Information, Knowledge, and Uncertainty

Charles Davi

October 16, 2021

Abstract

Below we present and apply a fundamental equation of epistemology, that relates information, knowledge, and uncertainty, and apply that same equation to random variables using information theory. Also attached is software that applies the results to deep learning classification problems, with a discussion of that application included below.[1]

[1] Any code not attached can be found in my full A.I. library, available on ResearchGate. The dataset is courtesy of Harvard University, and is available here

1 Introduction

Italian mathematicians, including Luca Pacioli, developed a system of accounting based upon the following fundamental equation of accounting:

$$A = E + L, \tag{1}$$

where A is assets, E is the owner's equity in those assets, and L is the owner's liabilities with respect to those assets. The intuition for Equation (1) is that all money is either yours, or someone else's. Therefore, when purchasing an asset, you are using either your money (represented as equity in the asset, E), or someone else's money (represented as a liability to another, L). You cannot prove the equation beyond the observation of the tautology that all money is either yours, or someone else's. Equation (1) is an instance of the more general tautology that all things are either in a given set, or not in that set.[2]

Similarly, we introduce an equation of epistemology rooted in another instance of this tautology, that all that can be known about a given system, is either known to you, or unknown to you. Expressed symbolically, if I is the measure of all that can be known about a given system, K is the measure of your ***knowledge*** of the system, and U, ***uncertainty***, is the measure of what is unknown to you about the system, then it must the case that,

$$I = K + U. \tag{2}$$

Information theory allows us to take this simple accounting, and apply it rigorously, by calculating specific values for I, K, and U, based upon the properties of a given system.

2 Basic Application

Imagine you're given a series of boxes labelled 1 through N, exactly one of which contains a pebble, though which box contains the pebble is unknown to you. In reality, you can do many things with a set of boxes, including stacking them upon each other, shuffling them about, removing the lids, etc., but for our purposes, we are interested only in the location of the pebble. As a consequence, the only actions that change the state of the system in question are those that move the pebble from one box to another. Because there are N boxes, and only

[2]Note that there are no paradoxes if you assume the universal set does not contain any sets, and contains only individual elements.

1 pebble, the system has exactly N states, each defined by a unique location of the pebble.

Because the system has N states, this treatment of the set of boxes and single pebble can be used to store exactly $\log(N)$ bits. For the same reason, at most $\log(N)$ bits are required to specify any given state of the system, with each state of the system uniquely characterized by a unique binary string of length $\log(N)$. Returning to Equation (2) above, because our inquiry is limited to the location of the pebble, it follows that the most you can ever know about the system is in this case the location of the pebble, which requires at most log(N) bits to specify. As such, it must be the case that,

$$I = \log(N).$$

If you know absolutely nothing about the possible location of the pebble, then it must be the case that your knowledge, K, is zero.

This in turn implies that, ex ante,

$$I = U = \log(N).$$

Now instead assume that you're told ex ante, from a perfectly reliable source, that the pebble is not in the first box, but you otherwise have no information about the location of the pebble. Upon receipt of this information, you are now considering a system that is equivalent to one comprised of $N - 1$ boxes and a single pebble, since you know with certainty, that the pebble is not in the first box. A system comprised of $N - 1$ boxes and a single pebble can be in exactly $N - 1$ states, which requires at most $\log(N - 1)$ bits to represent. Because the original system is, upon receipt of this information, equivalent to a system comprised of $N - 1$ boxes and a single pebble, it must be the case that your uncertainty is exactly what it would be, when considering a system of $N - 1$ boxes and a single pebble for the same purpose. This implies that, upon receipt of the information,

$$\log(N) = K + \log(N - 1),$$

and so,

$$K = \log(N) - \log(N - 1).$$

As a result, your knowledge is non-zero, but still incomplete. This example also implies the following definition:

Knowledge is information that reduces uncertainty.[3]

3 The Uncertainty of a Random Variable

Imagine you're told the pebble is equally likely to be in any one of the N boxes, which implies a uniform distribution. In the first example above, you were told nothing about the distribution ex ante, other than that the pebble is in one of the N boxes, and in that case, your knowledge was therefore zero. In this case, we have the additional information that the distribution over the possible outcomes is uniform. However, if you know nothing about the distribution, then the only distribution consistent with knowing nothing about the distribution, is a uniform distribution, since it treats all outcomes as equally likely, for any other distribution will imply a pair of probabilities, p_i and p_j, for which $p_i < p_j$. Because you have no knowledge about the system, you have no basis to assume that one probability differs from any other. As a result, in the absence of information about the system, assuming a uniform distribution is the only option, assuming there is a distribution in the first instance.[4]

This leads to the following proposition:

Proposition 3.1. *When your knowledge about a system is zero, your ex ante expectation as to the distribution of states of the system is given by the uniform distribution, and moreover, your uncertainty is given by,*

$$I = U = \log(N), \tag{3}$$

where N is the number of states of the system.

Now imagine that you have two sequences of observations, generated by two sources, $A = [aaabb]$ and $B = [aaabc]$. In both cases, the most likely modal event is a, with a probability of $\frac{3}{5}$. However, in the latter case of B, there's

[3]Note that if you receive the same message twice, the second message will not convey any knowledge, since your uncertainty will be unchanged, given receipt of the first message.

[4]Imagine instead that you're told there is no distribution, in that the distribution is unstable, and changes over time. For example, imagine that the position of the pebble is deliberately selected so that exactly this occurs over time. Even in this case, you have no reason to assume that one box is more likely to contain the pebble than any other. You simply have additional information, that over time, recording the frequency with which each box contains the pebble will not produce any stable distribution. As a result, ex ante, your expectation is that each box is equally likely to contain the pebble, producing a uniform distribution, despite knowing that the actual observed distribution cannot be a uniform distribution, since you are told beforehand that there is no stable distribution at all.

In the case where you don't know the possible states of a system, your knowledge is again zero, but your uncertainty is infinite, because the number of states is, from your perspective, unknown, resulting in an infinite set of possible states.

an additional possibility of state c, that is not present in A. In both cases, the rational prediction is a, as the most likely state, and moreover, because the probability of a is the same in both cases, the expected number of errors is the same, for any given number of observations. However, the latter case of B has an additional possibility not present in the first case, namely, state c. Intuitively, this greater multiplicity creates greater uncertainty, for the simple reason that more states are apparently possible in the case of B. As a result, B presents greater uncertainty than A, despite the most likely outcome being the same, with the same probability, in both cases.

It turns out that we can measure the uncertainty of probability distribution precisely using an equation presented by Claude Shannon in 1948 [1],[5] that provides a lower bound on the average minimum code length that can be used to encode a source, as a function of its distribution of states. Specifically,

$$H = \sum_{i=1}^{k} p_i \log(\frac{1}{p_i}), \tag{4}$$

where the distribution of the states of the source is given by $\{p_1, \ldots, p_k\}$. Said in words, if a source has a particular distribution of states, then the minimum average number of bits required to encode a single state of the source is given by Equation (4).

A lesser known result in that same paper, proves that equations of the form,

$$H = C \sum_{i=1}^{k} p_i \log(\frac{1}{p_i}),$$

where C is some constant, are the only equations that satisfy three primordial assumptions about the uncertainty of a distribution.[6] As a result, we can interpret Equation (4) as providing not only a lower bound on the average amount of information required to encode a source, but also a measure of the uncertainty of the distribution of the states of the source. Note that Equation (4) is maximized for the uniform distribution, and as such, for a system with N states, uncertainty, as measured by Equation (4), is in that case given by $\log(N)$, which is consistent with Equation (3) above.

[5] See, "A Mathematical Theory of Communication".

[6] See Section 6 of [1]: 1. The function is continuous over all possible probabilities; 2. If all the probabilities $\{p_1, \ldots, p_k\}$ are equal, then the function should increase as a function of k; 3. The total uncertainty of sequential outcomes is given by the weighted sum of the individual uncertainties, where the weight is determined by the frequency of each outcome. As a consequence, for a series of independent outcomes, the total uncertainty is given by the sum of the individual uncertainties.

As a general matter, consistent with Shannon's proof, we have the following proposition:

Proposition 3.2. *The average uncertainty of a random variable over a distribution* $\{p_1, \ldots, p_k\}$ *is given by,*

$$U = H = \sum_{i=1}^{k} p_i \log(\frac{1}{p_i}). \tag{5}$$

Note that a string generated by N independent observations of a system with k possible states, itself has k^N possible states. It follows in that case,

$$I = N \log(k).$$

Therefore, we have the following proposition:

Proposition 3.3. *Your total knowledge of a system with a distribution* $\{p_1, \ldots, p_k\}$*, given* N *observations, is given by,*

$$K = N \log(k) - NH. \tag{6}$$

4 Interpretation

We derived the value of K from tautologies, and a theorem, and so if you accept all assumptions made along the way, then you must accept Equation (6). That is, as a matter of logic, if you accept Shannon's assumptions regarding uncertainty, then you must accept Equation (4) as a measure of uncertainty. Moreover, because I is fixed, for any system, and since its value is equal to U in the case of $K = 0$, the value of I is always given by the logarithm of the number states of the system. If you have some number of independent observations of the system, then the total uncertainty is in that case given by the sum of those individual uncertainties, which is an assumption in Shannon's original paper (see footnote 6 above). And so in all cases, you must accept the value of I presented in Equation (6). K must measure your knowledge, for all that exists is either knowledge or uncertainty, and never both. And so if you accept Shannon's assumptions as they relate to uncertainty, you must accept Equation (6) as a measure of your knowledge given a distribution.

Note that this discussion relates only to the observed distribution, and has nothing to do with some purported true distribution that drives the behavior of the system in question. That is, Shannon's equation measures your uncertainty

given the observed distribution under consideration, and does not purport to relate to any underlying, ostensibly true, but unobserved distribution. This is simply the case if you're observing a possibly truly random system, since the distribution would in that case be unbounded, and so the entropy could for example, start out low, and then escalate, at any point in the future, reducing your knowledge to arbitrarily close to zero. This is simply the case, absent additional assumptions about the system.

5 Application to Prediction

We can apply Equation (6) to a problem that arises often in machine learning, which is predicting the ***classification*** of an observation: take a given observation, and predict, based upon the data within the observation, to which particular class, among some set of possible classes, the observation belongs. In this case, we'll begin with image classification, specifically, where a machine is responsible for determining what hand-written digit is displayed in an image, and the class is given by the number displayed, resulting in 10 classes, 0 through 9. In order to generate observations that lend themselves to the analysis above, given an input image, we will retrieve a set of similar images, known as a ***cluster***.[7] We can implement clustering by retrieving all images from a dataset that are sufficiently similar to the input image. This cluster will generate a distribution of classes associated with a given input image.

For example, given an image of a 1, let's assume the related cluster of images consists of the following distribution of classes: $[1, 1, 1, 7, 9]$. That is, the cluster associated with a given image of a 1, consists of, including itself, 3 images of 1's, an image of a 7, and an image of a 9. This distribution is not unrealistic, since the method in question makes use of Euclidean distance between image files, and the numbers, 1, 7, and 9, when drawn by hand, can at times all resemble each other.

In this case, the machine does not know the class of the input image, and is instead given only the distribution of classes in the cluster, which includes the class of the input image. We then predict the class of the input image by selecting the modal class as our predicted class. For example, given the same cluster vector $[1, 1, 1, 7, 9]$, the modal class is 1, the modal probability is $\frac{3}{5}$, the entropy of the distribution is $H = 1.3710$, the value of I is given by $5\log(10)$, and so $K = I - 5H = 9.7549$. In contrast, assume our observed cluster vector is $[1, 1, 1, 1, 1, 7]$. Recalculating all of these values, the modal class is 1, the modal probability is $\frac{5}{6}$, the entropy of the distribution is $H = 0.65002$, the value of I is given by $6\log(10)$, and so $K = I - 6H = 16.031$. Not surprisingly, the value

[7] See the code attached, written in Octave / Matlab, which instead classifies skin cancer lesions, using the exact same process. We describe the digit recognition hypothetical, to which the same code can be applied, e.g., using the MNIST Numerical Dataset, for simplicity.

of K is higher in the second case, because the vector is more consistent, and longer. It turns out that if you set a minimum threshold for both knowledge and the modal probability,[8] and increase that threshold, accuracy of prediction increases as a function of that threshold.[9]

Reporting a cluster of classes associated with an input is no different than making repeated observations of the same system, and recording the results.[10] Intuitively, the more consistent the observations, the less uncertainty there is with respect to the observations. More specifically, if there are a high number of possible outcomes and you make a high number of observations, then I will be high. If the resultant entropy is low, then you have a high degree of knowledge in your observations, since many outcomes could have occurred, but failed to. In contrast, if the resultant entropy is high, or you make a low number of observations, then you have a low degree of knowledge in your observations. And so this method allows you to compare, objectively, your knowledge in two potentially totally different sets of observations.[11] That this method works in practice is empirical evidence for the application of Equation (6) to probabilities, specifically, that probabilities become more reliable as a function of knowledge, and so in particular, the modal probability becomes more reliable in practice as a function of knowledge.[12]

You can develop a mechanical intuition for why this works, using combinatorics, which is that the number of ways you can be wrong depends upon the frequency distribution and the modal frequency, with the number of ways you can be wrong increasing as a function of increasingly equal frequencies (i.e., tending towards the uniform distribution) and a decreasing modal frequency. However, note that while modal frequency of course determines the expected number of prediction errors using the modal outcome, the combinatorial outcome space itself can change, even if the expected number of errors remains constant. As such, this view is consistent with Equation (4), which varies as a function of multiplicity of outcome generally. Specifically, because the algorithm does not know which entry in the cluster vector contains the input class, we can simply fix the first entry as the presumed index of the input class, and consider

[8] Knowledge can be mapped to $[0, 1]$ by simply dividing by the maximum knowledge observed for any cluster. Again, see the code attached.

[9] This is demonstrated in the code attached, which can be applied to any single object image classification task.

[10] You can perhaps debate this point, but clustering is, literally, in this case, the repeated observation of a region of Euclidean space.

[11] Note that if your observations do not consist of discrete labels, and instead, continuous data, then you can first cluster the observations, which will produce discrete categories of observations, to which this method can be applied. Alternatively, you can apply an analogous method using the equations we presented in a previous paper, "Sorting, Information, and Recursion".

[12] This implies that raising the threshold for the modal probability alone should not increase accuracy, and this is indeed the case. That is, you must also increase the threshold for knowledge in order to increase accuracy, which you can see demonstrated in the code attached.

all permutations of the cluster vector.[13] The more unique permutations there are, the greater the number of opportunities for error there are, and the number of unique permutations increases as the frequencies become more uniform.[14] Further, the lower the modal frequency, the greater the number of permutations there are that cause the wrong class to appear in the first entry, but as noted, this measure of uncertainty varies as a function of multiplicity generally. So perhaps a better mechanical intuition comes from multiplicity itself, in that by increasing the minimum threshold for knowledge, you are eliminating observations that have a large number of permutations, and therefore, it requires less data to generate a dataset that actually contains a greater portion of the possible permutations of a given observation, which will cause predictions to more closely comport with the probabilities implied by those observations.

Finally, note that we are not imputing any true underlying distribution or value to a set of observations, even if such a distribution or value exists. We are instead measuring knowledge in a set of observations, and treating those observations as all that is known. In the context of probabilities, it turns out that, empirically, increasing both knowledge and the modal probability, causes accuracy to increase. We can interpret this result as evidence for the claim that observed probabilities become more reliable as a function of increasing knowledge, regardless of whether or not there is additional data left unobserved.

As a general matter, the hypothesis to be tested is -

Observation literally carries information, and so the more information a set of observations carries, as measured by Equation (6), the more the logical implications of those observations comport to reality itself.

[13] Note there is an equivalence between permuting the labels of the elements of a vector, and holding the elements fixed, and permuting the elements, and holding the labels fixed.

[14] This is trivial to prove: consider $A = k!m!$, for $m < k$, and consider $B = \frac{m+1}{k}A = (k-1)!(m+1)! \leq A$, and apply it to the formula for unique permutations with repetition. However, this admittedly trivial fact suggests the possibility of a combinatorial theory of uncertainty that is consistent with Shannon's, where uncertainty is again the result of multiplicity.

```
%==========================================================================
%==========================================================================

%==========================================================================
%VECTORIZED IMAGE CLASSIFICATION - SKIN CANCER STAT CLUSTER PREDICTION
%==========================================================================

%==========================================================================
%==========================================================================

%COPYRIGHT CHARLES DAVI, 2021

%==========================================================================
%LOADS THE DATASET
%==========================================================================

clear
clc
pkg load image

%image directory
directory = '/Users/charlesdavi/Desktop/Datasets/Skin_Cancer/dataverse_files/1/';

%file that contains the full list of image file names
file = '/Users/charlesdavi/Desktop/Datasets/Skin_Cancer/dataverse_files/Image_ID.txt';

A = textread (file, "%s");

%file that contains the full list of corresponding classifiers
file = '/Users/charlesdavi/Desktop/Datasets/Skin_Cancer/dataverse_files/Class_ID.txt';

B = textread (file, "%f");

%file that contains the full list of patient IDs
file = '/Users/charlesdavi/Desktop/Datasets/Skin_Cancer/dataverse_files/
Patient_ID.txt';

C = textread (file, "%s");

%there are multiple similar images of the same patients, so we take uniques-----
unique_patient_IDs = unique(C);

max_num_images = size(unique_patient_IDs)

num_images = max_num_images; %number of images we select from the dataset

%generates indexes for the dataset, preventing similar duplicate images
for i = 1 : num_images

  temp_ID = unique_patient_IDs{i};

  counter = 0;
  current_ID ="HAM_ZZZZZZZ"; %this patient ID does not exist

  %iterates until we find the first image associated with the patient
  while(sum(current_ID == temp_ID) < 11) %each patient ID is 11 characters long

    counter = counter + 1;
    current_ID = C{counter};

  endwhile

  dataset_rows(i) = counter;

endfor
```

```
%-------------------------------------------------------------------------------

%loads the images and classifiers into memory
for i = 1 : num_images

  image_file = A{dataset_rows(i)};

  image_file = [directory image_file '.jpg'];
  I = imread(image_file);

  I = rgb2gray(I);

  IMG_array{i} = I;
  IMG_category(i) = B(dataset_rows(i)); %loads the classifier for each image

endfor

%==============================================================================
%EXTRACTS SHAPE INFORMATION
%==============================================================================

tic;
[final_avg_matrix final_indexes] = partition_image_vectorized_gs(IMG_array{1}); %this
is to size the partitions for the entire dataset
toc

N = size(final_avg_matrix,1);

tic;
%iterates through entire dataset
for i = 1 : num_images

  I = IMG_array{i};

  [avg_matrix] = calc_avg_color_vect(final_indexes, I, N); %this extracts shape
information

  input_vector = reshape(avg_matrix, [1 N^2]);

  input_vector(N^2+1) = IMG_category(i); %this is the hidden classifier

  dataset(i,:) = input_vector;

endfor
toc

%==============================================================================
%GENERATES CLUSTERS
%==============================================================================
tic;
N = N^2;
s = std(dataset(:,1:N)); %calculates the standard deviation of the dataset in each
dimension
s = mean(s); %takes the average standard deviation
s = s*N;

num_rows = size(dataset,1);

cluster_matrix = zeros(num_rows,num_rows);

delta = s/24; %this is the value of delta based upon experimentation

for i = 1 : num_rows

  input_vector = dataset(i,:);
  [cluster_vector diff_vector] = find_delta_cluster(input_vector, dataset, delta, N);
```

```
  cluster_matrix(i,:) = cluster_vector;

endfor
toc

%================================================================================
%GENERATES TRAINING / TESTING DATASET
%================================================================================

num_iterations = 500;

accuracy_p = [];
accuracy_c = [];
accuracy_cp = [];

num_rows = size(dataset,1);

for i = 1 : num_iterations

%permutes the dataset

%Generates a training and testing dataset----------------------------------------
num_training_rows = floor(.85*num_rows); %selects a portion of the dataset
training_rows = randperm(num_rows,num_training_rows);
testing_dataset = dataset;
testing_dataset(training_rows,:) = [];
training_dataset = dataset(training_rows,:);
num_testing_rows = size(testing_dataset,1);

%================================================================================
%CLUSTER PREDICTION STEP
%================================================================================

[predicted_class_vector confidence_vector probability_vector CP_accuracy error_vector]
= cluster_prediction(dataset, training_dataset, testing_dataset, training_rows,
cluster_matrix, N);

%================================================================================
%BENCHMARK PREDICTION STEP - NEAREST NEIGHBOR
%================================================================================

num_errors = 0;

for j = 1 : num_testing_rows

  input_vector = testing_dataset(j,:);

  [predicted_vector predicted_class prediction_row diff_vector] =
find_NN(input_vector, training_dataset,N);

  actual_class = input_vector(N+1);

  if(predicted_class != actual_class)

    num_errors = num_errors + 1;

  endif

endfor

NNaccuracy(i) = 1 - num_errors/num_testing_rows;

%================================================================================
%PROBABILITY; CONFIDENCE
%================================================================================
```

```
%probability-------------------------------------------------------------------
counter = 1;
increment = .01;
num_levels = size(0 : increment : 1,2);

for j = 0 : increment : 1

  x = find(probability_vector >= j);

  num_errors = sum(error_vector(x));

  num_predictions = size(x,2);

  if(num_predictions > 0)

    accuracy_p(i,counter) = 1 - num_errors/num_predictions;

  endif

  counter = counter + 1;

endfor

%confidence--------------------------------------------------------------------
counter = 1;
increment = .01;
num_levels = size(0 : increment : 1,2);

confidence_vector = confidence_vector/max(confidence_vector);

for j = 0 : increment : 1

  x = find(confidence_vector >= j);

  num_errors = sum(error_vector(x));

  num_predictions = size(x,2);

  if(num_predictions > 0)

    accuracy_c(i,counter) = 1 - num_errors/num_predictions;

  endif

  counter = counter + 1;

endfor

%confidence and probability----------------------------------------------------
counter = 1;
increment = .001;
num_levels = size(0 : increment : 1,2);

confidence_vector = confidence_vector/max(confidence_vector);

for j = 0 : increment : 1

  x = find(probability_vector >= j);
  y = find(confidence_vector >= j);

  temp1 = zeros(num_testing_rows,1);
  temp2 = zeros(num_testing_rows,1);

  temp1(x) = 1;
  temp2(y) = 1;
```

```
  z = temp1.*temp2;
  z = find(z == 1);
  z = z';
  num_errors = sum(error_vector(z));
  num_predictions = size(z,2);
  if(num_predictions > 0)
    accuracy_cp(i,counter) = 1 - num_errors/num_predictions;
  endif
  counter = counter + 1;
endfor
endfor %end of outer loop
plot_data_p = mean(accuracy_p);
plot_data_c = mean(accuracy_c);
plot_data_cp = mean(accuracy_cp);
figure, plot(plot_data_p)
figure, plot(plot_data_c)
figure, plot(plot_data_cp) %this is the plot generated by both thresholds
```