Encode e with $G_m(M(e))$ (asymmetric).



Mapped TSGD with $\theta = 0.75, s = 0.15$

How about TSGDs?

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

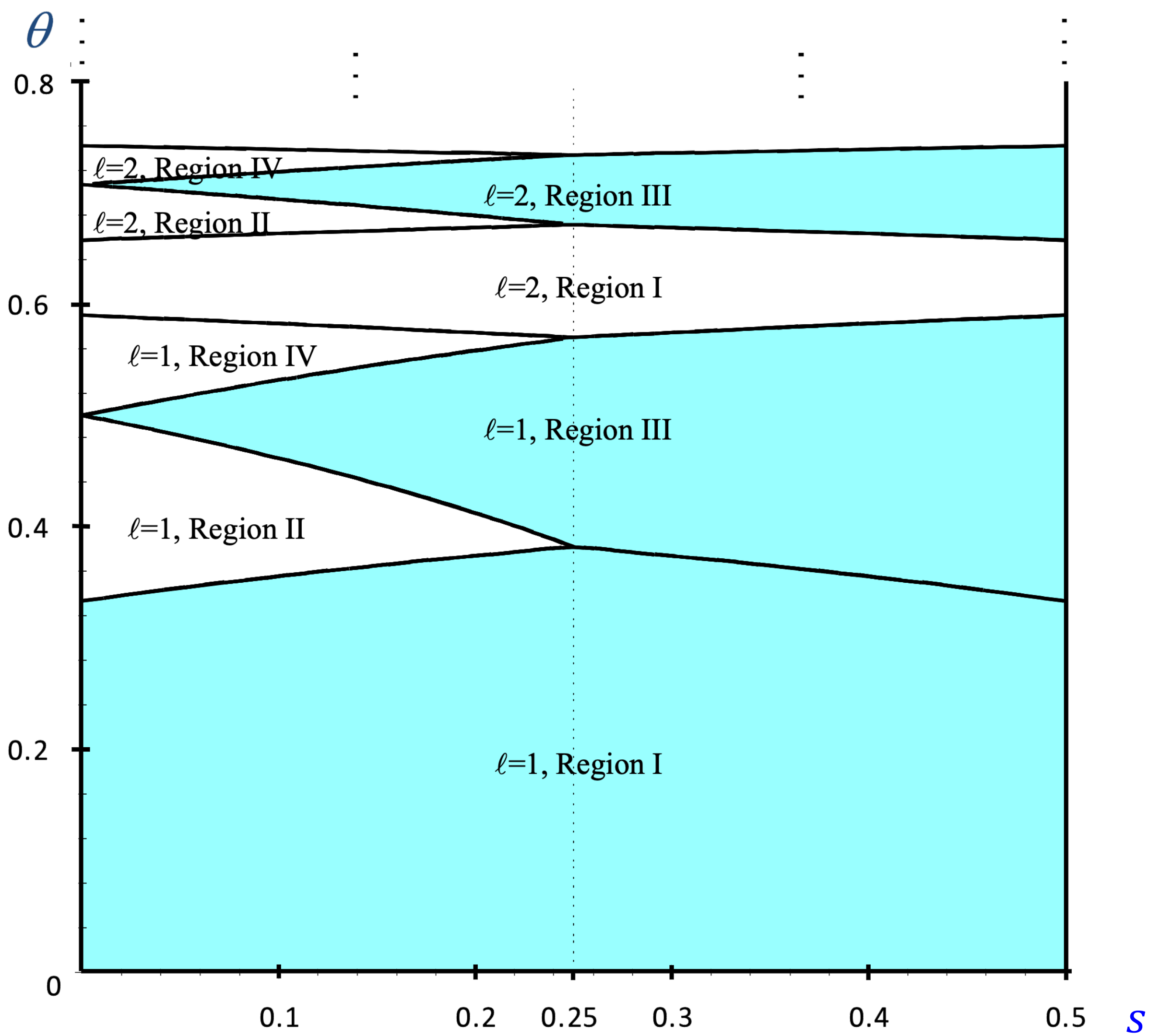
- However, the characterization of *optimal codes* for these distributions (analogous to Golomb codes for one-sided geometric distributions) was an *open question*, even for the simpler case $s = 0$.

- Optimal codes for TSGDs (with $s \in [0,1)$) were first characterized in

  N. Merhav, G. Seroussi and M.J. Weinberger, "Optimal prefix codes for sources with two-sided geometric distributions," *IEEE Trans. on Information Theory*, 46,, 2000, pp. 121–135.

Optimal Code Regions for TSG Distributions

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



Regions characterized by an integer parameter ℓ and a label I, II, III, or IV.

For each region, an optimal code is characterized.

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

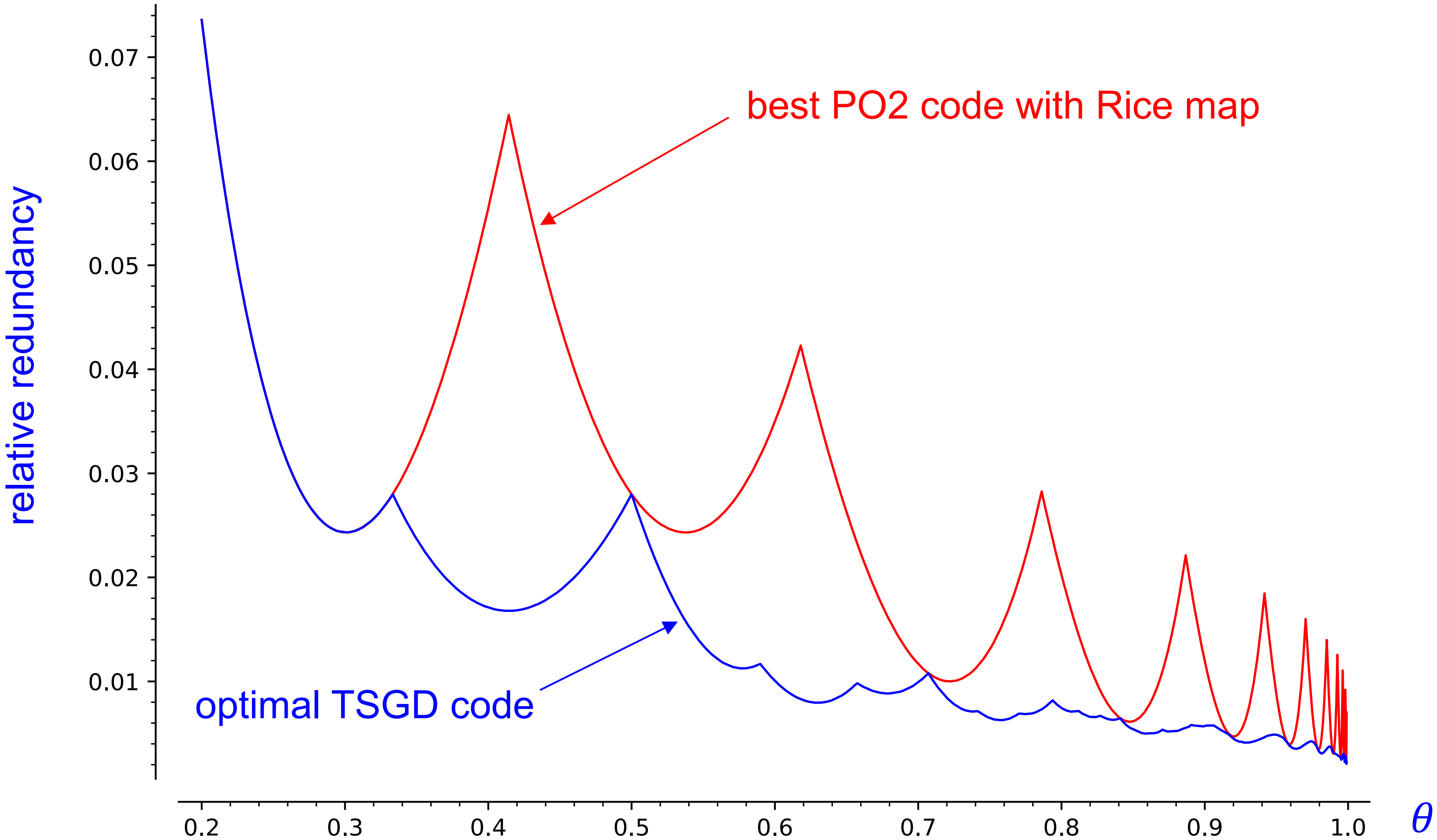
Region III: $G_{2\ell}(M(x))$

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

Golomb-PO2 code

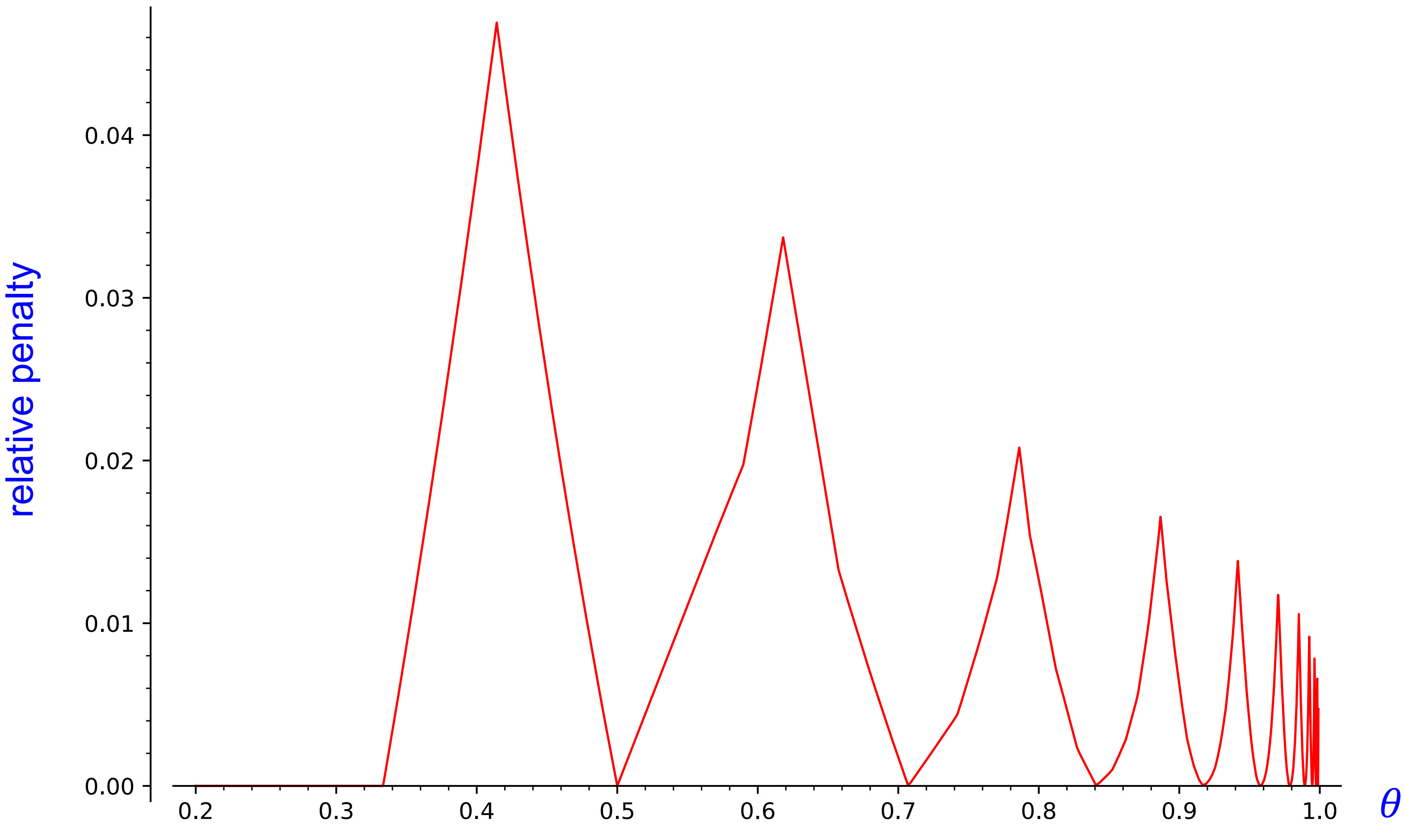
Relative redundancy of codes for TSGD

relative redundancy measured as $\frac{codelength - entropy}{entropy}$, for TSGD with $s = 0$



Penalty for use of PO2 codes vs. optimal for TSGD

relative penalty measured as $\frac{codelength(PO2)-codelength(optimal)}{codelength(optimal)}$, for TSGD with $s=0$



LOCO-I components

c	b	d
a	x	

loop:

- Get context pixels $a, b, c, d,$ next pixel x
- Compute gradients $d-b$, $b-c$, $c-a$ and quantize $\Rightarrow [q_1, q_2, q_3,\ \mathrm{sign}]$
- $[q_1, q_2, q_3] = 0$? YES: Go to **run state** NO: **proceed**
- $x_{\mathrm{pred}} = \mathrm{predict}(a, b, c)$
- Retrieve bias correction value for context, adjust sign if needed. Correct x_{pred}
- $\epsilon = x - x_{\mathrm{pred}}$. If $\mathrm{sign} < 0$ then $\epsilon = -\epsilon$. Map $\epsilon \mod \alpha$ to range $[-\frac{\alpha}{2}, \frac{\alpha}{2})$
- Estimate Golomb PO2 parameter k for the context
- Update stats for coding and bias correction
- Remap $\epsilon \to M(\epsilon)$ or $\epsilon \to M(-1-\epsilon)$
- Encode M with Golomb-PO2(k)
- Back to loop

run state:

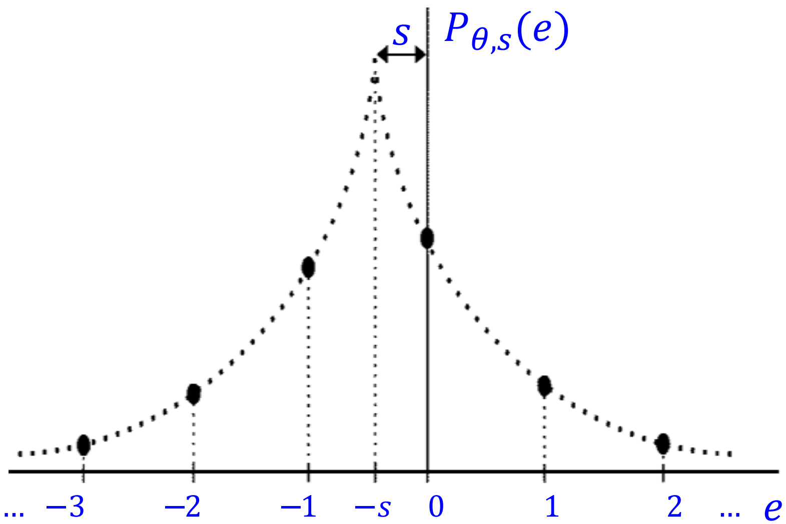
- Count run of a until $x \neq a \Rightarrow$ run length ℓ
- Encode ℓ using block-MELCODE
- Update MELCODE state
- Back to regular loop

Adaptive Coding of TSGD's in JPEG-LS

- Optimal prefix codes for TSGD's are approximated in JPEG-LS by applying the Golomb-PO2 subfamily to a mapped error value
  - two mappings are used in JPEG-LS

    $\epsilon \to M(\epsilon)$, $\quad 0, -1, +1, -2, +2, \ldots \to 0, 1, 2, 3, 4, \ldots$ (most cases)

    $\epsilon \to M(-1-\epsilon)$, $\quad -1, 0, -2, +1, -3, \ldots \to 0, 1, 2, 3, 4, \ldots$ (only with $k = 0$, $s > \frac{1}{2}$)



- Assumption $s \in [0,1)$ satisfied through the use of *adaptive correction* of the predictor, using, per context:

  B = accumulated sum of error values

  N = total number of samples
- For *adaptive coding*, we use (p/context):

  A = accumulated sum of error magnitudes

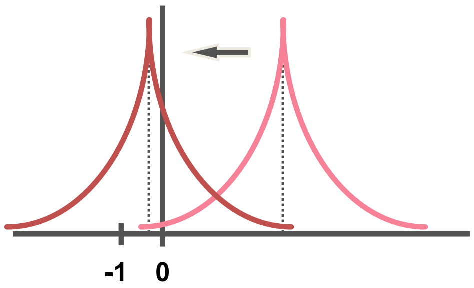
  $N\_$ = number of negative samples

Bias Correction

- In principle, $\mu = \lfloor B/N \rfloor$ gives the integer part of the bias
  - but we don't want to use division! (low complexity constraint)
  - instead, we implement the following procedure, which computes a correction value C for the fixed predictor $\hat{x}_{\mathrm{med}}$. Starting with $N = 1, B = C = 0$, for each sample, with prediction error ϵ, we do

```
B = B + ε;      /* accumulate prediction residual */
N = N + 1;      /* update occurrence counter */
/* update correction value and shift statistics */
if ( B ≤ -N ) {
    C = C - 1; B = B + N;
    if ( B ≤ -N ) B = -N + 1;
}
else if ( B > 0 ) {
    C = C + 1; B = B - N;
    if ( B > 0 ) B = 0;
}
```

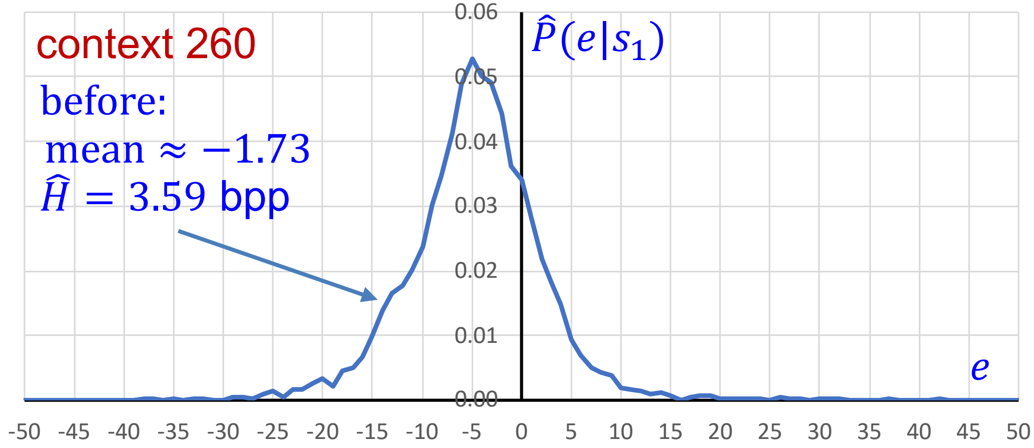
this is the correction computation for the *next* pixel, done *after* ϵ has been encoded

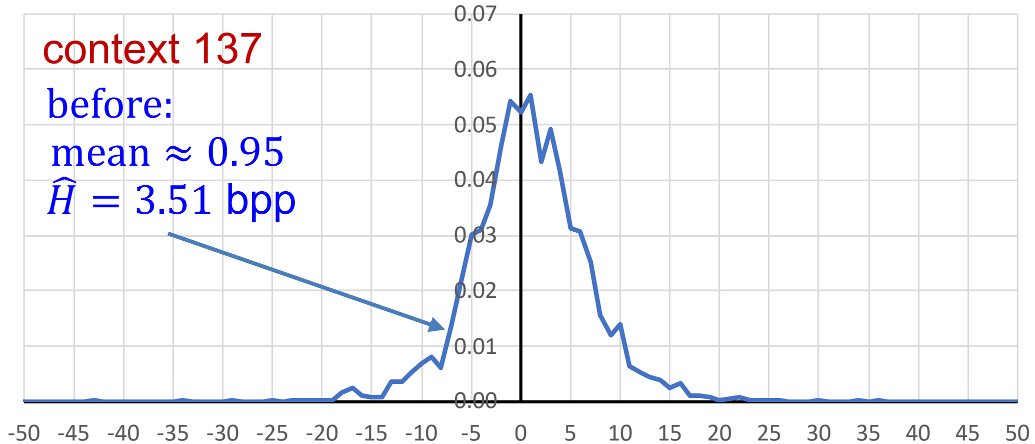


  - only additions/subtractions
  - B is kept in the range $-N < B \leq 0$
  - C tracks $\lfloor B/N \rfloor$; full predictor is $\hat{x} = \hat{x}_{\mathrm{med}} + C \cdot \mathrm{sign}$ (clipped)

Effect of bias correction

Grayscale image 720x576

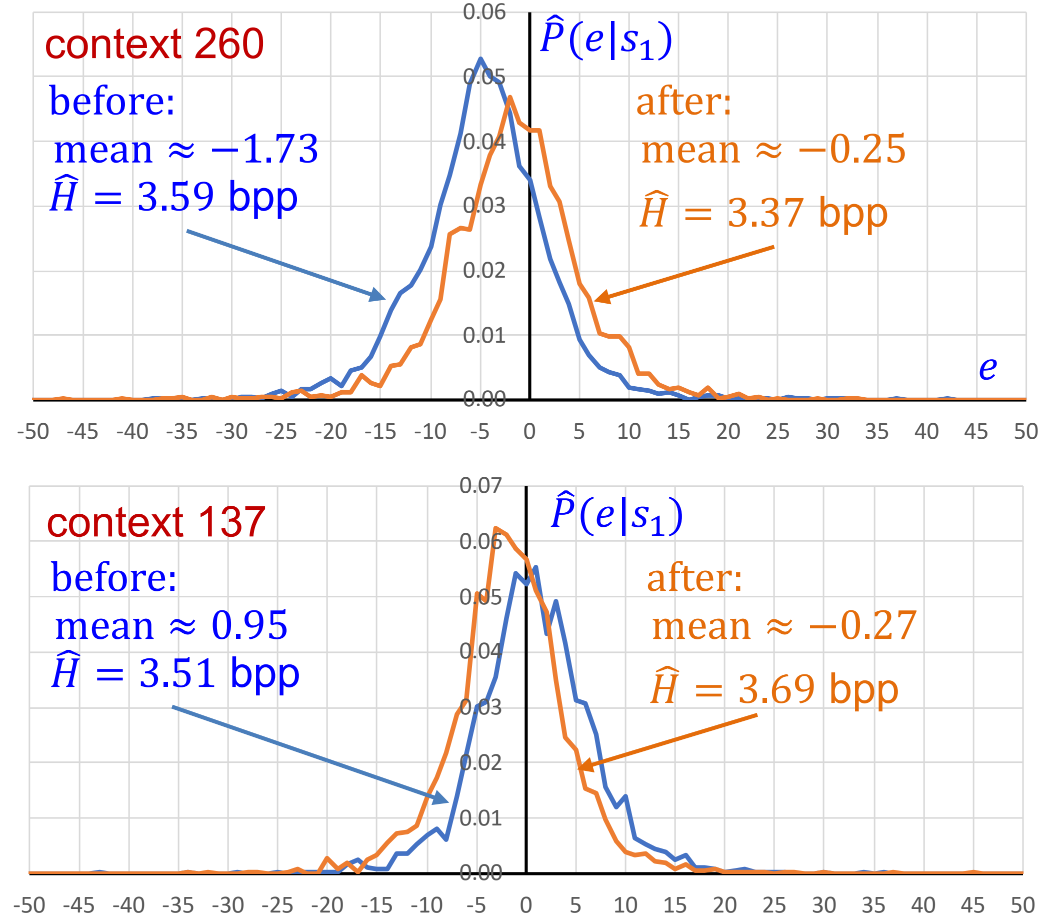




Distributions of prediction errors before and after bias correction

Effect of bias correction

Grayscale image 720x576



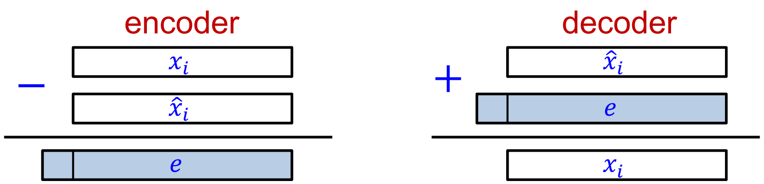
Distributions of prediction errors before and after bias correction

Prediction Errors Alphabet

- In principle, a prediction error $e = x_i - \hat{x}_i$ can assume values in the range $-\alpha + 1 \le e \le \alpha - 1 \Rightarrow$ alphabet for e is larger (1 bit longer)?
  - However, given $\hat{x}_i$, e can assume only α different values, $-\hat{x}_i \le e \le \alpha - 1 - \hat{x}_i \Rightarrow$ a larger alphabet should not be needed
  - Indeed, if we carry out all operations modulo α, reconstruction will be correct

$$e = x_i - \hat{x}_i \bmod \alpha \quad \text{(encoder side)}$$

$$x_i = \hat{x}_i + e \bmod \alpha \quad \text{(decoder side)}$$



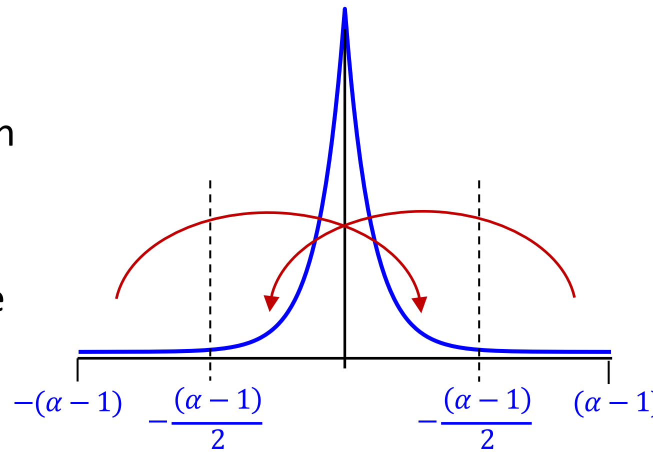
ignoring the extra (leftmost) bit cannot affect the result !

$\Rightarrow$ map the prediction error modulo α to a range of size α

Prediction Errors Alphabet

- Map the prediction error to a range of size α. What range?
  - Take residues in the range $-\left\lceil\frac{\alpha-1}{2}\right\rceil \le e \le \left\lfloor\frac{\alpha-1}{2}\right\rfloor$ to preserve TSGD assumption
  - the algorithm: if $e < -\left\lceil\frac{\alpha-1}{2}\right\rceil$: $\quad e = e + \alpha$

    else if $e > \left\lfloor\frac{\alpha-1}{2}\right\rfloor$: $\quad e = e - \alpha$
    - "folds" large prediction errors into small values, can help in edge regions; overall effect on compression is not large
    - practical advantage: all numbers (and registers) are of the same length
  - On decoder side, reduce $(\hat{x} + e) \bmod \alpha$ to range $0 \le x < \alpha$.
  - Example: $\alpha = 256,\ -128 \le e \le 127$



<u>Encoder</u>

$$\hat{x}_i = 240$$
$$x_i = 1$$
$$e = 1 - 240 = -239 \xrightarrow{+256} 17$$

<u>Decoder</u>

$$\hat{x}_i = 240$$
$$e = 17$$
$$x_i = 240 + 17 = 257 \xrightarrow{\text{mod } 256} 1$$

Adaptive Coding of TSGD's in JPEG-LS (cont.)

$$P_{\theta,s}(e) = c_0\theta^{|e+s|}, \qquad 0 < \theta < 1, \qquad 0 \le s < 1, \qquad c_0 = \frac{1-\theta}{\theta^{1-s}+\theta^{s}}$$

- Given a sequence of prediction errors $e_1^t = e_1 e_2 e_3 \ldots e_t$, let

  $A_t \; = \sum_i |e_i| =$ accumulated sum of error magnitudes

  $N_t^- \; = \sum_i 1_{e_i<0} =$ number of negative samples

  Then,

$$P_{\theta,s}(e_1^t) = (c_0)^t \prod_{e_i\ge 0} \theta^{|e_i|+s} \prod_{e_i<0} \theta^{|e_i|-s}$$

$$= (c_0)^t \prod_i \theta^{|e_i|} \prod_{e_i\ge 0} \theta^{s} \prod_{e_i<0} \theta^{-s} \;\; = (c_0)^t \theta^{A_t} \theta^{(t-N_t^-)s} \theta^{-N_t^- s}$$

$$= (c_0)^t \theta^{A_t + (t-2N_t^-)s}$$

$\Rightarrow$ A_t, N_t^- are *sufficient statistics* for θ, s (every sequence with the same values of t, A_t, N_t^- has the same probability)

Adaptive Coding of TSGD's in JPEG-LS (cont.)

$$\boxed{P_{\theta,s}(e) = c_0\theta^{|e+s|}, \qquad 0 < \theta < 1, \qquad 0 \le s < 1, \qquad c_0 = \frac{1-\theta}{\theta^{1-s}+\theta^s}}$$

$A_t = \sum_i |e_i|$ = accumulated sum of error magnitudes
$N_t^- = \sum_i 1_{e_i<0}$ = number of negative samples

- Define

$$S = \frac{\theta}{1-\theta}, \quad \rho = \frac{\theta^{1-s}}{\theta^{1-s}+\theta^s}$$

easy to show that
$\rho = P_{\theta,s}(e<0)$

$$\boxed{P_{\theta,\rho}(e) = \begin{cases} (1-\rho)(1-\theta)\theta^e, & e \ge 0, \\ \rho(1-\theta)\theta^{-1-e}, & e < 0\,. \end{cases}}$$

  - We'll use the parameter pair (S,ρ) instead of (θ,s) (it's clear that the pairs are in 1-1 correspondence).

- The ML estimators of S,ρ are given by

$$\hat{S}_t = (A_t - N_t^-)\,/t$$
$$\hat{\rho}_t = N_t^-/t$$

  - Also, the ML estimator for θ is $\hat{\theta}_t = \hat{S}_t/(\hat{S}_t+1)$ (much harder to get a "nice" one for s, that's why we reparametrized!)

$$\boxed{\hat{\theta}_t = \frac{A_t - N_t^-}{A_t - N_t^- + t}}$$

Our adaptation strategy will approximate one where $\hat{S}_t, \hat{\rho}_t$ are computed and the corresponding best code from the sub-family is selected.

Adaptive coding: choosing k

- For the Golomb code with parameter 2^k, $k > 0$, and mapping $M(\cdot)$:

  $$0, -1, 1, -2, 2, -3\ \dots\ \rightarrow\ 0, 1, 2, 3, 4, 5, \dots$$

  the average code length under $P_{\theta,s}(e)$ is

  $$L_k = k + 1 + \frac{\theta^{2^{k-1}}}{1 - \theta^{2^{k-1}}} \triangleq k + 1 + \frac{z}{1 - z} \qquad \boxed{z \triangleq \theta^{2^{k-1}}}$$

  Independent of s or ρ! Why?

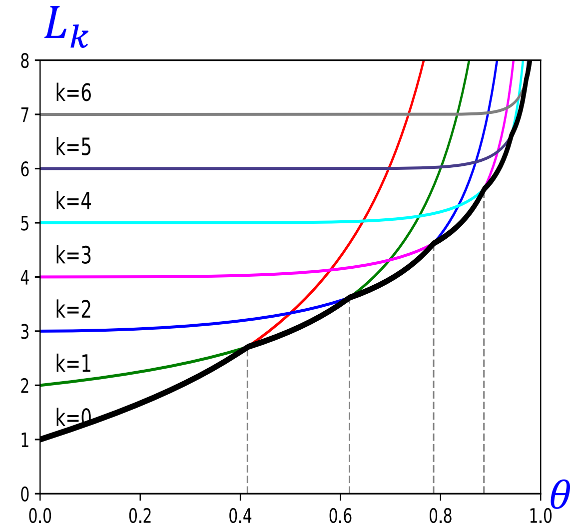
- The code G_{2^k}, $k \geq 1$, applied to integers mapped with $M(\cdot)$, assigns the same code length to integers in pairs $i, -(i+1)$.
  Example: $k = 1$

  $$\underbrace{0, -1}_{2}, \underbrace{1, -2}_{3}, \underbrace{2, -3}_{4}, \underbrace{3, -4}_{5}, \underbrace{4, -5}_{6}, 5, \dots$$

  length 2 3 4 5 6 ...

- On the other hand,
  $P(i) + P(-(i+1)) = \frac{1-\theta}{\theta^s + \theta^{1-s}}\left(\theta^{i+s} + \theta^{i+1-s}\right) = (1-\theta)\theta^i$ independent of s.

Adaptive coding: choosing k



- Transition points for $k > 0$:
  - $L_k = k + 1 + \frac{z}{1-z}, \quad z \triangleq \theta^{2^{k-1}}$
  - $L_k = L_{k+1} \Leftrightarrow k + 1 + \frac{z}{1-z} = k + 2 + \frac{z^2}{1-z^2}$

    $\Leftrightarrow \; z^2 + z - 1 = 0$

    positive root is at $z = \phi = \frac{1}{2}(\sqrt{5} - 1) \approx 0.618$

    $\Rightarrow$ transition points at $\theta^{2^{k-1}} = \phi \; \Rightarrow \; \theta = \phi^{2^{-k+1}}$
  - observe: $\theta^{-1} = \phi^{-2^{-k+1}} = e^{-2^{-k+1}\ln\phi} \approx 1 - 2^{-k+1}\ln\phi$,

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

    $-\ln\phi \approx 0.48 \approx \frac{1}{2}$

    $S = \frac{1}{\theta^{-1}-1} \approx \frac{1}{-2^{-k+1}\ln\phi} \approx 2^k$ power of 2!

    code transitions occur when S is near a power of 2
  - in fact, this is true even for small k if we take $S + 1/2$

k	θ	$S + 1/2$
1	0.6180	2.1
2	0.7862	4.2
3	0.8867	8.3
4	0.9416	16.6
5	0.9704	33.3
6	0.9851	66.5
7	0.9925	133.0
8	0.9962	266.0
9	0.9981	532.0
10	0.9991	1064.0
11	0.9995	2128.0

Adaptive coding: choosing k (cont.)

- The case $k = 0$. Here L_0 depends on s:

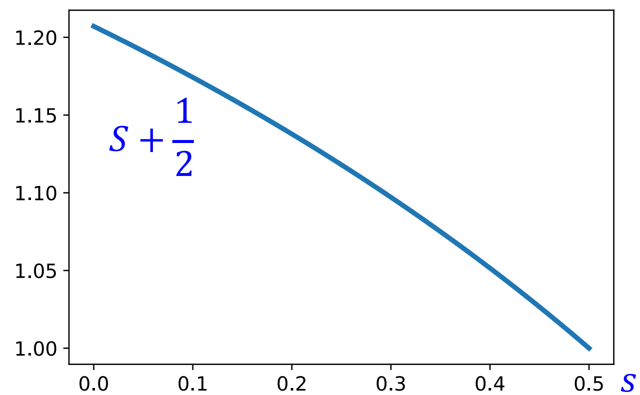
$$L_0 = \frac{2}{1-\theta} - \frac{\theta^s}{\theta^s+\theta^{1-s}}$$

- Transition $k = 0 \to k = 1$ with $s = 0$: $L_0 = \frac{1+3\theta}{(1-\theta)^2}, \quad L_1 = 2 + \frac{\theta}{1-\theta}$

$$L_0 = L_1 \;\Rightarrow\; \theta = \sqrt{2} - 1 \;\Rightarrow\; S = \frac{\theta}{1-\theta} = \frac{\sqrt{2}}{2} \approx 0.7 \Rightarrow S + \frac{1}{2} \approx 1.2$$

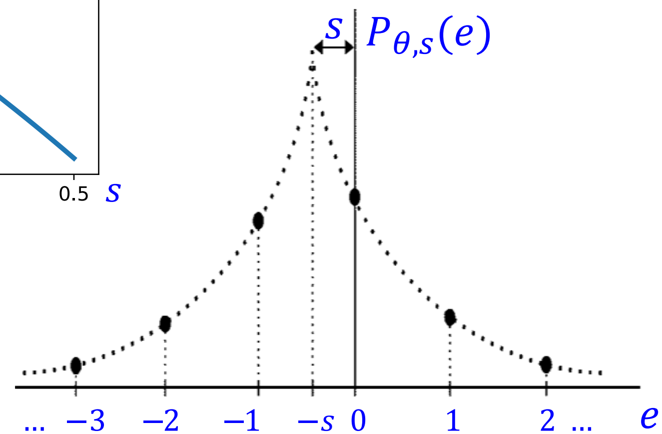
again close to power of 2

The approximation is even better for $s > 0$ (solved numerically)



- If $s > \frac{1}{2}$, then $P_{\theta,s}(-1) > P_{\theta,s}(0)$, and all codelengths are different. We want to give the shortest code to $-1 \Rightarrow$ use mapping $M(-e-1)$:

$$-1, 0, -2, 1, -3, 2, \ldots \to 0, 1, 2, 3, 4, 5, \ldots$$



recall $\rho = \frac{\theta^{1-s}}{\theta^s+\theta^{1-s}}$, so $s > \frac{1}{2} \Leftrightarrow \rho > \frac{1}{2}$

Adaptive coding: the solution

$\hat{S}_t = (A_t - N_t^-)/t \Rightarrow A_t = t\hat{S}_t + N_t^- \approx t\hat{S}_t + t/2 = t(\hat{S}_t + 1/2)$

$\Rightarrow$ transition points near $A_t/t =$ power of 2

k	$\boldsymbol{\theta}$	$S + 1/2$
0	0.4142	1.2
1	0.6180	2.1
2	0.7862	4.2
3	0.8867	8.3
4	0.9416	16.6
5	0.9704	33.3
6	0.9851	66.5
7	0.9925	133.0
8	0.9962	266.0
9	0.9981	532.0
10	0.9991	1064.0
11	0.9995	2128.0

Summary of code selection in JPEG-LS

- Using context statistics N, A, N^-, estimate

$$k \cong \left\lceil \log_2 \frac{A}{N} \right\rceil$$

or, simply,

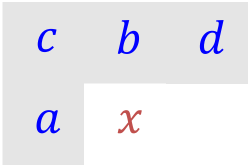
```
for ( k=0; (N<<k)< A; k++ );
```

- If $k = 0$, use N^-/N to estimate ρ and determine if $s > \frac{1}{2}$, then select a mapping:

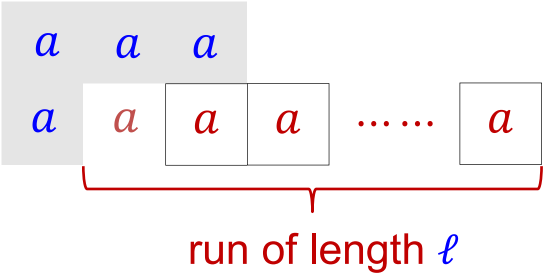
$$\begin{cases} M(e), & \text{if } 2N^- \leq N \\ M(-e-1), & \text{otherwise } (\hat{\rho} > \frac{1}{2}, s > \frac{1}{2}) \end{cases}$$

Embedded Run-length Coding

- Aimed at overcoming the basic limitation of 1 bit/pixel inherent to pixel-wise prefix codes, most damaging in flat, low-entropy regions
- Creates super-symbols representing runs of the same pixel value in the "flat region" $a = b = c = d \Rightarrow$ special context $[q_1, q_2, q_3] = [0,0,0]$.



What we're betting on:



- A run of a is counted and the count ℓ (which could be 0) is encoded using *block-MELCODE*, a variation of Golomb codes with fast adaptation.
  - Decoder sees the same special context and goes into "run mode" without need for additional signaling.
  - Run samples following the first a need not be in the special context.

LOCO-I in one page

c	b	d
a	x	

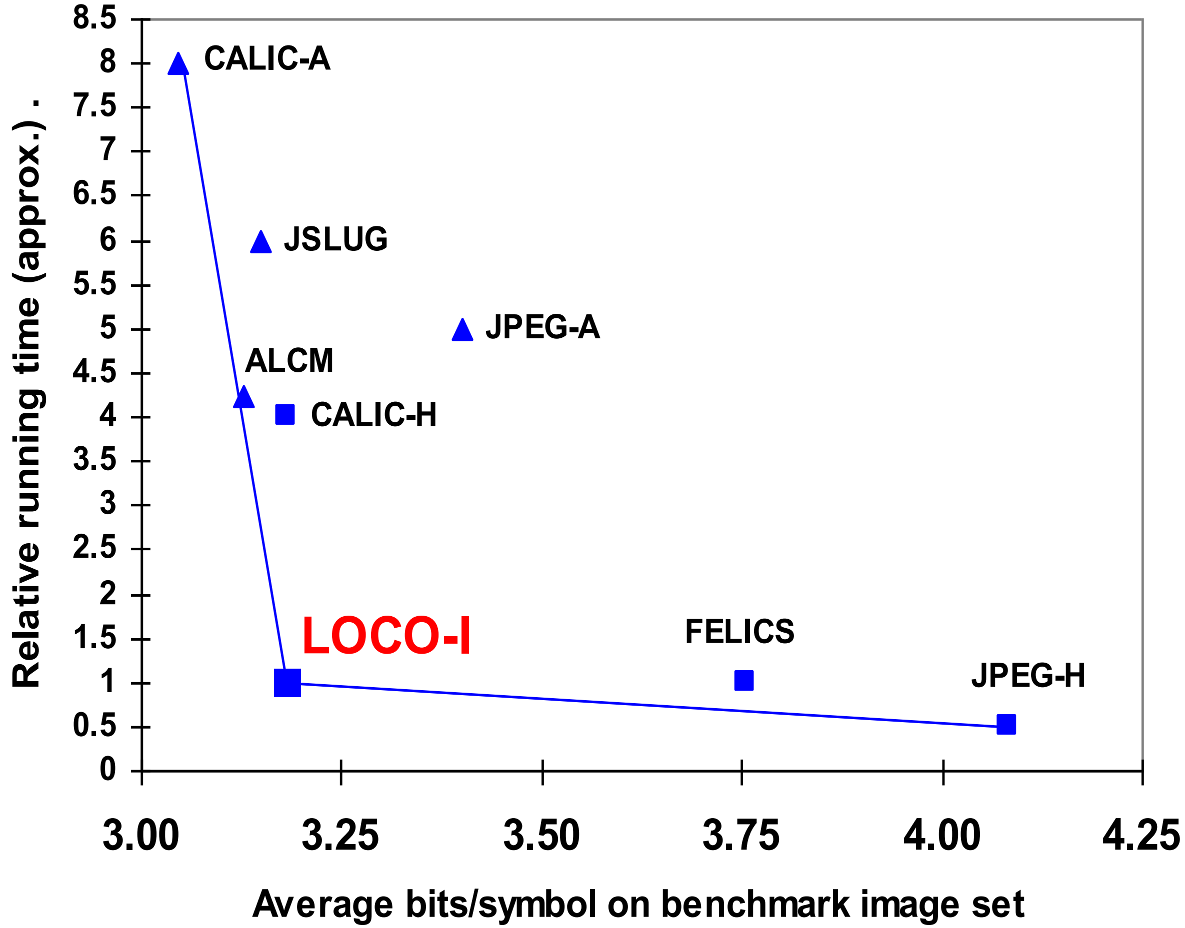
loop:

- Get context pixels $a, b, c, d,$ next pixel x
- Compute gradients $d-b,\ b-c,\ c-a$ and quantize $\Rightarrow [q_1, q_2, q_3,\ \text{sign}]$
- $[q_1, q_2, q_3] = 0$? YES: Go to **run state** NO: **proceed**
- $x_{\text{pred}} = \text{predict}(a, b, c)$
- Retrieve bias correction value for context, adjust sign if needed. Correct x_{pred}
- $\epsilon = x - x_{\text{pred}}$. If $\text{sign} < 0$ then $\epsilon = -\epsilon$. Map $\epsilon \bmod \alpha$ to range $[-\frac{\alpha}{2}, \frac{\alpha}{2})$
- Estimate Golomb PO2 parameter k for the context
- Update stats for coding and bias correction
- Remap $\epsilon \to M(\epsilon)$ or $\epsilon \to M(-1-\epsilon)$
- Encode M with Golomb-PO2(k)
- Back to loop

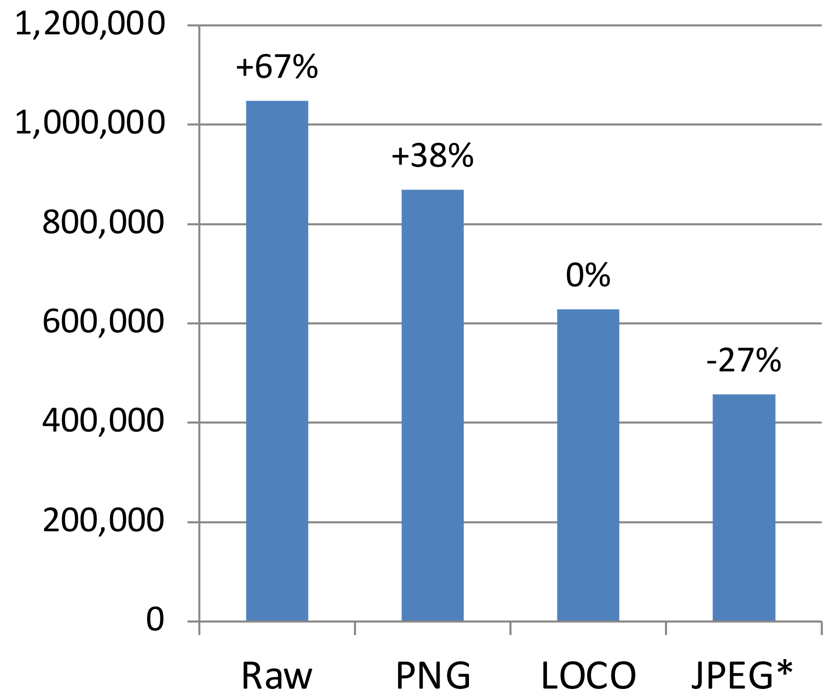
run state:

- Count run of a until $x \neq a \Rightarrow$ run length ℓ
- Encode ℓ using block-MELCODE
- Update MELCODE state
- Back to regular loop

Compression/Complexity trade-off



Mars Image Compressed on NASA "Curiosity" Rover with LOCO-I (August 8, 2012)



Lossy, as provided in NASA's web.

Near-Lossless compression

- Near-lossless compression: reconstructed sample differs from original by up to a preset (small) magnitude δ
  - Traditional DPCM/quantization loop, with prediction error quantized into bins of size $2\delta + 1$

$$\epsilon \to Q(\epsilon) = \left\lfloor \frac{\epsilon+\delta}{2\delta+1} \right\rfloor, \quad \epsilon \geq 0 \quad \text{(symmetric for } \epsilon < 0\text{)}$$

$$Q(\epsilon) \to \epsilon' = (2\delta + 1)Q(\epsilon) \quad \text{Reconstruction}$$

  - Lossless $\Leftrightarrow \delta = 0$
  - Run mode test relaxed to $|c - a|, |b - c|, |d - b| \leq \delta$ (causal template built of reconstructed samples)
  - Often outperforms lossy JPEG in the low-distortion region of the R-D curve

More Comparisons

- Lossless compression on JPEG-LS benchmark set (8 bps)
  - rich set including natural and aerial photographs, compound documents, scanned, medical and computer-generated images

	JPEG-LS	Lossless JPEG (H)	Lossless JPEG (A)	FELICS	PNG	CALIC	LOCO-A
Avg. CR (bps)	3.19	4.08	3.40	3.76	3.46	3.06	3.06
Δ/JPEG-LS	0%	+28%	+7%	+18%	+8%	-4%	-4%

extension of JPEG-LS with arithmetic coding

- Near-lossless: JPEG-LS outperforms JPEG at high bit-rates

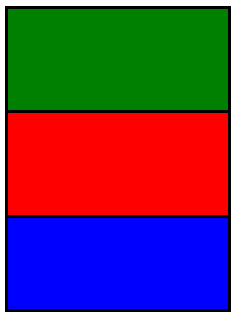
	JPEG-LS RMSE	JPEG RMSE
$\delta = 1$	0.82	1.50
$\delta = 3$	1.93	2.30

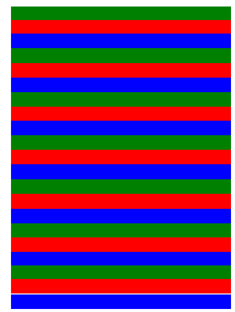
Typical RMSE at similar bit-rate, on original JPEG benchmark images

  - JPEG-LS also outperforms JPEG2000 at $\delta \leq 1$ (but not at $\delta > 2$)

JPEG-LS Features: Color Images

- Color images: 3 basic modes for color planes


non-interleaved


line-interleaved


sample-interleaved

- Statistics are shared among components in interleaved modes
- Lossless color decorrelation transforms specified in Part 2 of the standard. Very effective as pre-processing to JPEG-LS in some color spaces. Example:

$$R \rightarrow R - G$$
$$G \rightarrow G$$
$$B \rightarrow B - G$$

as with prediction errors, use subtraction mod α and remapping to $\left[-\frac{\alpha}{2}, \frac{\alpha}{2}\right)$ to preserve alphabet size

JPEG-LS Features: Color Images

- *Palletized images*: JPEG-LS syntax allows for description of palette tables and coding in index space

index	R	G	B
0	0	0	0
1	12	17	23
2	32	123	100
3	150	200	30
⋮	⋮	⋮	⋮
254	130	77	90
255	255	255	255

- Same feature useful for remapping images with “sparse histograms”