Introduction to Entropy Transforms

A family of rapid of data scan methods is introduced whereby conventional signal search might be accelerated by an order of magnitude or more at minimal cost to sensitivity.

Russell Leidich
https://agnentropy.blogspot.com

May 11, 2017

Keywords: search, needle, haystack, signal, detection, entropy, transform, noise, compressed sensing, agnentropy, exoentropy, exoelasticity, interentropy, diventropy, logfreedom, dyspoissonism, variance, kurtosis, sweep, transform

0. Abstract

We have at our disposal a wide variety of discrete transforms for the discovery of "interesting" signals in discrete data sets in any number of dimensions, which are of particular utility when the default assumption is that the set is mundane. SETI, the Search for Extraterrestrial Intelligence, is the archetypical case, although problems in drug discovery, malware detection, financial arbitrage, geologic exploration, forensic analysis, and other diverse fields are perpetual clients of such tools. Fundamentally, these include the Fourier, wavelet, curvelet, wave atom, contourlet, brushlet, etc. transforms which have churned out of math departments with increasing frequency since the days of Joseph Fourier. A mountain of optimized applications has been built on top of them, for example the Fastest Fourier Transform in the West[1] and the Wave Atom Toolbox[2].

Such transforms excel at discovering particular classes of signals. So much so that the return on investment in new math would appear to be approaching zero. What's missing, however, is efficiency: the question must be asked as to

when such transforms are computationally justifiable.

Herein we investigate a preprocessing technique, abstractly known as an "entropy transform", which, in a wide variety of practical applications, can discern in essentially real time whether or not an "interesting" signal exists within a particular data set. (Entropy transforms say nothing as to the *nature* of the signal, but merely how interesting a particular subset of the data appears to be.) Entropy transforms have the added advantage that they can also be tuned to behave as crude classifiers -- not as good as their deep learning counterparts, but requiring orders of magnitude less processing power. In applications where identifying many targets with moderate accuracy is more important than identifying a few targets with excellent accuracy, entropy transforms could bridge the gap to product viability.

It would be fair to say that in the realm of signal detection, discrete transforms should be the tool of choice because they tend to produce the most accurate and well characterized results. But processor power and execution time are not free! Particularly when, as in the case of SETI, the bottleneck is the rate at which newly acquired data can be processed, a more productive approach would be use to cheap but reasonably accurate O(N) transforms to filter out all but the most surprising subsets of the data. This would reserve processing capacity for those rare weird cases more deserving of closer inspection.

I published Agnentro[3], an open-source toolkit for signal search and comparison. The reason, first and foremost, was to support these broad and rather unintuitive assertions with numerical evidence. The goal of this paper is to formalize the underlying math.

1. Prerequisite Knowledge

To begin with, the reader is presumed to be familiar with the terminology of mask lists[4] and the math behind agnentropy[5]. A knowledge of logfreedom and dyspoissonism[6] would also be helpful because it provdes an intuitive sense of how it's possible to generalize the notion of "interesting" signals floating in a sea of noise.

2. The Semantics of Entropy

For our purposes here, "entropy" is simply a measure of information content. It can have units of bits, although for the sake of compuational accuracy it's preferable to work in nats (bits times (ln 2)). We'll follow that convention here.

"Entrometry", then, is the use of entropy metrics to study various phenomena, whether physical or purely mathematical in nature.

Some physicists will no doubt be offended by this etymology, as "entropy" has a specific thermodynamic definition which resolves only indirectly to a vague notion of information content. Sorry, but it just wastes too much time to repetitively refer to "information content" when we could just as easily refer to "entropy" and expect to be understood.

The most important thing to understand about entropy is that it only has meaning from the perspective of a probability model for the distribution of data sets which could possibly occur in the wild, to we which we refer as the "generator" in the agnentropy context. To be clear, there is no *absolute* level of entropy. Rather, the entropy of a set is just the negative of the log to some base (2 or *e* in the case of bits or nats, respectively) of the probability of the set in question actually occurring, as implied by said model. For example, if we expect ones and zeroes to occur with equal probability, and have no further assumptions, then the entropy in bits of any bitstring is ostensibly the just number of bits it contains -- but merely *ostensibly* because there is information in the *number of bits* itself, which must be conveyed in a so-called "universal code". And furthermore there's entropy in the assumption that said code precedes the bitstring, as opposed to being located elsewhere. We could continue indefinitely, but the bottom line is unavoidable: the definition of entropy depends on the expectations, whether implicit or explicit, of the machine seeking to measure it. In other words, entropy is in the eye of the beholder!

By the way, as I mentioned in [5], "metric" is used in the qualitative sense of

something that measuring something -- entropy, in this case. Mathematical purists will rightly point out that the formulae discussed herein are generally "divegences" because they violate the triangle inequality and thus do not fit the formal definition of a metric. By the more mathematical jargon one uses, the less likely one is to find an audience for one's ideas among engineers.

3. Sweep Transforms and the Windowing Problem

The aforementioned discrete transforms are all biased in the sense that they're optimized for perfectly aligned principal components of specific wavelengths, be they sine waves, Haar wavelets, curvelets, or whatever. Their tremendous success derives from the fact that, if properly applied to an amenable class of data sets, said components are nevertheless likely to resonate with similar counterparts in the sets. This is why, for instance, the JPEG compression algo seems to be able to compress just about any photo using the discrete cosine transform, despite the fact that the physical world is not composed of cosine wavelets.

But therein lies the problem: all discrete transforms suffer from so-called "windowing" or "filtering" problems. This refers to the inaccuracy induced in the transition from the analog world of calculus to the discrete world of integers. In particular, the continuum of amplitudes and wavelengths representable in the former yield to quantized approximations of themselves in the latter.

"Sweep transforms" seek to eliminate the *alignment* windowing problem to the maximum extent possible, in particular by sliding a window of some particular width across the data, one mask (sample) at a time. The *wavelength* windowing problem remains. However, entropy in all its manifestations explored herein has no sense of wavelength; the mask is the fundamental unit of analysis, and all that matters is their relative rates of occurrence ("frequency", which is an unfortunate term of art having nothing at all to do with the reciprocal of wavelength).

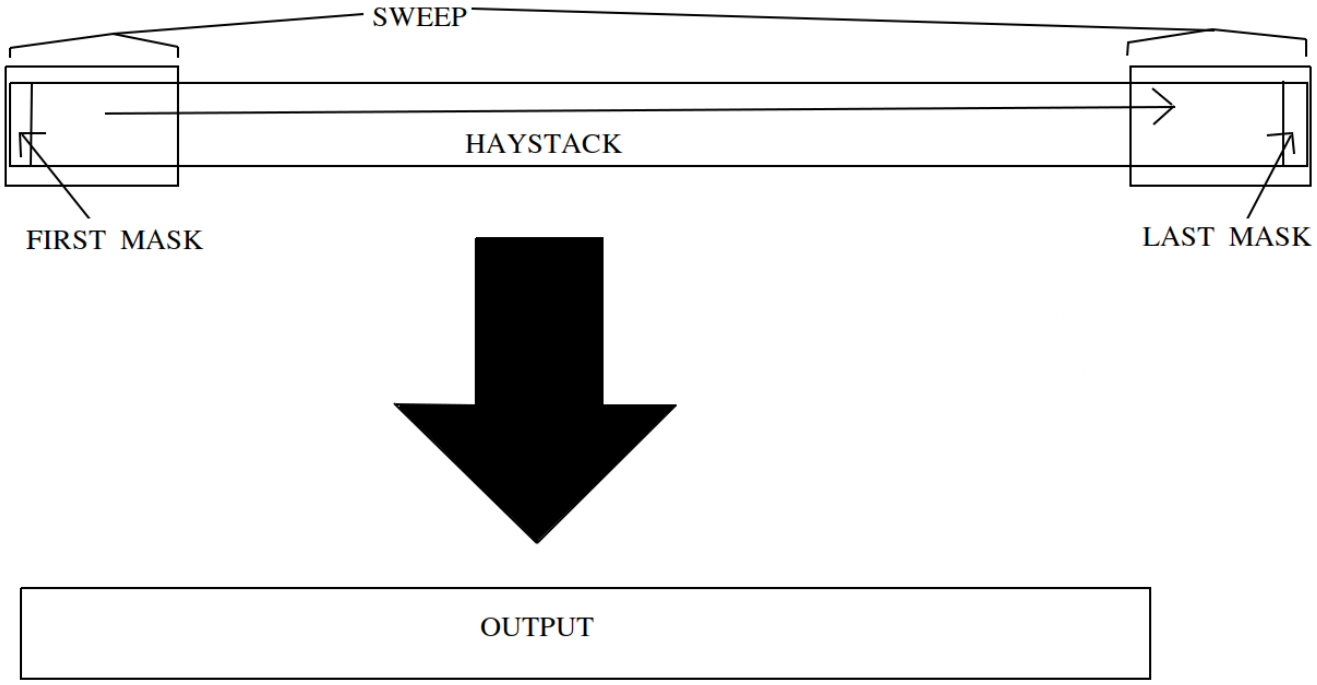
All entropy transforms are sweep transforms. Let's first define the latter.

3.1. Haystack, Needle, and Sweep

The "haystack" is the name we give to the data set itself (which must consist of at least one mask) in the sense of "something to be searched for something else". That "something else" is the "needle" -- a second data set, the size of which is also nonzero but otherwise unrestricted. It might seem counterintuitive to have a needle larger than the haystack which is to be searched for it, but the concepts are qualitative in the sense, for instance, that we might search a school photo for a face which happens to have been scanned a much higher resolution and therefore consists of more masks (pixels, in this case) than the former.

The "sweep" is simply the nonzero number of masks in the "sweep window" which moves from start to end of the haystack one mask at a time, such that the window always covers a contiguous and uniformally sized subset of haystack. Inside this window, some particular function is evaluated. In this case, but not in the general case, that function is an entropy metric.

There are 3 basic "sweep modes": haystack, fixed, and needle. In haystcak mode, the sweep equals the number of masks in the haystack; the window doesn't slide because it has no space to do so. In fixed mode, the sweep is a natural number (positive integer), which again cannot exceed the number of masks in the haystack. This is probably best conveyed graphically:



Notice that the output consists of at most as many masks as the haystack because (sweep minus one) masks are missing on account of the requirement that the sweep window be of uniform width. (This is why the output is visibly shorter than the haystack in the graphic.)

In needle mode, finally, which is only valid with "bivalent" sweep transforms such as the diventropy transform discussed later, the sweep is equal to the number of masks in the needle; this mode is only valid if the needle size doesn't exceed the haystack size.

3.2. The Sweep Transform Formula

Formally, then, if we have a haystack mask list H={H_0, H_1,... H_{Q-1}} consisting of Q whole numbers less than Z, then the "output list" Y of a sweep transform wherein a "sweep function" X is applied to the sweep window of width S starting at each zero-based "sweep base index" J, where (J<=(Q-S)), then

$$Y \equiv \{Y_0, Y_{1,...} Y_{Q-S}\}$$

where

$$Y_J \equiv X(\{H_J, H_{J+1}, \ldots H_{J+S-1}\})$$

Furthermore, to the extent that X is commutative, the quantity (Y_{J+S}-Y_J) is then purely a function of H_J, and H_{J+S} -- in other words, the mask next to leave and next to enter sweep window, respectively. All of the entropy metrics presented herein adhere to this criterion, which is the fundamental reason why their complexity is asymptotically O(Q). But in the general case, there's no requirement that Y_J be scalar or that the sweep function be linear, so one could for example imagine a "high accuracy" Fourier transform which consisted of a sweep transform *of* fast Fourier transforms.

3.3. Higher Dimensional Sweep Transforms

Sweep transforms are trivially extensible to higher dimensions. In D dimensions, Z remains a natural number; while J, Q and S generalize to D-tuples of naturals such that the components of J and S are subject to the same constraint with respect to the corresponding components of Q. Furthermore, the D-sweep-window moves in each dimension by toggling a (D-1)-hyperprism of constant geometry on each side, rather than a single mask.

In all respects but one, this generalization is so trivial as to discourage further discussion. That one respect is the path of traversal of the D-haystack. Notionally, the path is left to right, then down by one, then left to right, etc. through all D dimensions. But in practice, this is likely to be inefficient due to cache thrashing. For better performance, consider walking the sweep base index vectors in the order of their equivalent Hilbert curve[7] coordinates, assuming a curve of infinite length. This practice will reduce cache misses.

4.0. Entropy Transforms

Again, an entropy transform is a sweep transform in which the sweep function is an entropy metric. Those discussed here are defined for use in one dimension, but they generalize to D dimensions in the aforementioned manner.

The output list of an entropy transform consists of a set of real numbers

representing the entropy of the corresponding sweep window at each step. For its part, the Agnentro toolkit uses interval math to calculate entropy values, so the output is a list of "fractervals", which are rational fraction subintervals of [0, 1]. It also supports the computation of all entropy metrics presented herein. The use of interval arithmetic of one form or another is strongly encouraged, as entropy transforms involve repetitive feedback of previously generated results, which taxes numerical precision.

Generally, the sweep windows found to have the greatest, or the least, entropy are the most interesting in practice. However, particularly insofar as classification tasks are concerned, it often seems to be the case that similar objects will exhibit similar entropy when analyzed at the same scale. This facilitates "bandgap entrometry", which is the classification of phenomena depending upon the region into which their entropy falls. There are normalization methods which allow phenomena manifesting on various scales to be compared as though they manifest on the same scale, which we'll discuss later.

4.1 Monovalent Entropy Transforms

These transforms have only a haystack and a sweep as inputs, as the point is to compute some particular entropy parameter of all possible sweeps, then sort them in order to find the most surprising information. ("Monovalent" means "one face", which refers to the haystack.)

4.1.1. The Shannon Entropy Transform

The Shannon entropy E of a haystack H consisting of Q masks on the interval [0, Z-1] is given by

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

where $F_H(M)$ is the frequency of mask M in the haystack:

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

where (H_K=M) is one if H_K equals M, else zero.

Now, technically, E is the "compromised" Shannon entropy discussed in [5] -- not the true (analog) Shannon entropy. Regardless of this, it fails to account for the cost of conveying the frequency information itself (because it's an asymptotic metric unsuitable for "small" sets).

We can then define Y_J, which is the Shannon entropy of a sweep window of width S based at zero-based index J, (J<=(Q-S)):

$$Y_J \equiv (S \ln S) - \sum_{K=J}^{J+S-1} \ln F_S(J, H_K)$$

where we've broken out the log product terms into individual subtrahends and F_S(J, H_K) is the frequency of H_K inside the sweep window:

$$F_S(J, M) \equiv \sum_{K=J}^{J+S-1} (H_K = M)$$

We could compute all (Q-S) values of Y_J in order to fully populate Y. However, there is a faster, if more serialized and thus numerically taxing way: after computing Y_0 as above, compute the "entropy delta" ΔY_J, defined as (Y_{J+S}-Y_J), at each step starting with (J=0), then add it to the previous entropy:

$$\Delta Y_J \equiv \left(\sum_{K=J+1}^{J+S} \ln F_S(J, H_K) \right) - \left(\sum_{K=J}^{J+S-1} \ln F_S(J, H_K) \right)$$

If (H_J=H_{J+S}), then, as with all entropy transforms, ΔY_J is zero, so we can just set Y_{J+1} to Y_J, then move on to the next step. Otherwise things get complicated because a change in the frequency of a single mask may affect the log terms corresponding to many other copies of that mask within the sweep window. Fortunately, after accounting for those changes, the resulting expression for ΔY_J is relatively inexpensive to compute:

$$\begin{aligned}\Delta Y_J \equiv & F_S(J, H_J) \ln(F_S(J, H_J)) \\ & + F_S(J, H_{J+S}) \ln(F_S(J, H_{J+S})) \\ & - (F_S(J, H_J) - 1) \ln(F_S(J, H_J) - 1) \\ & - (F_S(J, H_{J+S}) + 1) \ln(F_S(J, H_{J+S}) + 1)\end{aligned}$$

which can be rapidly computed, given a lazily populated table of logs (and, for enhanced efficiency, the differences between logs of successive whole numbers). (If it occurs, (0 ln 0) must be treated as zero.) What this formula is saying, essentially, is that the entropy is changing by an amount which reflects the loss of H_J and the addition of H_{J+S}. In the rare case that $F_S(J, H_J)$ is initially equal to $(F_S(J, H_{J+S})+1)$, ΔY_J will vanish to zero; this fact can be exploited to reduce the unnecessary loss of numerical precision.

4.1.2. The Agnentropy Transform

Similarly to Shannon entropy, the agnentropy E of a haystack given by

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

gives rise to the following in a sweep context:

$$Y_J \equiv \ln((Q+Z-1)!) - \ln((Z-1)!) - \sum_{M=0}^{Z-1} \ln(F_S(J, M)!)$$

where $F_S(J, M)$ is the frequency of mask M within the sweep window, and we've recast the loggamma terms as sums of logs . This in turn implies that

$$\Delta Y_J \equiv \ln F_S(J, H_J) - \ln(F_S(J, H_{J+S}) + 1)$$

which is, of all the entropy transforms herein, by far the fastest to compute and least numerically taxing. (A simple lookup table would suffice for the logs.) And as in the Shannon case, ΔY_J is zero and move on if in fact $F_S(J, H_J)$ is initially equal to $(F_S(J, H_{J+S})+1)$. Moreover, because it implicitly accounts for the cost of encoding the frequency list, agnentropy is also more accurate than Shannon entropy, to the extent that all Z masks are actually possible.

4.1.3. The Obtuse Variance Transform

To be sure, variance isn't considered an entropy metric, but we can use it that way, as it turns out to be quite sensitive to faint sinosoidal signals shrouded in Gaussian noise. The variance E of a haystack is given by:

$$E \equiv \frac{1}{Q} \sum_{J=0}^{Q-1} (H_J - U)^2$$

where U is the mean (average) of the haystack H. Which in a sweep context gives rise to

$$Y_J \equiv \frac{1}{S} \sum_{K=J}^{J+S-1} (H_K - U)^2$$

provided that we make the "obtuse mean approximation" that U is the same in all sweep windows. (If we don't, then complexity explodes to O(QS) due to the fact that a change in the mean affects *all* addends of the above sum in a nonlinear manner.) This assumption facilitates rapid computation of the entropy delta:

$$\Delta Y_J \equiv \frac{(H_{J+S} - U)^2 - (H_J - U)^2}{S}$$

wherein the division by S can be deferred until and unless necessary (because in practice most results rank too low to be deemed interesting, and are thus discarded), leaving us with a simple difference-of-squares in most cases, the minuend and subtrahend of which could be cached in a lazily populated table of size O(Z).

For comparative analysis purposes, and to the extent that our assumption about the mean is accurate, the obtuse variance transform is equivalent to a standard deviation sweep transform because variance is monotonic with the latter. This assumption starts to fall apart under high noise regimes, however, in which agnentropy is an empirically superior metric.

4.1.4. The Obtuse Kurtosis Transform

We can similarly define an entropy metric E as the kurtosis of H:

$$E \equiv \frac{Q \sum_{J=0}^{Q-1} (H_J - U)^4}{\left(\sum_{J=0}^{Q-1} (H_J - U)^2\right)^2}$$

which in a sweep context, and under the same obtuse mean approximation, gives rise to

$$Y_J \equiv \frac{S \sum_{K=J}^{J+S-1} (H_K - U)^4}{\left(\sum_{K=J}^{J+S-1} (H_K - U)^2\right)^2}$$

which implies that

$$\Delta Y_J \equiv S \left\{ \sum_{K=J+1}^{J+S} \frac{(H_K - U)^4}{((H_K - U)^2)^2} - \sum_{K=J}^{J+S-1} \frac{(H_K - U)^4}{((H_K - U)^2)^2} \right\}$$

which demonstrates exactly why the obtuse kurtosis transform is so expensive: it's cheaper just to evaluate Y_J from scratch on every step, at best deferring the multiplication by S until and unless necessary.

Insofar as the detection of sinusoidal waves is concerned, obtuse kurtosis appears to be "obtuse" indeed, in the sense that it's empirically much less sensitive than other entropy metrics. Nevertheless it's included here because in theory it might excel in other regimes known to exhibit nongaussian kurtosis, such as asset price streams.

4.1.5. The Logfreedom Transform

The logfreedom formula[8] in "eubits" (the word I invented for nats, before I knew the latter) is:

$$L \equiv \ln Q! + \ln Z! - \ln H[0]! - \sum_{F=1}^{K} \ln H[F]! - \sum_{F=1}^{K} H[F] \ln F!$$

where K is the maximum value of frequency F having nonzero population. We can recast this formula in the style of the foregoing entropy metrics:

$$E \equiv \ln Q! + \ln Z! - \sum_{F=0}^{Q} (\ln P_H(F)! + P_H(F) \ln F!)$$

where $P_H(F)$ is H[F] in the former, simply because we've already assigned H to be the haystack. (We also change brackets to parentheses and subsume the (H[0]!) term into the sum, bearing in mind that zero factorial is one). For the sake of simplicity, we're summing over all possible values of F, the populations of most of which being zero. This formula applies straightforwardly to a sweep window:

$$Y_J \equiv \ln Q! + \ln Z! - \sum_{F=0}^{S} (\ln P_S(J,F)! + P_S(J,F) \ln F!)$$

where

$$P_S(J,F) \equiv \sum_{M=0}^{Z-1} (F_S(J,M) = F)$$

meaning that $P_S(J, F)$ is the population of frequency F within the sweep window with base index J. This yields

$$\begin{aligned}\Delta Y_J \equiv {} & \ln P_S(J, F_S(J, H_J)) + \ln(F_S(J, H_J)!) - \ln(P_S(J, F_S(J, H_J) - 1) + 1) - \ln(F_S(J, H_J) - 1)! \\ & + \ln P_S(J, F_S(J, H_{J+S})) + \ln F_S(J, H_{J+S})! - \ln(P_S(J, F_S(J, H_{J+S}) + 1) + 1) - \ln(F_S(J, H_{J+S}) + 1)!\end{aligned}$$

which must be computed in the following order:

1. Compute all terms involving H_J.

2. Decrement $P_S(J, F_S(J, H_J))$.

3. Increment $P_S(J, F_S(J, H_J)-1)$.

4. Decrement $F_S(J, H_J)$.

5. Compute all terms involving H_{J+S}, yielding ΔY_J.

6. Decrement $P_S(J, F_S(J, H_{J+S}))$.

7. Increment $P_S(J, F_S(J, H_{J+S})+1)$.

8. Increment $F_S(J, H_{J+S})$.

This ordering is necessary to avoid the possibility of colliding population updates.

Despite appearances, no log operand can actually be zero because: (1) $P_S(J, F_S(J, H_J))$ is nonzero because this term is evaluated before $F_S(J, H_J)$ is decremented. (2) $(P_S(J, F_S(J, H_J)-1)+1)$ and $(P_S(J, F_S(J, H_{J+S})+1)+1)$ are nonzero because populatin is by definition whole, and we're adding one to it. (3) $P_S(J, F_S(J, H_{J+S}))$ is nonzero because if $F_S(J, H_{J+S})$ zero, then $P_S(J, 0)$ is nonzero because at least one mask is missing from the sweep window; otherwise if $F_S(J, H_{J+S})$ is nonzero then it must appear in the sweep window already, in which case its frequency and thus the population of its frequency is by definition nonzero.

Further simplification yields

$$\Delta Y_J \equiv \ln P_S(J, F_S(J, H_J)) - \ln(P_S(J, F_S(J, H_J)-1)+1) + \ln F_S(J, H_J)$$
$$+\ln P_S(J, F_S(J, H_{J+S})) - \ln(P_S(J, F_S(J, H_{J+S})+1)+1) - \ln(F_S(J, H_{J+S})+1)$$

where the same order of operations applies and it's still impossible to generate a (ln 0) term. By the way, it often occurs that $F_S(J, H_J)$ equals $(F_S(J, H_{J+S})+1)$, in which case ΔY_J is simply zero, which can be exploited to thwart precision erosion.

For its part, $P_S(J, F)$ is best implemented via a sparsely populated list mapping frequency F to its population in the sweep window, which must be on [0, S]. But this can be unweildy if S is much larger than the hardware cache. So to save space, a Poisson cache of the sort used by Agnentro Scan

(via the included Poissocache library) would be useful.

4.1.6. The Exoentropy Transform

"Exoentropy" is short for "exotic entropy", which is a measure of how *unlike* the surrounding data the sweep window actually is. More precisely, it's the amount of information which would be required to losslessly encode the sweep window, given only Q, Z, and the mask probability distribution implied by the agnostic frequencies of all masks in the "exosweep", which is the haystack excluding the sweep window. This method is optimized for use in the search for anomalous signals of unforeseeable geometry. Indeed, in tests of signal discovery performance under high noise conditions, it outperformed every other entropy transform except for exoelasticity, a conceptual derivative of exoentropy which we'll discuss later.

Recalling from [5] that agnostic frequency is simply frequency plus one, it follows that the exoentropy of an entire haystack is simply its "raw entropy", because the implied probabilities of all masks are equal:

$$E \equiv Q \ln Z$$

But if, as is the usual case, the sweep window is smaller than the haystack, then the exoentropy becomes the Shannon entropy implied by the agnostic frequencies of the *other* subset of the haystack:

$$Y_J \equiv S \ln(Q+Z-S) - \sum_{K=J}^{J+S-1} \ln(F_E(J, H_K)+1)$$

where the "exofrequency" $F_E(J, H_K)$ is given by:

$$F_E(J, H_K) \equiv \sum_{K=0}^{Q-1} (H_K = M) - F_S(J, H_K)$$

so that

$$\Delta Y_J \equiv F_S(J, H_J) \ln(F_E(J, H_J)+1) - (F_S(J, H_J)-1) \ln(F_E(J, H_J)+2)$$
$$+ F_S(J, H_{J+S}) \ln(F_E(J, H_{J+S})+1) - (F_S(J, H_{J+S})+1) \ln F_E(J, H_{J+S})$$

$$\Delta Y_J \equiv \ln(F_E(J, H_J)+1) - (F_S(J, H_J)-1)(\ln(F_E(J, H_J)+2) - \ln(F_E(J, H_J)+1)) + F_S(J, H_{J+S})(\ln(F_E(J, H_{J+S})+1) - \ln F_E(J, H_{J+S})) - \ln F_E(J, H_{J+S})$$

$$\Delta Y_J \equiv \ln(F_E(J, H_J)+1) - (F_S(J, H_J)-1)\,\Delta ln(F_E(J, H_J)+1) + F_S(J, H_{J+S})\,\Delta ln\, F_E(J, H_{J+S}) - \ln F_E(J, H_{J+S})$$

where "Δln" denotes the "logdelta" function, which is easily cached for reuse:

$$\Delta ln(F) \equiv \ln(F+1) - \ln F, (F>0)$$

Note that there's no way that any of the terms of ΔY_J will result in (ln 0), in particalar because $F_E(J, H_{J+S})$ is at least one because this mask is known to be present in the haystack at index (J+S), which is outside the sweep window and is therefore counted in the exofrequency list.

4.2 Bivalent Entropy Transforms

These transforms have a haystack, a sweep, and also a needle as inputs. As with monovalent entropy transforms, the point is to compute some particular entropy parameter of all possible sweeps, all of which involving the entire needle as well, then sort them in order to find the most useful information. ("Bivalent" means "2 faces", which refers to the haystack and the needle.)

4.2.1. The Diventropy Transform

"Diventropy" is short for "divergent entropy", which measures how much a pair of data sets resemble one another. More precisely, it's the amount of information which would be required to losslessly encode the needle, given only Q, Z, and the mask probability distribution implied by the agnostic frequency list of the sweep window. So as the sweep window slides step by step, its changing agnostic frequency list is used to compute the Shannon entropy of the needle. The reason we use agnostic frequency is because there's no guarantee that all masks in the sweep window actually occur in the needle. The reason we use Shannon entropy instead of agnentropy is that, in the event that a strong match is found between the sweep window and the needle, it's assumed that the former already provides an accurate

approximation of the generator which gave rise to both of them, so further agnostic calibration would be of little value relative to its complexity burden. (This isn't *always* a valid approximation, so some other more complicated transform could do better.)

This is particularly useful for identifying parts of the haystack which are most like, or most unlike, the needle. A diventropy transform could also support a bandgap entrometry approach, in which objects are classified according to how similar to the needle they appear to be.

The diventropy E of a needle with mask frequencies F_N, with respect to a haystack with mask frequencies F_H, is given by:

$$E \equiv Q_N \ln(Q_H + Z) - \sum_{M=0}^{Z-1} F_N(M) \ln(F_H(M) + 1)$$

where the haystack and needle contain Q_H and Q_N masks on [0, Z-1], respectively; and where, as stated, F_H involves the entire haystack without regard to a sweep window.

Granted, it seems as though the downside of this approach is that it may be possible to find or construct a mask list *other* than the haystack which, when used as a needle, results a *lesser* diventropy than the haystack itself. (Beware the potential security ramifications of the *construction* case, for example to fool a classifier.) In a weird way, this is a reasonable result because it reflects the uncertainty in the generator model implied by the finiteness of Q_H. In other words, we're not saying that "H is unequal to H"; rather, we're saying that, based on only Q_H masks, there are better approximations of the generator which gave rise to H, than H itself. This is in turn a consequence of the agnostic assumptions given in [5].

Anyway, conceptually, the diventropy of a needle with respect a haystack is the Shannon entropy of the former as measured in terms of the *agnostic* frequency list of the latter. This similar to the Kullback-Leibler divergence[9], but lacks the singularities which can occur if a mask in the needle has frequency zero in the haystack (due to a lack of agnosticism).

Let's be clear about this: in the act of selecting Z, we're implying that *all* masks on [0, Z-1] are *possible* in *both* the haystack *and* the needle, in light of whatever scant prior knowledge we might have of the generator. (If the probabilities of some masks are zero in both, then a simple reassigment can "densify" the mask list, at the cost of some additional information required for invertibility. Fortunately, it's often possible to clip the range to minimum and maximum possible values, which requires much less information and thus introduces less error into the metric.) This doesn't mean that all such masks *occur*, however. The Kullback-Leibler approach presupposes that any mask which doesn't occur in the haystack *can't* occur in the needle. This makes sense in the limit of infinite haystacks, but we live in no such world, which is why it explodes with increasing probability as we examine progressively smaller haystacks. (Technically, it doesn't *explode*; it just isn't *defined*. Quoting Wikipedia: "The Kullback–Leibler divergence is defined only if [zero haystack probability] implies [zero needle probability]." Same difference.) Agnostic frequency has its own problems, in particular, the number of masks required for before a normalized agnostic frequency list accurately approximates the generator; we call this "agnostic drag". But, provided that Z is actually honest, we can do no better in practice than to start with the assumption that all masks have occurred exactly once. (In *theory,* we *can* do better, as explained in [5] -- "namely that the frequency of all masks is initially (1/Z)" -- but it makes little practical sense.) Now, if Z is unbounded, then we enter the uncharted territory of superagnentropy, as discussed in the same; fortunately, this isn't likely to be a practical concern.

Now, when a sweep window is involved, we replace $F_H(M)$ with $F_S(J, M)$, which yields

$$Y_J \equiv Q_N \ln(S+Z) - \sum_{M=0}^{Z-1} F_N(M) \ln(F_S(J,M)+1)$$

where S, as usual, is the sweep. Finally, this implies that

$$\Delta Y_J \equiv F_N(H_J)(\ln(F_S(J,H_J)+1) - \ln F_S(J,H_J)) + F_N(H_{J+S})(\ln(F_S(J+S,H_{J+S})+1) - \ln(F_S(J,H_{J+S}))+2)$$

$$\Delta Y_J \equiv F_N(H_J)\,\Delta ln\, F_S(J,H_J) - F_N(H_{J+S})\,\Delta ln(F_S(J+S,H_{J+S})+1)$$

which is the diventropy delta due to sliding the sweep window from index J to (J+1).

The diventropy transform seems to be a viable means of searching a large data set for "something that looks like" a smaller one. Intuitively, however, we should be able to achieve more accuracy by taking a weighted average of the diventropy of the needle with respect to the haystack, *and* visa versa.

4.2.2. Why Diventropy is Tough to Beat

Diventropy treats all sweeps windows identically, even if we consider the same sweep applied to many different mask lists, as is the case in a file system search for a particular needle. The reason is that all sweep windows need to compete, in effect, to compress the needle by as much as possible. Those which best approximate the generator should be best able to do so, within the limits of agnostic drag, assuming that there was in fact some common generator which gave rise to both the needle and some particular sweep window. (We want (Q>>Z) for an accurate ranking of results.)

In practice, this seems to work quite well. For example, I have an album consisting of almost 3000 photos, all of which in TARGA format. This is an uncompressed format involving, in this case, 24-bit pixels having 8 bits each of blue, green, and red. The photos vary in size by a factor of 10 or so, and were acquired with a variety of scanners and cameras. The subject matter is widely variable, as one would naturally expect of an album spanning many years. There are people, animals, landscapes, vehicles, buildings, food, machines, works of art, and more. As a test, I used Agnentro Find to search for a photo of my late great cat, Scurry. (By the way, it employs "divcompressivity", which is the normalized equivalent of diventropy that we'll discuss later.) The photo has been inserted here in a somewhat lossfully compressed JPEG format:

As expected, the highest ranking result was the image itself. The second highest was a shot taken with minutes of the first, which in fact I'd long forgotten. It, too, had been scanned from a physical photo, resulting in *entirely* different data on the pixel level. Note the change in scale (and paw position):

And the third highest was a photo of Purrsy, another charming family cat who passed away some years ago. After that, the images were not generally of cats.

Granted, there are several other photos of Scurry in the album, so it's likely that Purrsy ended up at #3 because Agnentro Find was zeroing in on his similar fur statistics, or perhaps the plush carpet beneath him. I did try a few other searches, and discovered that smooth, distinctly colored objects such as gray skyscrapers seem to produce the most consistent results in the top ranks. For example, a photo of a dense urban area was selected as the needle. Agnentro Find then turned up some other shots of the same city, as well as similar areas in other cities. To be sure, this is nothing to compete with a human or a thoroughly trained deep learning system. But the point here is to search *rapidly* and without *any* training. Given those constraints, it's frankly surprising how robust diventropy turns out to be.

Bear in mind, Agnentro Find knows nothing of pixels, chromatic components, or even 2D data structures, let alone cats. It could, with some heavy modification, be morphed into a powerful photo search utility. (I

would probably start by sorting the pixels in a photo by their Hilbert coordinates, then treating the result as a 1D mask list. This is a cheap trick to enhance accuracy without the overhead of a bona fide 2D diventropy transform.)

Again, it's reasonable to assume that by somehow combining the diventropy of the needle with respect to the haystack with diventropy from the opposite direction, we ought to be able to produce an even higher quality metric.

I personally put tremendous effort into this quest. But the results never beat plain old diventropy for quality, and ended up being much slower as well. Part of the reason is that when the direction of diventropy is reversed, the objective of optimization is no longer constant, as we're now trying to compress the *sweep window* from the perspective of the *needle*, which implies that the target data set changes from one base index to another and one file to another. Even attempts to normalize the opposing diventropies relative to their Shannon entropy failed. I then combined them in a manner weighted by the square root of the number of masks in the agnostic frequency list (for the sake of approximate proportionality to standard deviation), to no avail. But nevermind all that. Suffice to say, I've concluded that the unpredictability introduced by reversing the direction just overwhelms numerical stability, resulting in a persistently inferior metric. Perhaps someone else can improve upon the situation.

Meanwhile, diventropy is the only worthwhile bivalent entropy transform at out disposal.

5. Normalized Entropy Metrics

In the practical use of entropy transforms, there often arises the need to compare entropy metrics across various sweeps (as in, various *sizes* of sweep *windows*). For instance, if sweep windows X and Y are both remarkable in light of the entropy they contain, then which is more "interesting"? Perhaps Y has higher entropy, so in that sense it's more interesting, but maybe that's unremarkable because it's also *larger* than X. A reasonable solution to this conundrum is the use of "normalized" entropy metrics.

Normalization in its simplest form involves dividing a particular entropy metric by (Q ln Z), which as explained in [5] is its raw entropy, that is, the amount of information (in nats) required to store the mask list in base Z if all masks are equally likely.

An entropy metric is "compressive" if it results in an amount of information less than (Q ln Z), "expansive" if it results in more information, or "conservative" if it results in precisely the same amount. Due to these 3 distinct possibilities, we unfortunately can't always divide by the raw entropy in order to normalize a particular entropy metric, as we'll see in the following sections.

5.1. Dyspoissonism

As explained in [6], the dyspoissonism D of a mask list with logfreedom L and raw entropy (Q ln Z) is given by:

$$D \equiv 1 - \frac{L}{Q \ln Z}$$

recalling that we always assume (Z>1). In this case, normalization is straightforward because the numerator can never so much as equal the denominator. This is because the former excludes the information content of the population list. Without that list, one couldn't create an invertable code. (This is neverless a minor omission in comparison to Shannon entropy, which excludes the information content of the *frequency* list.)

Like logfreedom, dyspoissonism is a *randomness* metric which for most practical purposes can be conveniently misappropriated as an *entropy* metric: the lower the dyspoissonism, the more random the mask list. (And hence the more its population list resembles what a Poisson distribution would imply -- not that Poisson distributions imply maximum randomness, which is a popular misconception of an analog object which is merely asymptotically accurate.) Randomness and entropy are substantially similar, except that the entropy of permutations is maximal whereas their randomness is not. For example, {0, 1, 2, 3, 4, 5} has *more* entropy but actually *less* randomness

than {0, 1, 2, 3, 4, 4}. The reason is that an unbiased random number generator with (Z=6) would be more likely to generate the latter, even though it contains *less* information from an agnostic perspective. Insofaras security is concerned, it's more important to behave with maximum *randomness* than maximum *entropy*, if for no other reason than that systems exhibiting maximum entropy probably involve a highly ordered process buried within them (ironically). Apparently the designers of the AES encryption standard, for example, didn't understand this decades ago, which is certainly related to the inherent weaknesses which have since come to light. (Fortunately it remains "strong enough" for the moment, in our classical computing world.)

So the notion that more entropy means less order needs some rethinking. To put a finer point on it, the Kolmogorov complexity[10] of a counter is remarkably small, and yet its output has maximum Shannon entropy, maximum agnentropy, and *almost* maximum logfreedom.

To conclude, dyspoissonism is useful because it allows us to compare the randomness (entropy, roughly) of various mask lists of different Q and Z.

5.2. Shannonism

After the fashion of dyspoissonism, "shannonism" S of a mask list with Shannon entropy E and raw entropy (Q ln Z) is given by:

$$S \equiv 1 - \frac{E}{Q \ln Z}$$

Given some fixed Q and Z, just as mask lists with greater randomness have lesser dyspoissonism, lists with greater Shannon entropy have lesser shannonism. And likewise, shannonism can be used to compare the entropies of disparate mask lists in a normalized manner.

5.3. Compressivity

Similarly, lists with greater agnentropy have lesser "compressivity". But compressivity fundamentally differs from dyspoissonism and shannonism

because unlike logfreedom and Shannon entropy, respectively, agnentropy isn't always compressive. Therefore we define compressivity C of a mask list with agnentropy E and raw entropy (Q ln Z) by the following "compressivity formula":

$$C \equiv 1 - \frac{E}{2(Q \ln Z)}, (E \leq (Q \ln Z))$$

$$C \equiv \frac{Q \ln Z}{2E}, ((Q \ln Z) < E)$$

such that C is differentiable on (0, 1) and continuous on [0, 1]. This expression can be tricky to compute with interval math, as both partitions may apply to subsets of the intervals in question, in which case piecewise computation would be necessary.

Of the normalization methods presented thus far, compressivity is the most accurate means of comparing the relative entropies of disparate mask lists, subject to agnostic drag, for the same reasons that make agnentropy itself more accurate.

5.4. Exocompressivity

Compressivity is to agnentropy as exocompressivity is to exoentropy. If we replace the agnentropy E in the compressivity formula with exoentropy, then the output C is the corresponding exocompressivity. Note that the exocompressivity of an entire haystack is always (1/2) because by definition the exoentropy thereof is just its raw entropy. Thus, like exoentropy, exocompressivity is only useful in the context of sweep windows.

5.5. Exoelasticity

Exoelasticity is an inherently normalized entropy metric, without any unnormalized counterpart. It outperforms every other entropy metric in the SETI signal injection simulation ("setidemo") included in the Agnentro toolkit. The exoelasticity of a haystack is just its Shannon entropy divided by its raw entropy, which is by definition just its shannonism. So, like

exocompressivity, exoelasticity is only meaningful in a sweep window context:

$$Y_J \equiv \frac{Y_{SJ}}{Y_{EJ}}$$

meaning that the exoelasticity Y_J of a sweep window based at index J is just the ratio of its Shannon entropy Y_{SJ} to its exoentropy Y_{EJ}. We could expand the numerator and denominator; unfortunately they don't simplify, although there are some similar log terms.

For maximum speed, it's best to keep a separate accounting of Y_{SJ} and Y_{EJ}, so that their respective deltas can be efficiently computed as previously shown, while the sweep window slides along.

The downside of exoelasticity is that it's about 10 times as computationally expensive as exocompressivity, which is about twice as expensive as Shannon entropy, which in turn is about twice as expensive as agnentropy. This is largely on account of the divide operation. Nevertheless, if sufficient computing power is available, it provides excellent sensitivity in certain cases. It's somewhat unclear what distinguishes those cases, but overall it appears as though it works best on highly anomalous but very short pulses; whereas exocompressivity works best on slightly anomalous but longer pulses. Exoelasticity could be just the thing for laser SETI!

5.6. Divcompressivity

Compressivity is to agnentropy as divcompressivity is to diventropy. If we replace the agnentropy E in the compressivity formula with the diventropy (of a needle with respect to some haystack or sweep window), then the output C is the corresponding divcompressivity. Agnentro Find uses divcompressivity to great effect, in order to deliver "similar" files rapidly, as in the example previously presented.

6. Preprocessing Hacks

Getting the most out of entropy metrics, normalized or not, usually requires some preprocessing techniques. Those presented here are of O(Q) complexity, although more elaborate preprocessing could facilite higher dimensional searches, such as video search, without the burden of neural network training or expensive discrete transforms and convolutions. As always, it's a tradeoff among accuracy, energy, and latency.

6.1. Quantization and Entropy Contrast Optimization

Ostensibly, none of the entropy metrics presented herein apply to floating-point numbers (floats) We could, in theory, consider each such number as its own mask, and measure total set entropy accordingly. However, considering that there are billions of 32-bit floats and about the square as many 64-bit floats, such a task would generally require unweildy amounts of data. A better option is "quantization".

In its simplest form, quantization involves mapping a range of floats on the interval [X, Y] to a range of masks on [0, Z-1]. It's easy enough to discover X and Y by inspection, although in general they should be the same for the entire data set under analysis, as otherwise we would end up with a distorted concept of the relative entropies of various files. The trick, however, is to select an optimal value of Z.

On the one extreme, if we set Z to its minimum allowed value of 2, then we would destroy the richness of the data set, collapsing all floats into a single bit. Such severe degradation would probably leave us with unacceptably large uncertainty.

On the other extreme, we could set Z just high enough to ensure that different floats always map to different masks. This would preserve the ordering information embedded in all of the floats. On the other hand, if we end up with a huge set of masks, none of which occurring more than once, then every subset would have maximum entropy, so nothing would stand out from the crowd.

What we need to do here is select a value of Z which optimizes the "entropy

contrast" of our search results. This refers to the ratio of the maximum entropy on the rank list to the minimum entropy on the same list. (The "rank list" is the "high score" list -- just like in sports or video games. For its part, a "score" could be greatest when entropy is least, or visa vera; it all depends on the ranking rules, which are downstream of any entropy transforms.) To first approximation, we could discover an optimal value of Z via binary search on the interval [2, Q]. (Q is, in most practical cases, an upper bound for Z, although it's possible that much larger values would be required, depending on the "lumpiness" of the distribution of the particular floats involved.) This means we can potentially identify the most "interesting" sweep windows of a set of Q floats in O(Q ln Q) time, as compared to O(Q) time with integers. This is ostensibly sufficient time to do a full blown discrete transform such as a wavelet transform, but in fact it's nowhere near sufficient in the limit of large Q, considering that such transforms suffer from poor cache efficiency, unlike entropy transforms generally. The point here is that we might well be able to accomplish for floats what we can demonstrably accomplish for integers (and cats!): rapid identification of interesting or relevant signals without the burdern of complicated transforms or neural networks.

6.2. Channelized Deltafication

"Deltafication" is simply the discrete equivalent of differentiation in calculus: if we have the mask list {A, B, C, D}, then its "first delta" is {A, B-A, C-B, D-C}. Its 2nd delta is {A, B-2A, C-2B+A, D-2C+B}, and so on. Deltafication very often increases entropy contrast without modifying Z, simply because the first deltas of many real world data sets contain less entropy than the sets themselves. For example, the first delta of an uncompressed sound file is likely to compress better than the file itself. Which brings us to channelization.

Sound files typically have more than one speaker (for instance, left and right in the case of stereo). Uncompressed image files tend to have more than one color component (usually blue, green, and red). We could consider, say, the pair of 16-bit sound amplitude channels as a single 32-bit number. But this practice would probably result in poor entropy contrast because everything would appear to have high entropy. A much better approach would be to

deltafy the channels separately. For example, suppose we have {8, 5, 2, 7, 4, 1}, corresponding to 3 samples from 2 channels in alternation. The "2-channelized" first delta would then be: {8, 5, 2-8, 7-5, 4-2, 1-7}, with negative numbers converted to their equivalent values modulo Z. We could iterate this indefinitely, always resulting in an invertible mask list. Pixel data would probably involve 3-channelization, unless for example it contained a fourth component, such as a zero high byte, in which case it would be best to discard that byte prior to deltafication.

All forms of deltafication suffer from one major drawback, which is that the Nth delta contains N atypical masks at the beginning of the resulting list. This is a necessary side effect of the invertibility requirement. (These masks are analogous to constants of integration in calculus.) For better entropy contrast, they could be deleted and Q adjusted appropriately. However, except in the case of a very small mask list, doing so would probably not affect the results in any meaningful way. It's important to be cognizant of this defficiency, however, for those cases in which it matters.

Agnentro Find and Agnentro Scan both support up to 3 serialized deltafications, with optional byte channelization of N-byte masks, where N is up to 4.

6.3. Densification

Sometimes, and always if (Q<Z), there is a subset of *possible* masks which have frequency zero. In theory, this reduces entropy contrast and makes computations less precise as a result of the unnecessarily large integers involved. If this occurs, then densification may help.

The densification process maps a mask M to another mask M' in such a manner that (1) the ordering of masks is preserved (which is not always required, depending on the particular task) and (2) no mask M' has frequency zero. It therefore reduces Z to its minimum possible value. As a rule, densification takes place *after* deltafication, as it makes little sense to deltafy reassigned mask values.

By way of example, consider {1, 9, 0, 1, 7}. Its dense equivalent is {1, 3, 0, 1, 2}.

A theoretical side benefit of densification is speed, on account of more efficient cache utilization. However, this must be weighed against the cost of densifying in the first place, or doing so lazily inside the entropy transform pipeline itself. (One could also just *not* densify, but still keep track of the minimally sufficient value of Z, then compensate the resulting entropy metric accordingly.)

The point, in any case, is to improve entropy contrast. First of all, *global* densification, wherein Z is reduced to some minimally sufficient value which is then applied to all mask lists under comparative analysis, is essentially useless. One reason is purely mathematical: as is evident from a study of their respective formulae, the *ordering* of various mask lists by their Shannon entropy, agnentropy, or exoentropy is unaffected by densification; only logfreedom is affected, and usually in a minor way, on account of the population of frequency zero varying from one mask list to another. So despite an improvement in entropy contrast, the rank list is the same subject to the limits of numerical precision. But another reason is just that: precision.

The logs of smaller naturals require more terms to converge. Depending on the particular implementation of the Taylor series for the log (and related functions) with interval math, it may ironically turn out that *greater* operands produce *more* accurate results, up to some limit. So global densification ends up *reducing* the precision of the final result. Both caveats apply to *local* densification as well, which otherwise might prove beneficial.

Local densification involves minimizing Z on a per-file if not per-sweep-window basis. It's important to realize that doing so distorts an entropy metric in the sense that invertibility is lost, unless we somehow account for the cost of the information required to invert the new masks back to the old ones. But nevertheless this might be a useful exercise because it's very likely that Z is some convenient round number, such as 256, instead of a true representation of the number of masks which are are actually possible. Densification is a bet that Z was chosen in this haphazard manner. The flipside, though, is that if Z

was honest to begin with, then densification might result in a catastrophic underestimate of entropy, on account of the aforementioned discarded information. It's hard to predict, therefore, whether or not it will improve search quality in any particular case.

For its part, the Agnentro toolkit does not, as of this writing, support either form of densification. The reason is that while the local variety might help, it probably wouldn't help much, as the best entropy scanning metric appears to be exoentropy, which in the typical case of a large file results in a very accurate model of the generator, which probably would not be significantly improved by densification. And, frankly, the price of *local* densification in terms of speed and pipeline complexity doesn't seem to justify its benefits, if any. Nevertheless, there may be some applications wherein the opposite is true.

6.4. Mask Overlap

The entire point of diventropy is to crudely approximate the amount of information in a needle from the perspective of a haystack (or sweep window). To be sure, the approximation is quite precise, to the extent that Z is honest and the only statistical bias in either mask list is distributional, as opposed to contextual, in nature. (This distinction was discussed extensively in [5].)

Of course, reality is rife with contextually biased information, for example the words right here in this sentence. This means that the amount of information in the world is considerably less than we would conclude based on the foregoing assumptions. In theory, we could more accurately account for this deficit by trying to find some minimal representation of the needle in terms of the haystack. For example, one could better compress this paper by translating the word "haystack" into a couple of bytes, instead of decomposing it into its constituent letters. At a higher level of analysis, perhaps there are repeated phrases which would offer even more compression, and thus an even more accurate estimate of the entropy content herein, provided that sufficient information were saved in order to ensure invertibility back to this text.

The reason we *don't* measure entropy this way is because the discovery of such minimal representations is almost always computationally intractable. That said, one could rapidly derive a fairly accurate approximation. By way of proof, the history of lossless file compression dating back to the advent of the ZIP archive format has been based on such approximately minimal representations.

If someone were to recast a compression algo such as Lempel-Ziv-Welch into an entropy transform, the result could be a major breakthrough in the field of signal analysis. However, such an approach would need to be mindful of computational complexity, because at a certain threshold, it would become more efficient (and perhaps more effective) to use established deep learning approaches instead. That said, neural networking in general appears to be overused in cases where a lighter entropy analysis approach would get more work done faster, so the climate is ripe for improvement on both fronts.

In the interim, we have a technique called "mask overlap", which is not formally part of the theory of entropy transforms, but seems to enhance entropy contrast nonetheless. It works like this:

Suppose we have a series of 3-byte pixels, say {{9, 2, 7}, {1, 0, 8}, {4, 5, 5}, {4, 3, 6}}. (Perhaps these aren't actually pixels, but rather the 3-channelized deltafication thereof. It doesn't matter insofar as this technique is concerned.) Conventionally, we would treat each triplet as its own mask. But we could also create new "virtual masks" out by shifting our "pixel window" by a byte at a time, asymptotically resulting in triple as many masks: {{9, 2, 7}, {2, 7, 1}, {7, 1, 0}, {1, 0, 8}, {0, 8, 4}, {8, 4, 5}, {4, 5, 5}}. But how would this help?

Mask overlap is essentially a hack which causes contextual bias to manifest as distributional bias, so we can use a fast computational process for detecting the latter in order to detect the former. If in fact there is no contextual bias to be found, beyond what distributional bias would imply, then overlap will merely waste time and precision. But in cases such as DNA comparison, natural language analysis, and malware analysis, where context

is of the utmost importance, it can make the difference between detection and nondetection of a salient feature.

Agnentro Find and Agnentro Scan offer overlap as a mask list geometry option.

7. Remarks

The revolution in artificial intelligence has grown explosively on the broad utility of neural networking. As a result of this tidal wave of experimental success, comparatively little effort has been dedicated to the question of whether less computationally intensive methods might be appropriate for certain problem classes traditionally recognized as the domain of such networks. Likewise, not much investigation seems to have been directed at the question of whether there might exist more efficient or even more accurate methods of signal detection, comparison, or classification based on mathematical models which could not have evolved in nature as easily as multilayer nested weighting schemes. Entropy transforms are one family of methods which go some way toward addressing both questions while demonstrating economic utility.

Bibliography

[1] http://fftw.org

[2] http://waveatom.org

[3] https://agnentropy.blogspot.com

[4] https://dyspoissonism.blogspot.com/2015/05/the-terminology-of-mask-lists.html

[5] https://vixra.org "Introduction to Agnentropy"

[6] https://dyspoissonism.blogspot.com

[7] https://en.wikipedia.org/wiki/Hilbert_curve

[8] https://dyspoissonism.blogspot.com/2015/05/the-logfreedom-formula.html

[9] https://en.wikipedia.org/wiki/Kullback-Leibler_divergence

[10] https://en.wikipedia.org/wiki/Kolmogorov_complexity