Entropy Maps

2021-06-17

Abstract

Entropy maps model the concept of Bernoulli maps, a probabilistic model designed to underpin the understanding and implementation of probabilistic data structures. Entropy maps are presented as a novel methodology for enabling oblivious functions through hashing mechanisms that map domain values to codomain values with prefix-free codes, thus facilitating operations without revealing plaintext data. By abstracting the probabilistic structure of Bernoulli maps, this study highlights their utility in propagating error probabilities and managing set membership queries in a manner that preserves privacy and security.

Contents

1 Entropy Maps

The basic theory behind an entropy map is to map values in the domain to values in the codomain by *hashing* to a prefix-free code in the codomain. We do not store anything related to the domain, since we are simply hashing them, and a prefix of that hash will be used as a code for a value in the codomain.

We actually allow for many different codes for each value in the codomain, so that, for instance, a code for, say, the value `a` may be `00`, `01`, `10`, and `11`. Notice that we can efficiently decode this as `a` if the hash is less than 4.

Suppose $\Pr\{f(X) = y\} = p_y$, $X \sim p_X$, then the optimally space-efficient code, assuming a uniform hash function h, is to assign prefix-free codes for y whose probability of being mapped to by h sums to p_y. In this case, the expected bit length is given by

$$\ell = -\sum_{y \in \mathcal{Y}} p_y \log_2 p_y,$$

which if we imagine sampling x from $\mathcal{X}$ with p_X and then mapping to $y = f(x)$ and observing the sequence of y's, then the expected bit length is the entropy of the sequence of y's. This is why we call it an entroy map.

If $\mathcal{X}$ is finite, then we can just imagine implicitly encoding the domain and then for each value in the domain, storing the prefix-free code that it maps to, which has an average bit length of $\ell\$$ and a total bit length of $|X|\ell$.

1.1 Rate distortion: Bernoulli maps

We can allow rate distortion, too, by failing to code for some of the elements properly. For instance, a popular choice is when one of the values, say y', is *extremely* common such that, for instance, $p_{y'} > .99$, then we can give it a prefix-free code that sums to $p_{y'}$ and then *not* code for it in the entropy map, in which case it will, for some randomly selected $x \in \mathcal{X}$, be mapped to y' with probability $p_{y'}$ (which is sufficiently large, and can be made as close to 1 as desired if we wish to trade space for accuracy), and then for the remaining

values in the domain, code for them correctly (or also allow errors on them, too, but only after trying to find correct codes for each of them).

1.1.1 Bernoulli set-indicator function

For instance, suppose we have a set-indicator function

$$1_{\mathcal{A}} : \mathcal{X} \to \{0, 1\},$$

where $\mathcal{A} \subseteq \mathcal{X}$, and $\mathcal{X}$ is a very large set (even infinite), then we may assign prefix-free codes for the codomain value 1 s.t. a priori, a random hash function hashes an element in $\mathcal{X}$ to a prefix-free code for 1 with probability ε, where ε is very small, e.g., 2^{-10}.

There are a countably infinite set of random hash functions which hash all elements in $\mathcal{A} \subseteq \mathcal{X}$ to prefix-free codes for 1 and all other elements, $\mathcal{A}' = \mathcal{X} - \mathcal{A}$, to prefix codes either for 0 or 1. If we are choosing a random hash function that satisfies this property, then it is expected that ε of the elements in $\mathcal{A}'$ will hash to a prefix-free code for 1, and the remaining $1 - \varepsilon$ will hash to a prefix-free code for 0.

For any $x \in \mathcal{X}$, we can test if $1_{\mathcal{A}}(x) = 1$ by testing if a prefix of $h(x)$ is a prefix-free code for 0 or 1, and if it is a code for 0, then we know that it is definitely not a member of $\mathcal{A}$, but if it is a code for 1, then it is a member of $\mathcal{A}$ with a false positive rate of ε and a true positive rate 1, since a randomly drawn element in $\mathcal{A}'$ will hash to 0 with probability $1 - \varepsilon$ and any element in $\mathcal{A}$ will map to 1 with probability 1 (since we explicitly chose a random hash function that hashes all of the elements in $\mathcal{A}$ to a prefix-free code for 1).

It is interesting to note that the entropy map initially frames the problem as a compression problem, but we can also think of it as a rate-distortion problem. Implicitly, in the above set-indicator function approximation, we are choosing to minimize a loss function in which false negatives are much more costly than false negatives, either because it is unlikely we will test a negative element for membership, or because false positives are not nearly as costly as false negatives, e.g., falsely thinking a rustling in the bushes is a tiger (false positive) is much less costly than failing to notice a tiger in the bushes (false negative).

In either case, we call this set-indicator approximation a Bernoulli set-indicator function, `bernoulli<(set<X>, X) -> bool>{` 1_A `}`. This is the function that is communicated, not the latent set-indicator function 1_A.

A randomly chosen random hash function that satisfies (is conditioned on) the property that it hashes all elements in $\mathcal{A}$ to a prefix-free code for 1 has the confusion matrix in Table 1.

Table 1: Conditional distribution of Bernoulli set-indicator functions given latent set-indicator function on $\mathcal{X} = \{a, b\}$

latent/observed	$1_\emptyset$	$1_{\{a\}}$	$1_{\{b\}}$	$1_{\{a,b\}}$
$1_\emptyset$	$(1-\varepsilon)^2$	$(1-\varepsilon)\varepsilon$	$(1-\varepsilon)\varepsilon$	ε^2
$1_{\{a\}}$	0	$1-\varepsilon$	0	ε
$1_{\{b\}}$	0	0	$1-\varepsilon$	ε
$1_{\{a,b\}}$	0	0	0	1

We see that the constraint of no false negatives generates a confusion matrix with a lot of zeros. If we observe `bernoulli<set<X>,X) -> bool>{`$1_{\{a\}}$`}`, then the latent set-indicator function is either $1_\emptyset$ or $1_{\{a\}}$. Since ε is very small, we can be fairly certain that the latent set-indicator function is $1_{\{a\}}$.

What is the total degrees-of-freedom for a confusion matrix of this type?

Table 2: Confusion matrix with maximum degrees-of-freedom

latent/observed	$1_\emptyset$	$1_{\{a\}}$	$1_{\{b\}}$	$1_{\{a,b\}}$
$1_\emptyset$	p_{11}	p_{12}	p_{13}	$1 - p_{11} - p_{12} - p_{13}$

latent/observed	$1_{\emptyset}$	$1_{\{a\}}$	$1_{\{b\}}$	$1_{\{a,b\}}$
$1_{\{a\}}$	p_{21}	p_{22}	p_{23}	$1-p_{21}-p_{22}-p_{23}$
$1_{\{b\}}$	p_{31}	p_{32}	p_{33}	$1-p_{31}-p_{32}-p_{33}$
$1_{\{a,b\}}$	p_{41}	p_{42}	p_{43}	$1-p_{41}-p_{42}-p_{43}$

We see that there are $4 \times (4-1) = 12$ degrees-of-freedom for the confusion matrix in Table 2. For the confusion matrix in Table 1, we have 1 degrees-of-freedom, since we have 1 parameter, ε.

The degree-of-freedom is one way to think about the complexity of a model. The more degrees-of-freedom, the more complex the model. The more complex the model, the more data we need to estimate the parameters of the model, although frequently we already know the parameters of the model, since it may have been specified as a part of the algrorithm that generated the Bernoulli approximation.

The confusion matrix in Tables 1 and 2 represent the conditional distribution of the Bernoulli set-indicator function given the latent set-indicator function, which we denote by `bernoulli<set<X>,X) -> bool>`.

1.1.2 Boolean Bernoulli as constant function

How many functions are there of type `() -> bool`? There are two, `true` and `false`. That is, there are $|\{true, false\}|^{|\{1\}|} = 2^1 = 2$ functions.

So, we can also think of Boolean values as functions of type `() -> bool`. Then, when we apply the Bernoulli model `bernoulli<() -> bool>`, we get the same result as before.

Table 3: Confusion matrix for Bernoulli model applied to Boolean values

latent/observed	`true`	`false`
`true`	p_{11}	$1-p_{11}$
`false`	$1-p_{22}$	p_{22}

This confusion matrix has a maximum of two degrees-of-freedom, since there are two parameters, p_{11} and p_{22}, since we have the constraint that the sum of the probabilities in each row is 1.

In the binary symmetric channel, $p_{11} = p_{22}$:

Table 4: Confusion matrix for Bernoulli model applied to Boolean values

latent/observed	`true`	`false`
`true`	p	$1-p$
`false`	$1-p$	p

1.1.3 Conditional distribution of latent function given observed function

Once we *have* an observation, say `bernoulli<set<X>,X) -> bool>{x}`, what does the confusion matrix tell us? Let's abstract the problem a bit so we can focus on deriving the result.

Let X and Y be random variables. Assume that $P(X = x|Y = y)$ is difficult to compute, but $P(Y = y|X = x)$ is easy. (This is the case for the confusion matrix. We know the conditional distribution of the observed set-indicator function given the latent set-indicator function, but we want to know the conditional distribution of the latent set-indicator function given the observed set-indicator function, which is not directly available.)

Suppose we are interested in $P(X = x|Y = y)$ but we only know $P(Y = y|X = x)$. Then, we can use Bayes' rule to compute $P(X = x|Y = y)$:

$$P(X = x|Y = y) = \frac{P(Y = y|X = x)P(X = x)}{P(Y = y)}$$

So, to compute $P(X = x|Y = y)$, we need to know two additional things. First, what is $P(X = x)$? This is usually a prior. If we know something about the distribution of X, then encode that information in $P(X = x)$, otherwise we can use an uninformed prior, e.g., assign a uniform probability to each possibility.

Second, what is $P(Y = y)$? This is just a normalizing constant that makes the conditional distribution of X given Y sum to 1, but it can be computed as follows:

$$P(Y = y) = \sum_{x'} P(Y = y|X = x')P(X = x')$$

If $P(X = x)$ is a uniform prior and $|X|$ is finite then $P(X = x) = 1/|X|$. Combining this with the above equation, we get:

$$P(X = x|Y = y) = \frac{P(Y = y|X = x)}{\sum_{x'} P(Y = y|X = x')}$$

So, let's replace X with the latent set-indicator function `x` and Y with the observed `bernoulli<(set<X>,X> -> bool>{y}`. Then, we can compute the conditional distribution of the latent `x` given the observed `bernoulli<(set<X>,X> -> bool>{y}` by looking at the confusion matrix in Table 2 and picking out the specific row and column of interest and then normalizing by the sum of the column.

For instance, suppose we observe $1_{\{a\}}$. Then, we want to know the conditional probability of the latent set-indicator function given the observed set-indicator. The column for $1_{\{a\}}$ is $(p_{12}, p_{22}, p_{32}, p_{42})'$. The sum of this column is $p_{12} + p_{22} + p_{32} + p_{42} = 1$. So, the conditional probability of the latent set-indicator function given the observed set-indicator is given by

$$p_{k|2} = \frac{p_{k2}}{\sum_{j=1}^{4} p_{j2}},$$

where k is the row corresponding to the latent set-indicator function of interest and we conditioning on column 2, the index for the observed set-indicator function $1_{\{a\}}$.

More generally, we have

$$p_{k|i} = \frac{p_{ki}}{\sum_{j=1}^{4} p_{ji}},$$

where k is the row corresponding to the latent set-indicator function of interest and we are conditioning on column i, the index for the observed set-indicator function. If we do this for the four possible observed set-indicator functions for Table 2 (confusion matrix with only one degree-of-freedom), we get Table 5.

Table 5: Conditional probability of latent set-indicator function given observed set-indicator function

observed/latent	$1_\emptyset$	$1_{\{a\}}$	$1_{\{b\}}$	$1_{\{a,b\}}$
$1_\emptyset$	1	0	0	0
$1_{\{a\}}$	$\varepsilon/(1+\varepsilon)$	$1/(1+\varepsilon)$	0	0
$1_{\{b\}}$	$\varepsilon/(1+\varepsilon)$	0	$1/(1+\varepsilon)$	0
$1_{\{a,b\}}$	$\varepsilon^2/(1+\varepsilon)^2$	$\varepsilon/(1+\varepsilon)^2$	$\varepsilon/(1+\varepsilon)^2$	$1/(1+\varepsilon)^2$

The conditional distribution in Table 5 is one way to think about the uncertainty of the latent set-indicator function given the observed set-indicator function. The entropy is another way to think about the uncertainty, but we will not compute it here.

We see that when we observe the empty set for a Bernoulli model in which false negatives are not possible, then we know for certain that the latent set-indicator function is $1_\emptyset$. However, when we observe $1_{\{a\}}$, we are uncertain about the latent set-indicator function. We know that it is either $1_\emptyset$ or $1_{\{a\}}$, but we do not know which one it is. Since ε is small, it is more much likely to be $1_{\{a\}}$ than $1_\emptyset$, though. A similar argument holds for the other two observed set-indicator functions.

1.2 Algorithms

The simplest algorithm is a one-level hash function evaluation, where we hash the domain values concatenated with some bit string b such that when we decode the values $h(x + b)$, $x \in \mathcal{X}$, we get a prefix-free code for $y = f(x)$.

1.2.1 Two-level hash function evaluation

The more practical solution is a two-level hash scheme. First, we hash each $x \in \mathcal{X}$ concatented with the same bit string b, same as before. However, we use this hash value to index into a hash table H at, say, index j. Now, we choose a bit string for $H[j]$ for each $x \in \mathcal{X}$ that hashes to j such that $f(x) = \text{decode}(h(x + H[j]))$.

This way, we can keep the probability $p_j = \prod_x \Pr\{f(x) = \text{decode}(h(x + H[j]))\}$ for each x that hashes to j more or less constant, independent of the size of the codomain $\mathcal{X}$, by choosing an appropriately-sized hash table H.

Since each decoding is an independent Bernoulli trial, we see that the probability that a particular x that hashes to j is decoded correctly is the number of hashes that are a prefix-free code for $f(x)$ divided by the total number of hashes (e.g., an N bit hash function has 2^N possible values).

1.2.2 Oblivious entropy maps

An *oblivious* entropy map is just an entropy map where the hash function is applied to trapdoors of $\mathcal{X}$ and the prefix-free codes for $\mathcal{Y}$ have no rythm or reason to them, e.g., a random selection of hash values for each value in $\mathcal{Y}$.