Surround Codes, Densification, and Entropy Scan Performance

Russell Leidich
https://agnentropy.blogspot.com

November 11, 2017

Keywords: surround code, surround function, densification, entropy, sparsity, compression

0. Abstract

Herein we present the “surround” function, which is intended to produce a set of “surround codes” which enhance the sparsity of integer sets which have discrete derivatives of lesser Shannon entropy than the sets themselves. In various cases, the surround function is expected to provide further entropy reduction beyond that provided by straightforward delta (difference) encoding alone.

We then present the simple concept of “densification”, which facilitates the elimination of entropy overhead due to masks (symbols) which were considered possible but do not actually occur in a given mask list (set of symbols).

Finally we discuss the ramifications of these techniques for the sake of enhancing the speed and sensitivity of various entropy scans.

1. The First Delta (Discrete Derivative)

Suppose we have a mask list H, which we define as a set of Q whole numbers less than Z, where (Q>0) and (Z>1). For example:

$$H = \left[2, 3, 6, 5, 2, 0, 1, 3, 2\right]$$

such that (Q=9) and (Z=7). Its Shannon entropy E_H is then given by

$$E_H \equiv (Q \ln Q) - \sum_{M=0}^{Z-1} F_H(M) \ln F_H(M)$$

where (0 ln 0) terms are treated as zero, and $F_H(M)$ is the frequency of mask M in H:

$$F_H(M) \equiv \sum_{K=0}^{Q-1} (H_K = M)$$

where $(H_K=M)$ is one if H_K equals M, else zero. (For its part, H_K is the mask at zero-based index K of mask list H.) So in this case, we have

$$F_H = \{1, 1, 3, 2, 0, 1, 1\}$$

because there is one zero, one one, 3 2s, 2 3s, etc. Therefore

$$E_H = (9 \ln 9) - (1 \ln 1) - (1 \ln 1) - (3 \ln 3) - (2 \ln 2) - (0 \ln 0) - (1 \ln 1) - (1 \ln 1)$$
$$E_H \approx 15.1$$

We could encode this sequence with less entropy by starting with an implicit zero and “deltafying” from left to right, resulting in the “first delta” H_1 of H:

$$H_1 = \{2, 1, 3, -1, -3, -2, 1, 2, -1\}$$

Note that, for all components H_{1K} of H_1

$$(-Z) < H_{1K} < Z$$

due to the fact that $(H_K<Z)$ for all K. Furthermore, the first (leftmost) masks of H and H_1 are always equal because “deltafication” begins from an implied first source mask of zero, which is sufficient to guarantee invertibility. So if we consider the frequency list F_{H1} of H_1, starting with the frequency of (-6) on the left and ending with the frequency of 6 on the right, we have

$$F_{H1} = \{0, 0, 0, 1, 1, 2, 0, 2, 2, 1, 0, 0, 0\}$$

because H_1 contains one (-3), one (-2), etc. Therefore

$$E_{H1} = (9 \ln 9) - (1 \ln 1) - (1 \ln 1) - (2 \ln 2) - (0 \ln 0) - (2 \ln 2) - (2 \ln 2) - (1 \ln 1)$$
$$E_{H1} \approx 15.6$$

which has actually foiled our attempt at sparsity enhancement (effectively, data compression) because E_{H1} is actually slightly greater than E_H, even though a quick inspection of the latter might lead us to suspect that it's sparse in the first delta. What went wrong?

First of all, note that Shannon entropy is only a rough estimate of encoding cost when Q and Z are "small", as in this case; a more precise answer would be had from agnentropy, as I described in "Introduction to Agnentropy".

That said, the larger problem is that we're permitting the expression of states which cannot necessarily occur. For example, when deltafying from 3 to something, that "something" cannot be less than zero because masks are always whole; however the above encoding scheme would allow for that, which is wasteful. We could then say that codes wrap from (-1) to (Z-1), (-2) to (Z-2), etc., but in that case we would have 2 codes for every nonzero delta, which is still wasteful. So we could then define the delta to be less than Z, and whole, so that it wraps back to zero after (Z-1). But this idea is also problematic because then the same code could actually imply big deltas or small deltas, depending on the circumstances, which would still result in inefficiency.

2. The Surround Function

The surround function generates whole numbers which preserve all of the information in a mask list (which excludes Z itself). In various practical cases it reduces mask list entropy to a further extent than deltafications. As before, assume we have mast list H given by

$$H = \lfloor 2, 3, 6, 5, 2, 0, 1, 3, 2 \rfloor$$

where (Q=9) and (Z=7).

We will now take the surround codes from left to right. Just like deltas, surround codes have the sense of “from” and “to” within the limit of some given Z. The first mask on the left, which is 2, needs to be “surrounded from” an implied source value. This value is defined as the integer floor of (Z/2). That is to say we must compute

$$H_{S0}=S(\lfloor\frac{Z}{2}\rfloor,H_0,Z)=S(3,2,7)$$

which is “the surround from the integer floor of (Z/2) to H_0 when all masks are known to be less than Z” or in this specific case “the surround from 3 to 2 when (Z=7)”. But first of all, why (Z/2)?

We start with (Z/2) as our implied source simply because we assume that H was derived by biasing a raw list of (maybe signed) integers so that its minimum possible (but not necessarily present) value is zero. So for example we might map temperature values of (-10) to 50 to mask values of zero to 60. We expect that those particular extrema were chosen so that the most frequent values end up close to the middle. Furthermore, the floor function is important because if Z is even, then we want to stick with the convention of the *even* “middle” value being treated as “the” middle value; and if Z is odd, then we want to select the *actual* middle value because it’s unique. Of course, it would be conceptually simpler to start with a source of zero. However, that convention would likely result in the first surround code appearing to be more anomalous (information rich) than it actually is.

We then proceed as follows:

$$H_{S1}=S(2,3,7)$$

$$H_{S2}=S(3,6,7)$$

etc. so that the “surround list” H_S of H is given by:

$$H_S(H,Q,Z)\equiv\left\{S(\lfloor\frac{Z}{2}\rfloor,H_0,Z),S(H_0,H_1,Z),S(H_1,H_2,Z)\ldots S(H_{Q-2},H_{Q-1},Z)\right\}$$

The surround list, then, contains Q items, just like H. Furthermore, all surround codes are whole numbers less than Z, as opposed to integers in the range of (1-Z) to (Z-1), as in the case of signed deltas. But what, then, is this the surround function S?

To answer that, let's begin with an example. We need to find S(2, 5, 7). Start by making a sorted list of whole numbers less than Z, starting from zero:

$$\{0,1,2,3,4,5,6\}$$

Next, replace each item with its distance to the source (which is 2):

$$\{2,1,0,1,2,3,4\}$$

(Thus we have "surrounded" 2, hence the name.) Obviously, distance alone does not constitute an invertible code. So for example we need to distinguish zero (at distance 2) from 4 (also at distance 2). We do this so that lesser destinations end up with odd codes, and greater destinations end up with even codes, but only for source values which would otherwise result in ambiguous cases; remaining codes are assigned monotonically with distance. So now we have

$$\{3,1,0,2,4,5,6\}$$

Now all the codes are unique whole numbers less than Z. We can simply look up S(2, 5, 7), which turns out to be 5. Let's try another example.

Suppose we want to compute S(7, 5, 8), which implicitly means that now (Z=8). We start with

$$\{0,1,2,3,4,5,6,7\}$$

then replace each item with its distance to 7, resulting in

$$\{7,6,5,4,3,2,1,0\}$$

In this case, there are no ambiguous codes, so we're done. Therefore S(7, 5, 8) is also 5. Let's try one more example.

Suppose we want to compute S(4, 2, 8). We start with

$$\{0,1,2,3,4,5,6,7\}$$

then replace each item with its distance to 4, resulting in

$$\{4,3,2,1,0,1,2,3\}$$

Now replace ambiguous codes in the usual manner, resulting in

$$\{7,5,3,1,0,2,4,6\}$$

Thus S(4, 2, 8) is 3.

So returning once again to the initial mask list

$$H=\{2,3,6,5,2,0,1,3,2\}$$

where (Q=9) and (Z=7), we can compute its corresponding surround list using the method described above, resulting in

$$H_S(H,Q,Z)=\left\{S(\lfloor\frac{7}{2}\rfloor,2,7),S(2,3,7),S(3,6,7)\ldots S(3,2,7)\right\}$$

$$H_S(H,Q,Z)=\{1,2,6,1,4,3,1,3,1\}$$

which has a corresponding frequency list given by

$$F_{H1}=\{0,4,1,2,1,0,1\}$$

From which we can compute the Shannon entropy E_{HS} as

$$E_{HS}=(9\ln 9)-(4\ln 4)-(1\ln 1)-(2\ln 2)-(1\ln 1)-(0\ln 0)-(1\ln 1)$$
$$E_{HS}\approx 12.8$$

which is indeed lesser than either H or its first delta H_1. This would presumably result in smaller agnentropy as well, which implies smaller storage requirements. But more to the point, the comparatively low value of E_{HS} implies that it should be more responsive to small changes in the masks of H, than E_H itself, on a fractional basis. To put it another way, we expect that "surroundification" (the conversion of a mask list to a surround list) will enhance our sensitivity to signals which are sparse in their first delta.

In practice, deltafication *followed by* surroundification seems most effective for sake of detecting such signals. surround codes are most usefully applied after deltafication. If deltafication is not appropriate (because the first delta of the mask list in question is *not* sparse), then surround codes are best applied after densification, which we'll explore later in this paper.

Critically, surround codes tend to *enhance* the sensitivity of "absolute entropy" functions such as Shannon entropy or agnentropy; whereas they tend to *degrade* the sensitivity of (exo)divergences, which I discussed in "Anomaly Detection and Approximate Matching via Entropy Divergences". The reason seems to be that while the additional sparsity afforded by surroundification helps reduce noise in the former, it comes at the cost of obscuring the physical meaning of a mask; the effect is to weakly "encrypt" the signal, thereby reducing entropy contrast (to the detriment of scans which measure divergences) while at the same time also reducing absolute entropy (to the benefit of scans which measure it).

3. A Procedural Definition of the Surround Function

$S(M_0, M_1, Z)$ in the sense of "the surround S from mask M_0 to mask M_1 with neither mask exceeding (Z-1)" is procedurally defined by

```
if [M0<=M1] {
   S=M1
   if [Floor(M1/2))<M0] {
       S=2(M1-M0)
   }
```

} else {
   S=Z-1-M_1
   if [Floor(S/2)<=(Z-1-M_0)] {
      S=2(M_0-M_1)-1
   }
}

where, as always, (Z-1) is merely an upper bound which might in fact exceed every mask in the entire list. We can of course also invert surround codes using “unsurround” codes.

4. A Procedural Definition of the Unsurround Function

Surround codes would be of little use if they could not be inverted. In order to do this, we move from left to right across the surround list, converting it back into the original mask list. As before, we start with the integer floor of (Z/2) as the implicit source. The only implied information is Z, which must be the same value as that used to create the surround list in first place.

U(M_0, S, Z) in the sense of “the unsurround U from mask M_0 to surround S, where S was computed with the same Z” is procedurally defined by

if [M_0<=Floor((Z-1)/2)] {
   U=S
   if [Floor(S/2)<M_0] {
      U=Floor(S/2)
      if [S is even]{
         U=M_0+U
      }else{
         U=M_0-U-1
      }
   }
} else {

```
    U=Z-1-S
    if [Floor(S/2) <= (Z-1-M0)] {
        U=Floor(S/2)
        if [S is even] {
            U=M0+U
        } else {
            U=M0-U-1
        }
    }
}
```

5. Potential Generalization of the Surround Function

(Un)surround codes readily generalize to more than one dimension, provided that one begins with the geometric “distance index” definition first illustrated above. The only additional consideration is that, in higher dimensions, we have mask N-tuples instead of masks, so multiple such vectors might be equidistant from the same source. This complicates code disambiguation. One solution is to prioritize codes by X coordinate, then Y coordinate, etc., such that moving down implies lesser codes than moving up by the same distance. Or we could prioritize by angle from each axis, with the X axis dominating over the Y axis, etc. Yet another approach would be to sort by Hilbert (space-filling) curve coordinate. Suffice to say that different generalizations are suitable to different applications, but all of them are beyond the scope of this paper.

6. Densification (Mask Span Minimization)

Recall that Z is defined such that it exceeds all masks in a given mask list. However, there is no requirement that it exceed the maximum mask by exactly one. Furthermore there is no requirement that every *possible* mask actually occur. For example, suppose (Z=9) and we have

$$H = [3, 7, 1, 2, 4, 5, 2, 7, 4]$$

In this case, clearly, Z could just as easily be 8. In fact, reassigning it as such would not affect the Shannon entropy because such entropy does not account for the overhead of determining which masks, from zero to (Z-1), actually occur with which frequencies. (This is why Shannon entropy does not accurately anticipate the size of arithmetically compressed mask lists, regardless of implementation efficiency.) Other forms of entropy, such as agnentropy, do however depend on Z, and would be reduced by a reduction in Z. But there is more we can do.

Note that H is missing zero and 6 (even if we reduce Z to 8). So we could "densify" it, resulting in the following densification H_D:

$$H_D = \{2, 5, 0, 1, 3, 4, 1, 5, 3\}$$

First of all, note that mask order has been preserved. So the old minimum mask (one) has been mapped to the new minimum (by definition, zero). Secondly, Z has dropped yet again, this time to 6. For the aforementioned reasons, however, the Shannon entropy remains unchanged.

There is one obvious reason to densify, which is to improve the speed of entropy scans. In theory, acceleration is afforded by improved memory footprint density, for instance in the course of accessing lookup tables for logs.

And of course if we actually intend to encode H using arithmetic compression, reducing Z will facilitate reduced compressed size.

The less obvious reason to densify is to enhance the *sensitivity* of entropy scans. Given, for example, that agnentropy is monotonically related to Z, then minimizing Z should make changes to H_D stand out from the noise more starkly than changes to H itself, in terms of fractional changes in agnentropy.

Empirically, however, this remains an open question. On the face of it, the tightness of H_D should cause log operands to be smaller and closer in magnitude, leading to greater precision and thus sensitivity. However, smaller

log operands actually require *more* terms to converge to the same degree of precision, so in fact the sensitivity (and even the speed) may degrade.

But there's more to the mystery than meets the eye. Consider the formula for the agnentropy E of a mask list with mask frequencies $F_H(M)$ for all (M<Z), mask count Q, and mask span Z:

$$E \equiv \ln((Q+Z-1)!) - \ln((Z-1)!) - \sum_{M=0}^{Z-1} \ln((F_H(M))!)$$

(Beware the factorials.) Obviously, reducing Z reduces E. But given a *pair* of mask lists, it's not clear that reducing Z in the first case to Z_1 and in the second case to Z_2 can even *possibly* change the *order* of their agnentropies – regardless of the relationship among Z_1, Z_2, and said agnentropies. The reason is that the sum term is actually constrained in bizarre combinatorical ways by Z itself, so for the most part it just seems to come along for the ride in such a manner as to conserve order despite the separate densification processes. So in practice, it would seem that densification has no bearing on *relative* sensitivity in the sense of being able to determine whether mask list X contains more information than mask list Y. (*How much* more *can* of course change pursuant to densification.) I cannot prove this order invariance, and I haven't tried very hard to discover a counterexample, but in practice it always seems to work out this way, much to my consternation.

Nevertheless, I've added both surroundification and densification to the open-source Agnentro entropy toolkit for your experimental purposes.

7. Remarks

The utility of surroundification and densification for the purposes of entropy scan acceleration and enhancement varies considerably by data set and the particular entropy measurement in question. At least, they come at a low computational cost and achieve improved signal sparsity. Plenty of room remains for further study.