Decision Trees and the Dynamics of Classification

Alex Blaine

anblaine@ucdavis.edu

Department of Mathematics

Abstract

This project was to explore the inner workings of decision tree classification by performing a block entropy analysis on diagnostic binary sequences. These sequences were generated for a variety of parameters and data sets to analyze how much structure is imposed by the algorithm on the data and the dependence on the parameters. I have found that for the majority of the data sets studied and parameters chosen we have a approximately reverse sigmoid behavior to the entropy rate, with the asymptotic value approaching a constant depending on the maximum depth of the decision tree and the dataset chosen.

1. Introduction

Over the last decade Machine Learning has become the new "en vogue" topic in a variety of research fields. In everything from Speech Recognition to Cancer Screenings to Experimental Particle Physics, Machine Learning has gone from a "cute toy" of an application to the new state of the art. One of the simplest techniques in Machine Learning is a decision tree, where repeated binary cuts are made on variables from the data and the final leaves are given specific values. However, we still don't understand enough about how the algorithms behind decision trees represent information about a particular system. This is immediately clear from how we need to train our decision trees; we generally need to select a variety of maximum depths, and then use cross-validation to pick a "best" maximum depth. But cross-validation can become quite computationally costly, so this is a less than ideal solution. The goal of this project is to use block entropy analysis to glean some insight into the internal workings of how decision trees impose structure on data.

2. Background

First I'll begin by reviewing the definitions of the quantities I'll be using in my analysis. Block entropy is a generalization of Shannon entropy to sequences of length $L \geq 1$:

$$H(L) = -\sum_{i \in I} p_i^{(L)} \log p_i^{(L)}$$

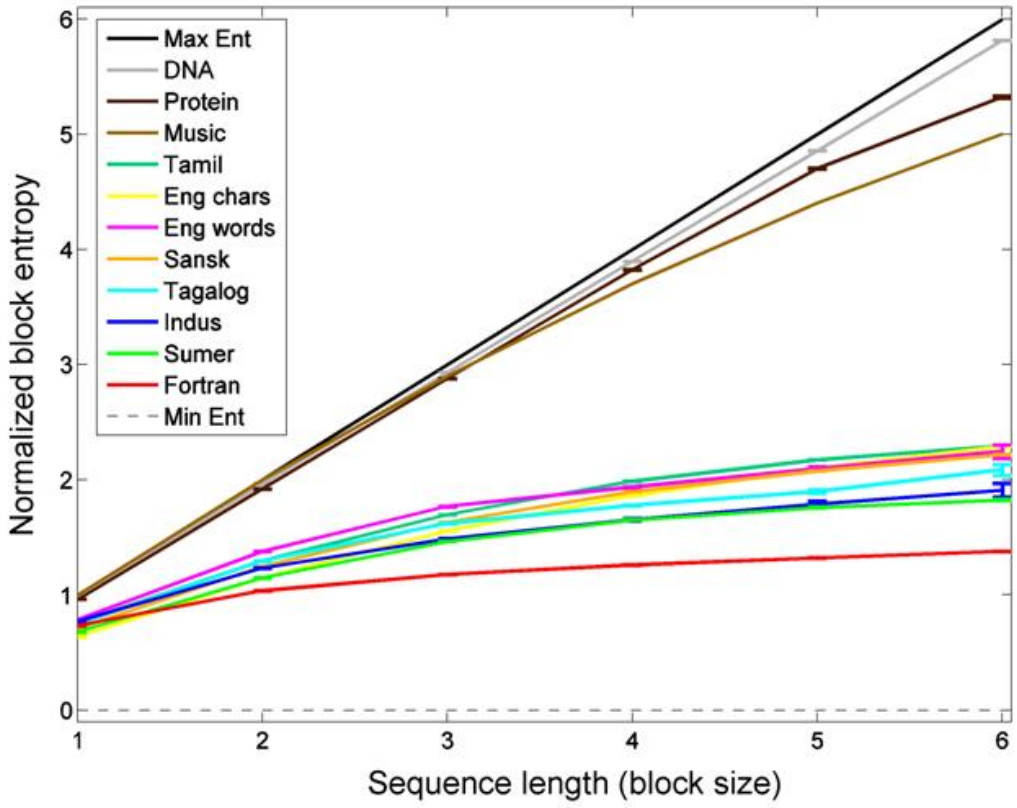
As we're analyzing sequences of bits if the sequence was uniformly random we'd expect our block entropy to have linear growth with a slope of 1. By taking differences between consecutive block entropy values and taking the limit we can define another useful quantity, the entropy rate:

$$h_\mu = \lim_{L \to \infty} H(L) - H(L-1)$$

This can be seen as the slope of our block entropy plots as $L \to \infty$. Using these two quantities we can calculate one of the most interesting quantities for my project, the total excess entropy:

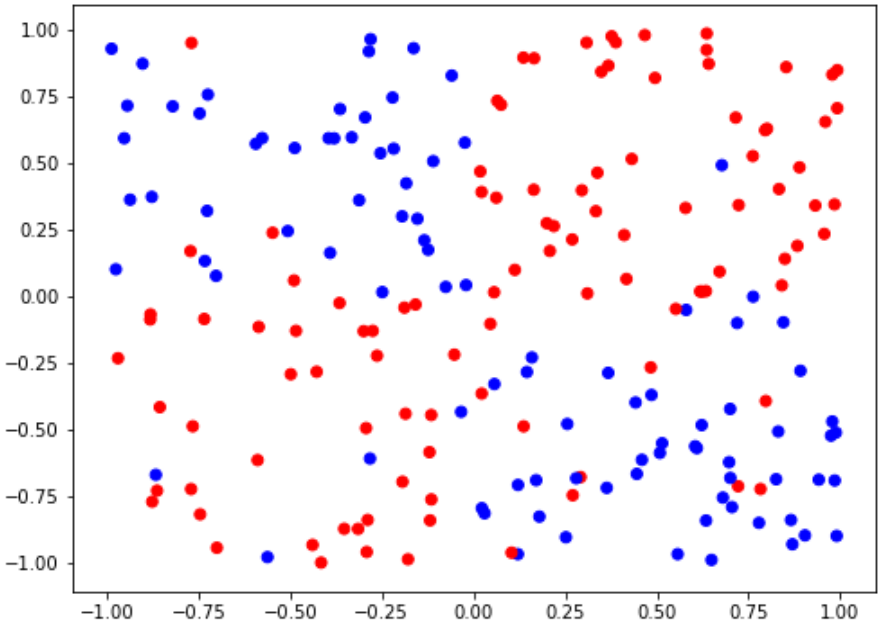
$$E = \sum_{L=1}^{\infty} H(L) - H(L-1) - h_\mu L$$

The total excess entropy can be viewed as the amount of randomness that is explained by considering longer sequences. Thus, smaller values of total excess entropy means less structure to our data (as we have less randomness that can be explained). As an example, here's a plot from a study of block entropy for natural languages[1]:

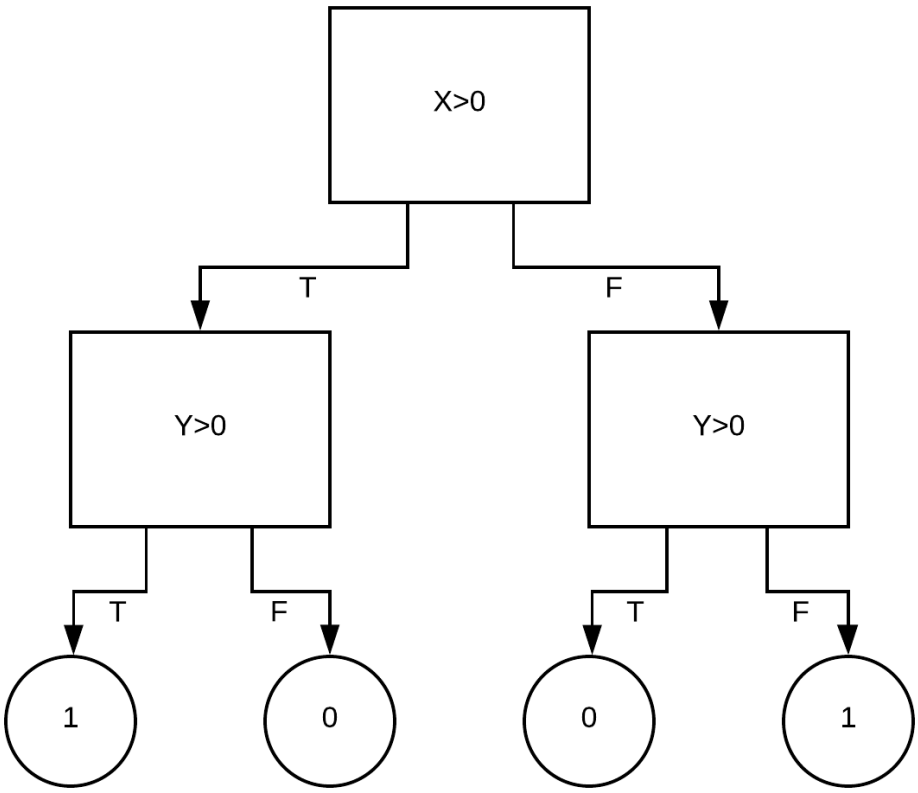


As we can see most languages exhibit fairly slow growth in block entropy, indicating that as the sequence length increases we have less randomness as some sequences become forbidden and others become common. This slow growth has a noticeable decrease after $L = 2$, something noted in the referenced paper, meaning just seeing two symbols from the sequence gives us a lot of power in predicting what comes after.

Now, lets discuss the basics of decision trees. A decision tree uses repeated binary cuts on variables to break down the domain into smaller regions. Each region is then given a particular value, and future points are given the value from the region they belong to. These values can be discrete or continuous, but we'll limit ourself to a subset of the discrete case: classification. In this case we're trying to predict a class for each point, we'll illustrate this with an example:



In this example I took 200 points generated uniformly on $[-1, 1]^2$, gave them a class defined by $z_i = \text{sgn}(x_i \cdot y_i)$. I then replaced the class of 30 points with it's inverse: $z_i' = 1 - z_i$. Clearly if we give a class based on the equation above it gets most of the points correct, which would correspond to the following decision tree:



Now, lets discuss a bit about popular algorithms for generating decision trees. In particular, lets look at the two most popular criteria for deciding which variable to split on: the Gini impurity and the Information Gain. First, the Gini impurity measures how often an element of the subset would be mislabeled if it was given a random label from collection of labels for that particular subset, which can be calculated as follows (for a set with N classes):

$$I_G(p) = 1 - \sum_{i=1}^{N} p_i^2$$

An algorithm using this criterion would seek to minimize the Gini impurity for each new split added, as this reaches zero when each side of the split contains only one class. Background on this criterion was taken from the

original 1984 monograph on the CART algorithm[2].

Next, the Information Gain criterion. This criterion seeks to maximize the purity of the child nodes, which is done by calculating the Information Gain:

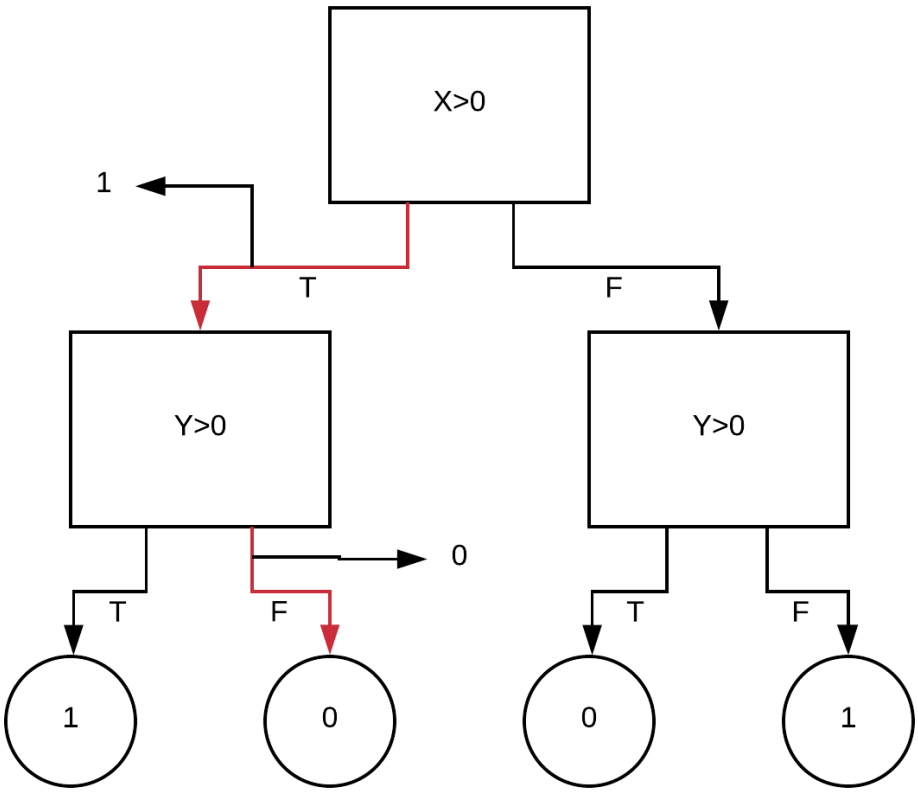
$$\overbrace{IG(T,a)}^{\text{Information Gain}} = \overbrace{H(T)}^{\text{Entropy(parent)}} - \overbrace{H(T|a)}^{\text{Weighted Sum of Entropy(Children)}}$$

$$= -\sum_{i=1}^{N} p_i \log_2 p_i - \sum_{a} p(a) \sum_{i=1}^{N} -P(i|a) \log_2 P(i|a)$$

An algorithm using this criterion would calculate the Information Gain for each possible split, then choose the maximum. Background on this criterion was taken from Wikipedia, particularly it had the nice equation form above for information gain.

3. System

Now, to be able to look inside the classification process I had to modify my decision trees to output a diagnostic bit stream. The process I used was based on the decision path for each data point, with following the left branch corresponding to a 1 and the right branch corresponding to a 0. This generated up to M bits per data point, where M is the maximum depth parameter from the decision tree algorithm. Using the example from before lets consider the point $(1, -1)$: this would output the sequence of bits $'10'$ and would be illustrated as follows.



4. Methods

For this project I considered two datasets: a credit card fraud dataset, and an adversarially generated dataset.

The credit card fraud dataset was downloaded from the machine learning competition site Kaggle, and was chosen simply so that I might have many data points to generate a long sequence.

The adversarial dataset was generated by taking 20 uniform random variables on $[-1, 1]$, taking the sign of their product, and then replacing 2 of these variables with a new uniform random variable.

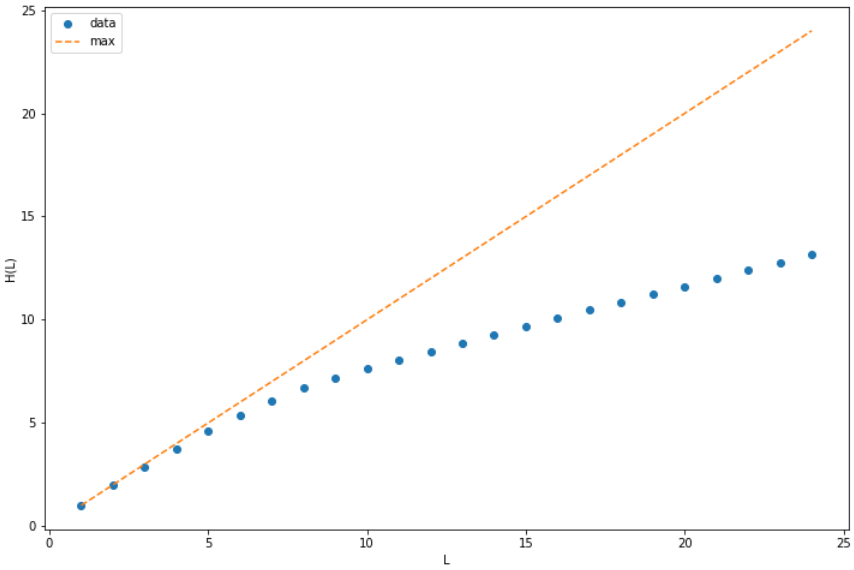
For the credit card fraud dataset I took a binary representation of the data as a baseline bit sequence to compare to.

Next, I fit a decision tree to the dataset using a variety of initial parameters: Gini impurity vs Information Gain, and a spectrum of values for max depth. The end goal here was to analyze the total excess entropy and entropy rate for both of the criteria, and then see what effect max depth had.

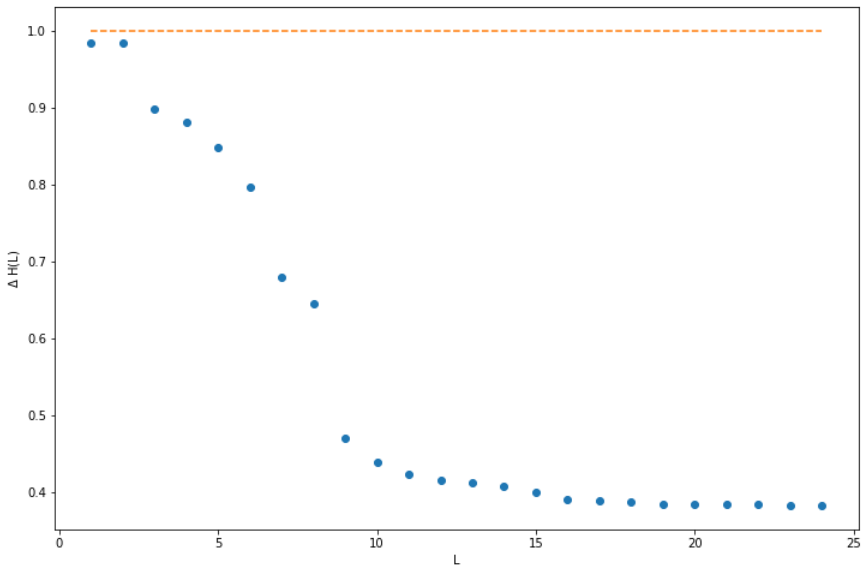
To this end, I then calculated block entropies for each of the bit sequences.

5. Results

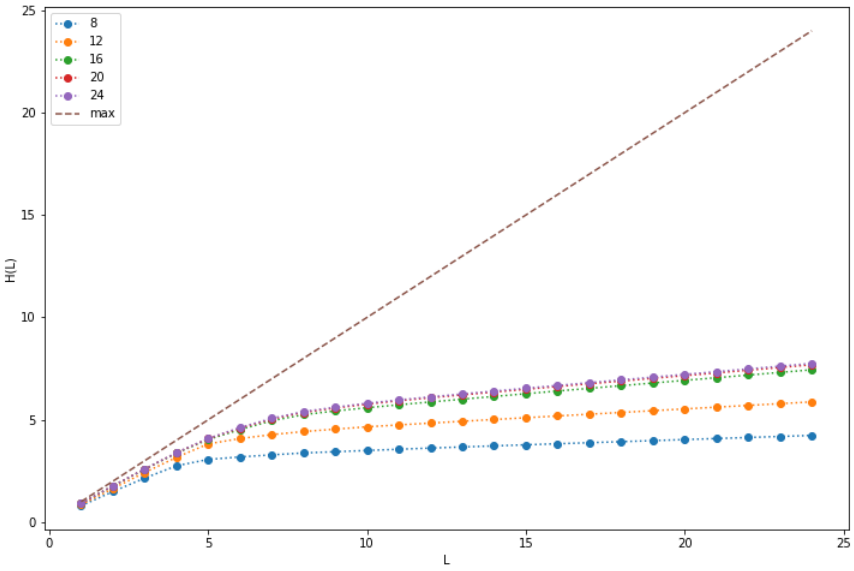
First, lets look at our block entropies for our baseline bit sequence from the binary representation of the credit card fraud dataset.



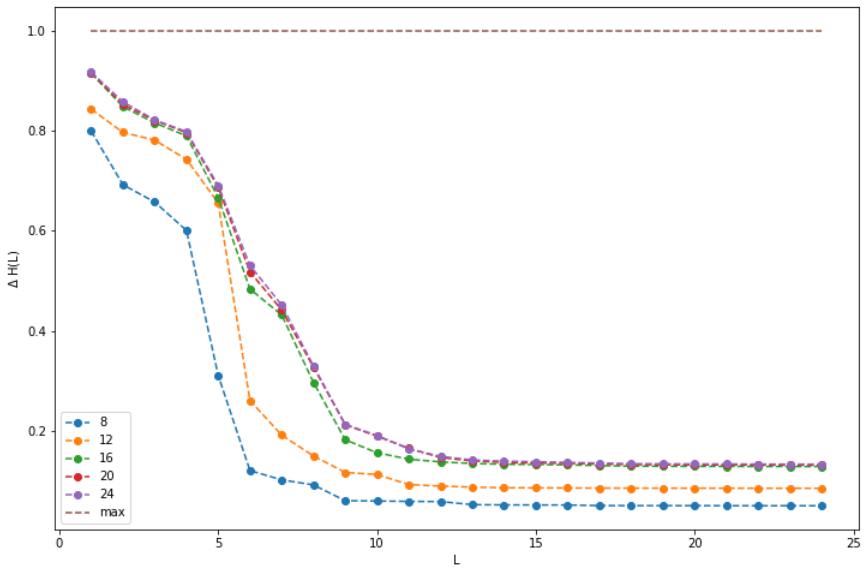
As we can see we have a rate of close to 1 for small sequences, but then it quickly decreases to a smaller constant. This smaller constant can be seen more clearly in the plot of differences:



Here we can see it quickly the rate decreases to approximately 0.382 (approximated using the last value calculated). Using this value we have an excess entropy of approximately 3.98 bits. Next, lets look at the block entropies for our bit sequences generated by decision trees using the Gini impurity criterion. First, our credit card fraud dataset using the following max depths: $L \in \{8, 12, 16, 20, 24\}$.



Here we can see similar behavior to the baseline bit sequence, but with a much smaller slope and a faster decrease in slope.



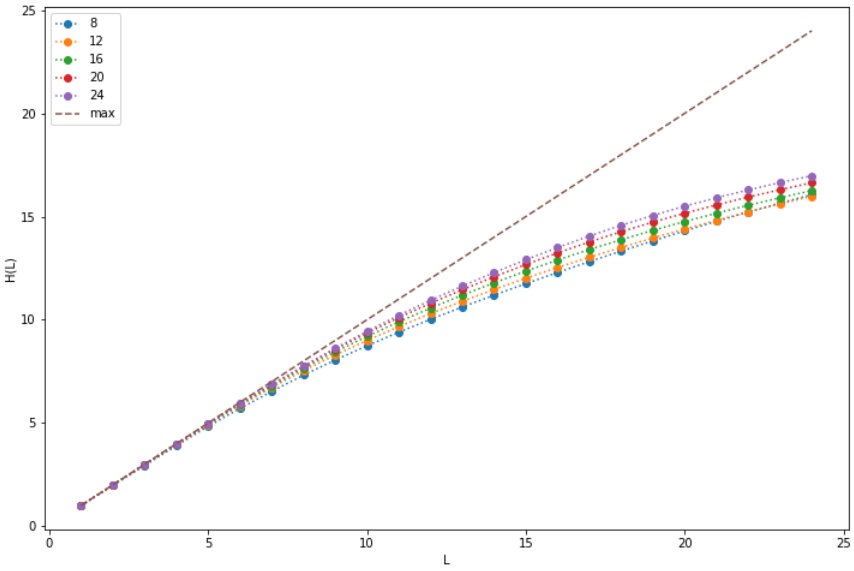
From this graph we can see that it quickly decreases then flattens out like before, so lets approximate the entropy rate using the last calculated value:

L	8	12	16	20	24
h_μ	0.0508	0.0857	0.1291	0.1318	0.1338

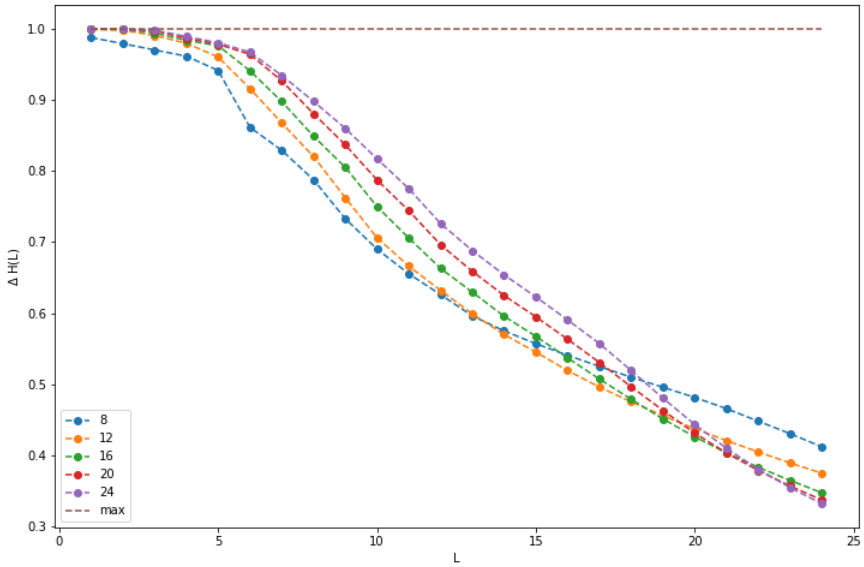
Now, using these we can calculate the total excess entropies for this case:

L	8	12	16	20	24
E	3.016	3.816	4.343	4.521	4.542

Here we can see total excess entropy increases with max depth, meaning deeper decision trees enforce more structure on the data. But, we're not seeing significantly more structure here than the baseline, with about 0.5 bits more total excess entropy for the deeper decision trees. Now, lets look at our adversarial datasets:



Now we have significantly different behavior than before, with the divergence from max taking much longer and being much slower.



Clearly we don't have the same rapid falloff behavior as before, with our rate steadily decreasing over the range studied. I couldn't study longer sequences as my code blew up in memory use for sequences longer than $L = 24$. But, we can use the last value for each of the rates as an upper bound to give us a lower bound on the excess entropy here:

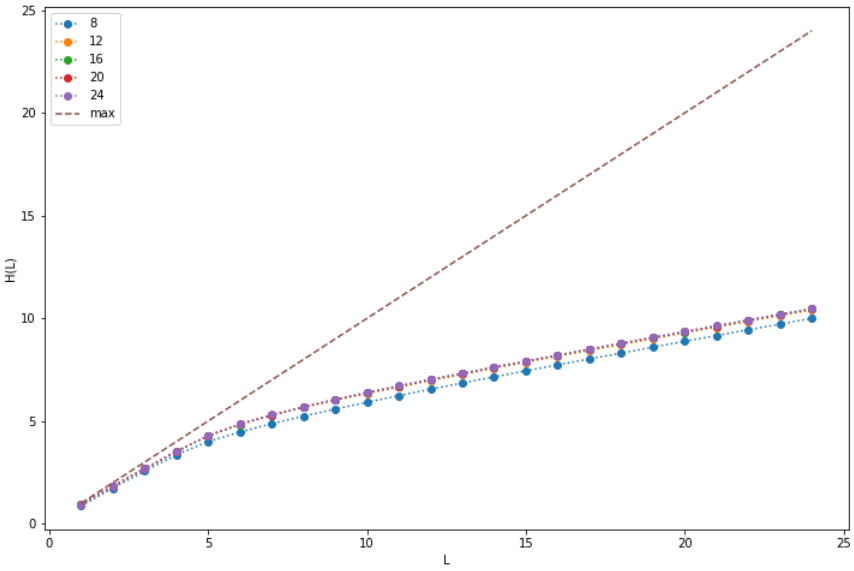
L	8	12	16	20	24
h_μ	0.4121	0.3749	0.3474	0.337	0.332

L	8	12	16	20	24
E	6.169	6.988	7.920	8.543	9.010

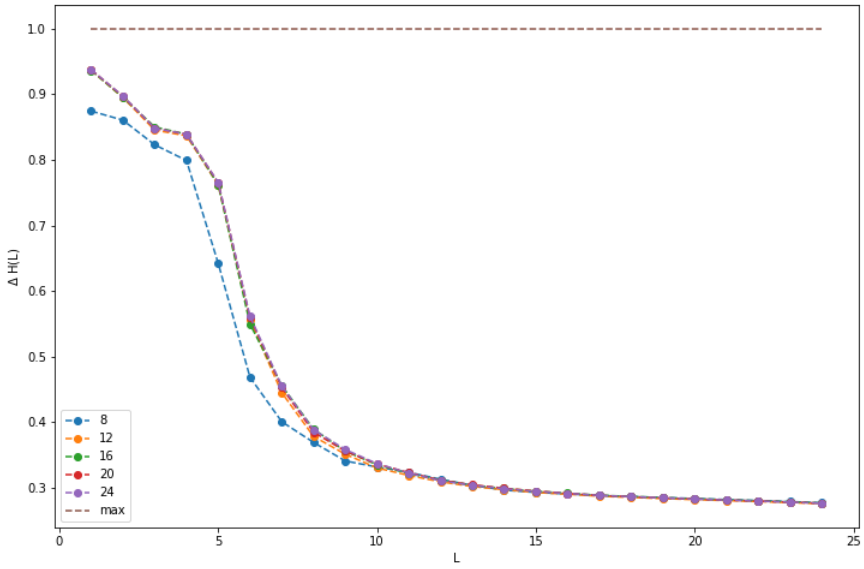
So we can see much higher total excess entropies than the previous examples.

Now, lets look at the block entropies for the decision trees using the Information Gain criterion.

We'll use the same max depths as before, first lets look at our credit card fraud dataset:



Here we see much of what we saw before with the credit card fraud dataset, although noticeably closer together.

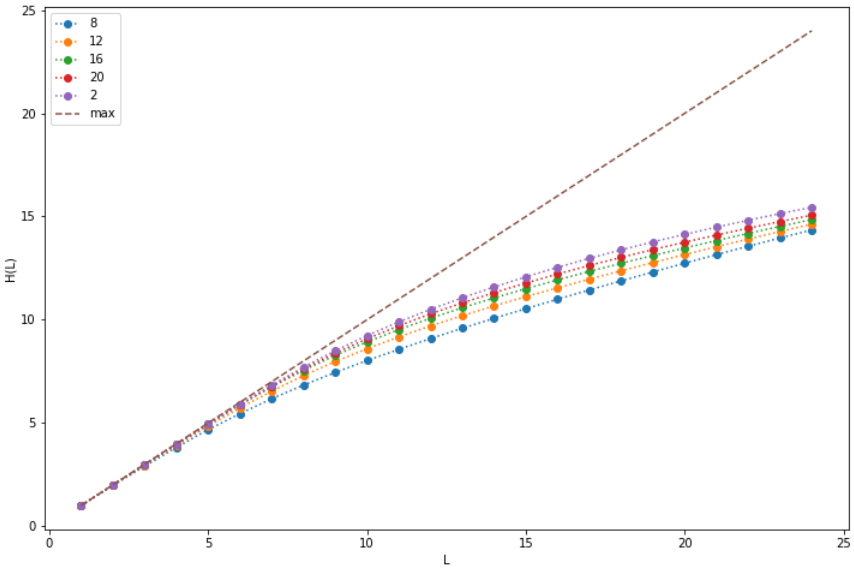


This is particularly interesting, there's much less spread to the rates than before, with them all converging to roughly the same asymptotic value. But, this time it seems to be still decreasing very slowly. So again we'll calculate our entropy rates and total excess entropy.

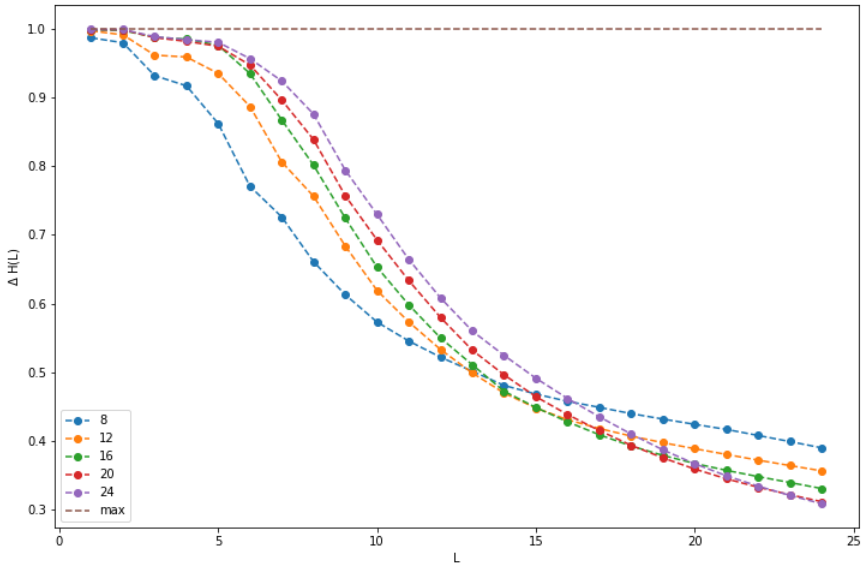
L	8	12	16	20	24
h_μ	0.2773	0.2759	0.2769	0.2765	0.2765

L	8	12	16	20	24
E	3.342	3.781	3.811	3.825	3.836

And here we find something interesting, across the board we find lower excess entropy values than the baseline binary representation, which would imply less structured data. Now, back to our adversarial dataset:



In this case we see much of what we saw before, but they do seem to be a bit more spread out.



Another plot that is qualitatively similar to before, and again we have seemingly still decreasing values so we'll have bounds instead of approximate values.

L	8	12	16	20	24
h_μ	0.3897	0.3562	0.3304	0.31131	0.3083

L	8	12	16	20	24
E	4.999	6.081	6.922	7.595	8.050

And again, we see a fairly significant increase in total excess entropy for deeper decision trees.

6. Conclusion

In conclusion, I identified two key trends: increasing total excess entropy for deeper decision trees, and lower total entropies for the Information Gain criterion over the Gini impurity criterion. This can be interpreted as deeper trees enforcing more structure on our data than shallower trees, and the Gini impurity criterion further emphasizing this. Neither result is a surprise,

as deeper decision trees can produce longer blocks, and a consequence of maximizing information gain is you minimize entropy.

Additionally, it was particularly interesting that for more realistic datasets the Gini impurity criterion generated entropy rates that increased with max depth, while the entropy rates with the Information Gain criterion generated approximately equal entropy rates. I'm not quite sure why this is the case, but looking at the adversarial dataset for longer sequences might provide some more insight.

Some areas that might be of interest for future studies could be other adversarial datasets, calculating block entropies for larger sequences in the cases that seemed to be still decreasing at a noticeable rate, expanding to other tree algorithms such as random forests, and looking at the effects of the other parameters for the decision tree algorithms. This last one would be interesting as it could provide insight necessary for the research being done in determining the optimum parameters for the algorithm without using cross-validation. If a sufficiently efficient technique to generate these parameters is developed this could significantly decrease the computing strain from cross-validation.

Appendix A. References

[1] Rajesh P.N. Rao, *Probabilistic Analysis of an Ancient Undeciphered Script*. Computer (Volume: 43, Issue: 4, April 2010)

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.