

Indefinite Order Markov Models and Their Usage in Local and Total Entropy Calculations

Leo Miao

leomiao241543903@gmail.com

Abstract

Markov chains are an important and useful tool for calculating entropy, but many methods of generating Markov models are prone to overfitting, which leads to meaningless entropy calculations. With the naïve approach of tabulating frequencies of n -grams within data, one quickly discovers they run out of enough data to construct a meaningful Markov chain for sufficiently large n s; our method instead relies on the property that one can construct an equivalent $(n-1)$ th-order Markov chain from the relative probabilities of a n th-order Markov chain's steady state. Thus, we can first calculate the $(n-1)$ th-order Markov chain, then construct a probability distribution of n th-order Markov chain probabilities (similar to the beta distribution) that takes into account the data's n -gram frequencies and satisfies the constraint that the relative probabilities of the n th-order Markov chain's steady state must match that of the $(n-1)$ th-order Markov chain and calculate its weighted average through sampling from a nonuniform distribution (to ensure the n th-order Markov chain is the trivial extension of the $(n-1)$ th order Markov chain in the degenerate case such that there are no n -grams) to get the n th-order Markov chain; this process can be repeated to construct longer and longer Markov chains, gaining increased accuracy without sacrificing the predictive power of small-order Markov chains. As of

now, the probability distributions this method relies on are only suited for two-symbol Markov chains, but hopefully in the future, they can be extended to work for poly-symbol data.

1 Introduction

In 1948, Shannon published his paper “A Mathematical Theory of Communication,” where he laid out the definition of Shannon entropy. According to the paper, the entropy of a random event is calculated by the equation

$$\sum_{i=0}^m -p(i) \ln p(i)$$

Where m is the number of possible events^[2]. Using this formula, the entropy of data can be calculated by calculating the frequencies of symbols within the data to calculate $p(i)$, using the above formula to calculate entropy, then multiplying the result by the size of the data. Unfortunately, this process will miss any patterns that might appear in the sequence.

2 Previous Methods

2.1 Basic Methods

While Shannon entropy does have its shortcomings, there are a few obvious ways to modify it to produce more accurate results. Unfortunately, as we can see, those methods also have their own problems.

2.1.1 Higher Order Entropy

In Shannon’s original paper^[2], he also suggested the use of bigrams and trigrams instead of individual symbols. Extending this concept, Shannon suggests calculating symbol

probabilities using Markov chains rather than frequency. While this does factor in more potential correlations between neighboring symbols in data, it also leaves the question of how one should design said Markov chain. If we construct our Markov chain by calculating the relative frequencies of n-grams in the data, too high of a Markov chain leads to there being not enough information within the data to do anything meaningful (for instance, choosing an order the size of your data leaves you with only one n-gram, making any relative frequency comparisons somewhat limited), while a lower order Markov model might lead to inter-data correlations being missed.

2.1.2 Moving Window

Another solution is to construct a moving window and calculate the entropy of each symbol as the entropy of a small subsequence surrounding the symbol. Unfortunately, as in the case of the naïve method the optimal window size is highly context specific, and many of the same problems occur.

2.2 Variable-Order Markov Models

Variable-Order Markov Models attempt to solve the information problem by using an evaluation function to optimize the order of a Markov chain. For instance, Rissanen in their writing^[3] proposes to first start off with the maximal order Markov chain for a piece of data, then cut off “branches” until a given gain function has been maximized. While this method and many others yield good results, it produces a Markov chain with a finite order, requires the “user” to manually set parameters, and is based off optimizing a gain function, while what we want is a more “straightforward” calculation that can construct a Markov chain of arbitrary order (even if

the order is larger than the data itself) without the risk of overfitting even without the option to manually tweak model parameters to fit the problem at hand.

3 Solution

To understand our solution, let's first look at the n th-order Markov chain:

$$P(x_i = a_0 | x_{i-1} = a_1, x_{i-2} = a_2, x_{i-3} = a_3, \dots, x_{i-n} = a_n)$$

If we choose to, we may represent it in the form of a first-order Markov chain:

$$P(X_i = A_0 | X_{i-1} = A_1)$$

Where

$$X_i = \{x_i, x_{i-1}, x_{i-2}, \dots, x_{i-n+1}\}$$

$$A_0 = \{a_0, a_1, a_2, \dots, a_{n-1}\}$$

$$X_{i-1} = \{x_{i-1}, x_{i-2}, x_{i-3}, \dots, x_{i-n}\}$$

$$A_1 = \{a_1, a_2, a_3, \dots, a_n\}$$

If we calculate the steady state probabilities of the first-order Markov chain, we will get the long-run probabilities of each sequence A . Now let's look at two such sequences:

$$U = \{u_0, u_1, u_2, \dots, u_{n-1}\}$$

$$V = \{u_0, u_1, u_2, \dots, v_{n-1}\}$$

$$R = \{v_0, u_1, u_2, \dots, u_{n-1}\}$$

$$S = \{v_0, u_1, u_2, \dots, v_{n-1}\}$$

Assuming that U and V are both binary sequences, the probability

$$P(x_i = u_0 | x_{i-1} = u_1, x_{i-2} = u_2, x_{i-3} = u_3, \dots, x_{i-n} = u_{n-1})$$

Is equal to

$$\frac{P(U) + P(V)}{P(U) + P(V) + P(R) + P(S)}$$

Effectively reducing a n th order Markov chain to a $(n-1)$ th order Markov chain.

A more efficient way to approximate the same result would be to first construct the Markov chain with the transition matrix

$$\begin{bmatrix} P(x_i = u_0 | x_{i-1} = a_1, x_{i-2} = a_2, x_{i-3} = a_3, \dots, x_{i-n} = u_n) & P(x_i = v_0 | x_{i-1} = a_1, x_{i-2} = a_2, x_{i-3} = a_3, \dots, x_{i-n} = u_n) \\ P(x_i = u_0 | x_{i-1} = a_1, x_{i-2} = a_2, x_{i-3} = a_3, \dots, x_{i-n} = v_n) & P(x_i = v_0 | x_{i-1} = a_1, x_{i-2} = a_2, x_{i-3} = a_3, \dots, x_{i-n} = v_n) \end{bmatrix}$$

By taking its steady state probabilities, you get

$$[P(x_i = u_0 | x_{i-1} = a_1, x_{i-2} = a_2, x_{i-3} = a_3, \dots, x_{i-n+1} = u_{n-1}) \quad P(x_i = v_0 | x_{i-1} = a_1, x_{i-2} = a_2, x_{i-3} = a_3, \dots, x_{i-n+1} = u_{n-1})]$$

Repeating this process with all the transition probabilities of your n th-order Markov chain, you'll be able to reduce it to a $(n-1)$ th order Markov chain.

For us, instead of “collapsing” higher order Markov chains into lower order Markov chains, we'll be doing the opposite. If we know that in a data set, there'll be A instances of sequence U , B instances of sequence V , C instances of sequence R , and D instances of sequence S , then we can deduce that if

$$x = P(x_i = u_0 | x_{i-1} = a_1, x_{i-2} = a_2, x_{i-3} = a_3, \dots, x_{i-n+1} = u_{n-1})$$

$$y = P(x_i = v_0 | x_{i-1} = a_1, x_{i-2} = a_2, x_{i-3} = a_3, \dots, x_{i-n+1} = u_{n-1})$$

$$P(E) = P(x_i = u_0 | x_{i-1} = a_1, x_{i-2} = a_2, x_{i-3} = a_3, \dots, x_{i-n} = u_n)$$

Then

$$P(P(E) = p | A = a, B = b, C = c, D = d) = \frac{p^a (1-p)^b \left(\frac{x}{y}(1-p)\right)^c \left(1 - \frac{x}{y}(1-p)\right)^d}{\int_0^1 t^a (1-t)^b \left(\frac{x}{y}(1-t)\right)^c \left(1 - \frac{x}{y}(1-t)\right)^d dt}$$

Now, you'd think that to calculate the expected value for $P(E)$, all you'd need to do is to take the expected value of the above distribution, which we can do with random sampling from a uniform distribution, but there's a problem. If we set a , b , c , and d to zero, the expected value would evaluate to one half, rather than x . Solving this problem, however, just involves sampling from a nonuniform distribution, which in this case would be the truncated exponential distribution with mean/expected value x bounded between zero and one, as to satisfy the Principle of Maximum Entropy. The proof can be done with some basic Calculus of Variations.

Proof: The truncated exponential distribution is the maximum entropy distribution for a given mean and bounds

Let us first define our entropy functional:

$$E[P(x)] = \int_0^1 -P(x) \ln P(x) dx$$

Now, let us replace $P(x)$ with

$$\bar{P}(x) = P(x) + \varepsilon \eta(x)$$

Where

$$\int_0^1 \bar{P}(x) dx = 1$$

And

$$\int_0^1 x\bar{P}(x) dx = \mu$$

We can deduce that

$$\int_0^1 P(x) dx = 1 = \int_0^1 \bar{P}(x) dx = \int_0^1 P(x) + \varepsilon \eta(x) dx = \int_0^1 P(x) dx + \varepsilon \int_0^1 \eta(x) dx$$

And

$$\int_0^1 xP(x) dx = 1 = \int_0^1 x\bar{P}(x) dx = \int_0^1 xP(x) + \varepsilon \eta(x) dx = \int_0^1 xP(x) dx + \varepsilon \int_0^1 x\eta(x) dx$$

So

$$\int_0^1 \eta(x) dx = 0$$

And

$$\int_0^1 x\eta(x) dx = 0$$

Or

$$\int_0^1 (ax + b)\eta(x)dx = 0$$

We can also see that

$$\frac{\delta E}{\delta P} = \frac{d}{d\varepsilon} \int_0^1 -\bar{P}(x) \ln \bar{P}(x) dx$$

And

$$\frac{\delta^2 E}{\delta P^2} = \frac{d}{d\varepsilon} \frac{d}{d\varepsilon} \int_0^1 -\bar{P}(x) \ln \bar{P}(x) dx$$

When epsilon is zero. Simplifying, we get

$$\begin{aligned} \frac{d}{d\varepsilon} \int_0^1 -\bar{P}(x) \ln \bar{P}(x) dx &= \frac{d}{d\varepsilon} \int_0^1 -(P(x) + \varepsilon\eta(x)) \ln(P(x) + \varepsilon\eta(x)) dx \\ &= \int_0^1 -\eta(x)(\ln(P(x) + \varepsilon\eta(x)) + 1) dx \end{aligned}$$

And

$$\frac{d}{d\varepsilon} \frac{d}{d\varepsilon} \int_0^1 -\bar{P}(x) \ln \bar{P}(x) dx = \frac{d}{d\varepsilon} \int_0^1 -\eta(x)(\ln(P(x) + \varepsilon\eta(x)) + 1) dx = \int_0^1 -\frac{\eta^2(x)}{P(x) + \varepsilon\eta(x)} dx$$

Substitution epsilon for zero gets us

$$\frac{\delta E}{\delta P} = \int_0^1 -\eta(x)(\ln P(x) + 1) dx$$

And

$$\frac{\delta^2 E}{\delta P^2} = \int_0^1 -\frac{\eta^2(x)}{P(x)} dx$$

For entropy to be maximized, we know that

$$\frac{\delta E}{\delta P} = 0$$

And

$$\frac{\delta^2 E}{\delta P^2} < 0$$

Using our previous constraints, we can see that

$$\begin{aligned} \int_0^1 -\eta(x)(\ln P(x) + 1) dx &= \int_0^1 -\eta(x) \ln P(x) dx = \int_0^1 \eta(x) \ln P(x) dx = 0 = \int_0^1 x\eta(x) dx \\ &= \int_0^1 (ax + b)\eta(x) dx \end{aligned}$$

Which means that

$$\ln P(x) = ax + b$$

And

$$P(x) = \beta e^{ax} = \frac{e^{ax}}{\int_0^1 e^{at} dt} = \frac{ae^{ax}}{e^a - 1}$$

Plugging this value into

$$\frac{\delta^2 E}{\delta P^2} = \int_0^1 -\frac{\eta^2(x)}{P(x)} dx$$

We can see that

$$\eta^2(x) > 0$$

For all x and that

$$P(x) > 0$$

For all x , meaning that

$$\frac{\delta^2 E}{\delta P^2} < 0$$

Thus completing our proof. ■

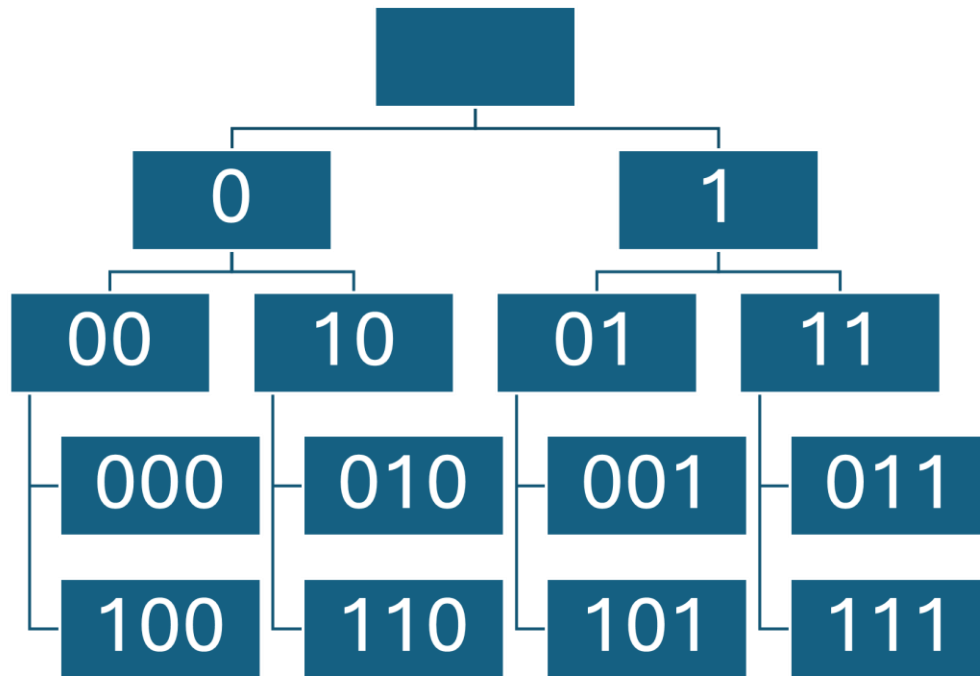
3.1 Constructing Our Markov Chain

With the methods outlined in the section above, let us construct our Markov chain.

Obviously, as we have not yet invented a computer with infinite memory, we will stop increasing

the Markov chain order when it reaches n . To begin, we represent a family of two symbol

Markov chains in the form of the (incomplete) tree below:



Taking a look at a node k layers down, we can treat said node as a $(n-k)$ th order sub-Markov chain. Therefore, we can construct this tree by first starting off with a zeroth order Markov chain, then using n -gram frequencies from the string to recursively build the tree. With this construction, if we choose to construct two different Markov models of different order with the same data, their steady state behavior and general behavior, while obviously not the same, do not contradict each other.

Calculating Entropy

Once we have our Markov chain, there are two ways that we can use it: total entropy and “point” entropy, the first of which is relatively simple to calculate. If we name our Markov chain M and our data I , the total entropy of the image amounts to the negative logarithm of $P(I|M)$. From this definition, we can derive “point” entropy. Assuming I is our binary data, and J is the same binary data but with our datum of interest flipped, point entropy is thus equal to the negative logarithm of

$$\frac{P(I|M)}{P(I|M) + P(J|M)}$$

Please note that in this case, the sum of all “point” entropies in data is not equal to the total entropy of the same data.

4 Extension to two dimensions

The extension of our model to two dimensions is based off the two-dimensional Markov chain model proposed by Ettore Fornasini and Richard A. Brualdi^[1]. To adapt their model to higher-order chains, we propose the following, where $h(i, j)$ is the probability that a horizontally-aligned Markov chain would correctly predict the symbol at (i, j) and $v(i, j)$ is the probability that a vertically-aligned Markov chain would correctly predict the symbol at (i, j) :

$$p(i, j) = a h(i, j) + (1 - a) v(i, j)$$

For accuracy, we once again propose that we use a family of values for a , one for each combination of horizontally and vertically aligned Markov chain orders.



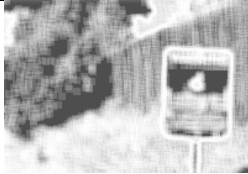




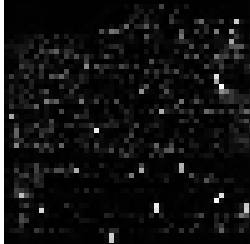
In our case, our method for calculating a for a Markov chain is relatively like calculating our transition probabilities. If we create the probability distribution

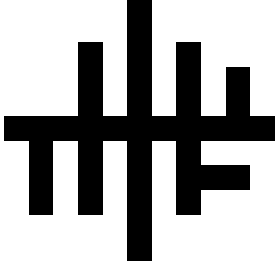
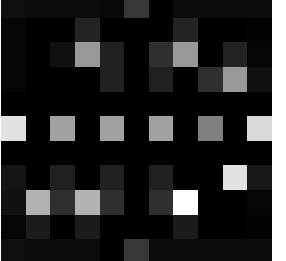

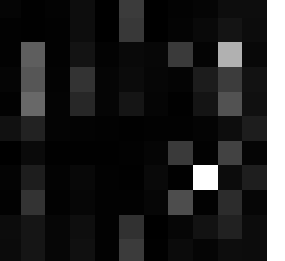
$$P(a) = \prod_{i,j=0}^{n-1,m-1} a h(i, j) + (1 - a) v(i, j)$$

Where n and m are the dimensions of our image, we can calculate a by finding the expected value of $P(a)$. Thankfully, this time, we can simply sample $P(a)$ from a uniform distribution, for the entire family of Markov chains described above.

5 Examples

Below is a comparison of our proposed method, the naïve method of calculating entropy with a Markov-like model where the probability of each pixel depends on its eight surrounding pixels, and the moving window method involving calculating entropy within a 7x7 moving window. We used a Markov chain order of ten and a sample size of 20000 for our calculations. The images are point entropy heat maps, while the entropy is total entropy.

Input	Naïve Method	Moving Window Method	Our Method
 Size: 200x190 pixels	 Entropy: 12200.442185014672	 Entropy: 14875.690657891133	 Entropy: 12090.600843742544
 Size: 50x50 pixels	 Entropy: 957.1679396585347	 Entropy: 1560.36975678541	 Entropy: 1169.460549787477

			
Size: 11x11 pixels	Entropy: 29.835322955374686	Entropy: 81.32169551611537	Entropy: 42.9516076493923

As we can observe, the entropy calculated by our method is greater than the entropy calculated by the naïve method when the image is smaller than the entropy calculated by our method, but greater than the naïve method when the image is larger, showing how our method is more consistent across varying input sizes. The moving window method, on the other hand, constantly overestimated the entropy, but that’s probably because it was never intended to be a method for calculating global entropy. The fact that the Markov order we chose for our method was higher than the one we chose for the naïve method and the moving window method also shows its ability to solve the information problem we discussed above.

6 Conclusion and Further Research

In our paper, we proposed a method for using relatively simple methods for constructing a Markov chain from data in a way that avoids overfitting or requires the user to provide parameters, then generalized this method to two dimensions. We then used this method to calculate both the total entropy of images and the entropy of specific pixels within images. From the results we have gained, we observe that the entropy calculated by our method remains sufficiently “stable” across various image sizes compared to a naïve Markov model based off an eighth order Markov model, which was prone to overfitting when given smaller images.

However, this model is still only an approximation, and does not work for grayscale or full color images. Thus, some areas for future research could include finding a way to efficiently calculate the exact transition probabilities rather than closely approximating them and extending this model to work with more than two symbols.

Works Cited

1. Fornasini, E. (1990). 2d Markov chains. *Linear Algebra and Its Applications*, 140, 101–127. [https://doi.org/10.1016/0024-3795\(90\)90224-z](https://doi.org/10.1016/0024-3795(90)90224-z)
2. Rissanen, J. (1983). A Universal Data Compression System. *IEEE Transactions on Information Theory*, 29(5), 656–664. <https://doi.org/10.1109/tit.1983.1056741>

3. Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 379–423. <https://doi.org/10.1002/j.1538-7305.1948.tb01338.x>