Decoding Complexity: Bifurcation Diagram Generation and Symbolic Time Series Analysis of a Logistic Map

Isak Ederlöv

April 21, 2023

1 Background and objectives

A logistic map is a discrete dynamical system that represents the population size at each time step as a function of its previous size, with a parameter that determines the rate of population growth. The map is typically represented graphically, with the population size on the vertical axis and time (or iteration number) on the horizontal axis. The plot of population size over time often exhibits complex behavior, including periodic oscillations, chaotic dynamics, and the emergence of stable and unstable fixed points. One of the main objectives of this report is to analyze the time series data in the chaotic portions of the data set and investigate if there are any patterns to uncover.

The equation for the logistic map is

$$x_{n+1} = rn(1 - x_n) \tag{1}$$

where x_n is the population at a given iteration, and r represents the rate of population growth or reproduction in a discrete dynamical system; it determines how much the population size increases at each time step based on its previous size.

1.1 Bifurcation diagram

A bifurcation diagram is a graphical representation of the behavior of a dynamical system, often used in chaos theory and nonlinear dynamics. It shows how the system's solutions or trajectories change as a parameter is varied.

In the context of a logistic map, a bifurcation diagram illustrates how the population size changes as r is varied. The x-axis of the bifurcation diagram represents different values of r, while the y-axis represents the population size or its asymptotes. By iterating the logistic map equation for different values of r and plotting the resulting population sizes or asymptotes on the bifurcation diagram, patterns and structures can emerge.

1.2 The Feigenbaum constant

the Feigenbaum constant, δ, is a mathematical constant that describes the period-doubling behavior in complex dynamical systems. In the context of the bifurcation diagram, the Feigenbaum constant quantifies the rate at which period-doubling bifurcations occur as the growth rate parameter r is varied. In this report, the bifurcation points will be found via data analysis and be used in the calculation of the Feigenbaum constant, to see how closely the calculated value matches the empirical value of 4.669.

1.3 Ordinal pattern analysis

Ordinal patterns are ways to categorize different sequences of numbers according to the "shape" of the sequence. The shape describes in what order the points in the sequence are increasing or decreasing. Ordinal pattern analysis can be used for feature extraction, reducing the dimensionality of data and also making predictions of data that seems chaotic.

1.4 Permutation and network entropy

Permutational entropy (PE) is a measure of the complexity of a time series that quantifies the degree of disorder in its permutations, in this case ordinals. In the context of a bifurcation diagram of logistic map data, PE can be used to study the dynamics of the system as it undergoes

bifurcations, where increased PE indicates either less predictability in the data overall, or a long repeating sequence (a 12-long repeating pattern could result in a low PE if the ordinal length is 3 or 4).

The general formula for the permutational entropy is

$$H = \sum_{i=1}^{N} p_i ln(p_i)$$

where p_i is the probability of observing an ordinal pattern in the data set for the given parameter.

2 Methodology

2.1 Bifurcation diagram

```
def logistic_map_asymptotes(r: float, x0: float, depth: int = 10000, y_range_tolerance: int = 3) ->
↪  list:
    peaks =  []
    ser = [x0]*depth

    for xi in range (depth-1):
        ser[xi+1] = r*ser[xi]*(1-ser[xi]) #Logistic map formula

    peaks = set([round(x, y_range_tolerance) for x in ser[-100:]])
    # implicit rounding and uniquifying

    return peaks
```

The core methodology during the development was to break the problem down into parts separate enough to simplify, yet large enough to independently solve a problem and to be generalizable to other problems.

Since the bifurcation map is a plot showing the asymptotes of a time series, the first function *logistic_map_asymptotes* was assembled and used to calculate the asymptotes for a given value of r. To build a bifurcation diagram, this function was run for 2000 linearly spaced values of r in the range of 2.5 to 4.0. The generated asymptotes were then represented vertically for the given iteration. This generated the graphic 2.

2.2 Experimental calculation of the Feigenbaum constant

Since a bifurcation is defined as a division of one or more branches, and the time series asymptotes are shown vertically in the bifurcation map, an increase in asymptote amounts for a given r constitutes a bifurcation. The index of bifurcations can thus be found by appending the *main loop* with a check that adds the current iteration index to a bifurcation index list if a bifurcation is present in the iteration according to the above criteria for a bifurcation. To counteract the impact of noise in the data on the quality of the analysis, a simple monotonicity filter was applied to the time series to make it monotonically increasing, and thus removing the peaks.

2.3 Ordinal symbol analysis

The ordinal symbol extraction was based on a simple algorithm for converting three consecutive numbers into a number called the *ordinal*, derived from the definition in *Figure* 8

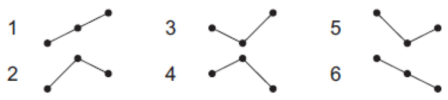


Figure 1: Comparative value combinations and their respective ordinals. (Prof. Cristina Masoller, "Introduction: From dynamical systems to complex systems", Presentation Slides, 2022)

```
def ordinal_symbol(ts: list) -> int:
    if ts[0]<ts[1]:
        symbol = 1 if ts[1]<ts[2] else (2 if ts[2]>ts[0] else 4)
    else:
        symbol = 6 if ts[1]>ts[2] else (3 if ts[2]>ts[0] else 5)
    return symbol
```

The time series of the transient-removed single-r logistic map equation was iteratively transformed, using the *ordinal_symbol* function as a mask. This generated a collection of ordinals present in the time series for $r = 3.99$.

To create a ordinal bifurcation diagram, the code used to compute the single-r ordinal distribution was refactored to compute an ordinal distribution for a parameterized r, and then iterated through a linearly spaced array of 201 values in the desired range [3.5, 4.0]. The resulting two-dimensional array was transposed to take the shape of six timeseries, one per symbol, with varying r.

```
r_values = np.linspace(3.5, 4.0, 201)
def ordinal_distribution(r):
    series = logistic_map_no_transients(r, 0.1)
    logistic_map_symbols = [ordinal_symbol(series[xi:xi+3])for xi in range(len(series)-3)]
    counts = np.bincount(logistic_map_symbols, minlength=7)[1:] #removing first position because of
    ↪  zero-indexing
    probabilities = counts / len(logistic_map_symbols)
    return(probabilities)
ordinal_probabilities = np.transpose(np.array([ordinal_distribution(r) for r in r_values]))
```

3 PE and NE

For the calculation of the permutation entropy, the following algorithm was used.

```
def permutation_entropy(r: float):
    #Generate the distribution of ordinal probabilities
    p = np.array([ordinal_probability(n, r) for n in range(1,6)])
    p = p[p != 0] #To avoid division by zero

    H = -np.sum(p * np.log(p))
    return H
```

In one of the runs, noise was added with the function *random.normal* in *numpy*. The noise had unit variance and zero mean.

For the network entropy, a similar approach was used. The difference is that a network is now constructed using *networkx* and a degree distribution probability map was used instead of a ordinal probability.

```
def network_entropy(r):
    ts = np.array(logistic_map(r, 0.1, 11000)[1000:])
    windows = np.reshape(ts, (200, 50))
    M = 200

    # With the data split into windows, now make a network of each
    # window and calculate the degree probability distribution
    # Repeat for every window and avergage
    for w in windows:
        wij = visibility_graph(w).degree()
        wij = np.transpose(np.array(wij))[1]
        s = []
        #How much variability is there in the amount of edges?
        probabilities = np.array([np.count_nonzero(wij == x)/50 for x in wij])
        s.append(sum(-probabilities*np.log(probabilities)))

    return sum([si/M for si in s])
```

4 Results

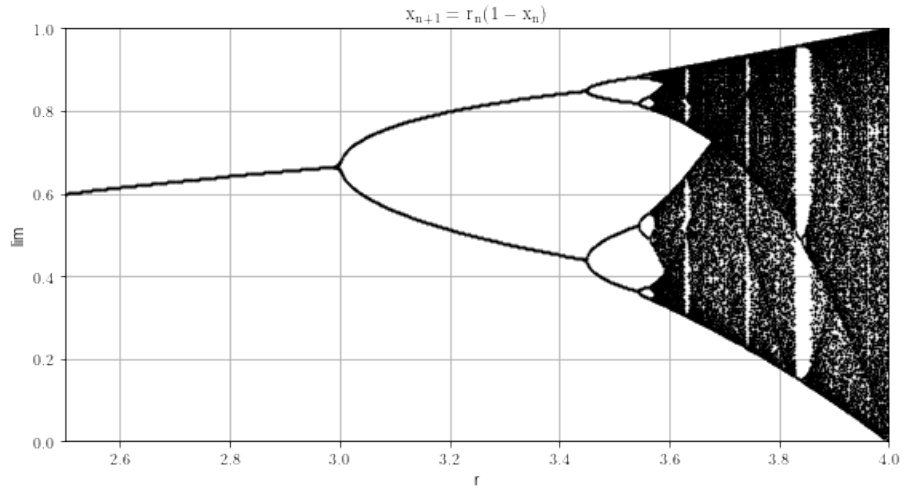


Figure 2: Bifurcation diagram of the logistic map.

The bifurcation diagram obtained from the analysis of the logistic map dynamics is shown in Figure 2. It exhibits the characteristic pattern of bifurcations, with the emergence of multiple branches as the value of r increases. The Feigenbaum constant was calculated to be $\delta = 4.721$, which is close to the known value of 4.669. These results suggest that the logistic map exhibits chaotic behavior in the analyzed portion of the data set, with the occurrence of bifurcations and the emergence of multiple branches.

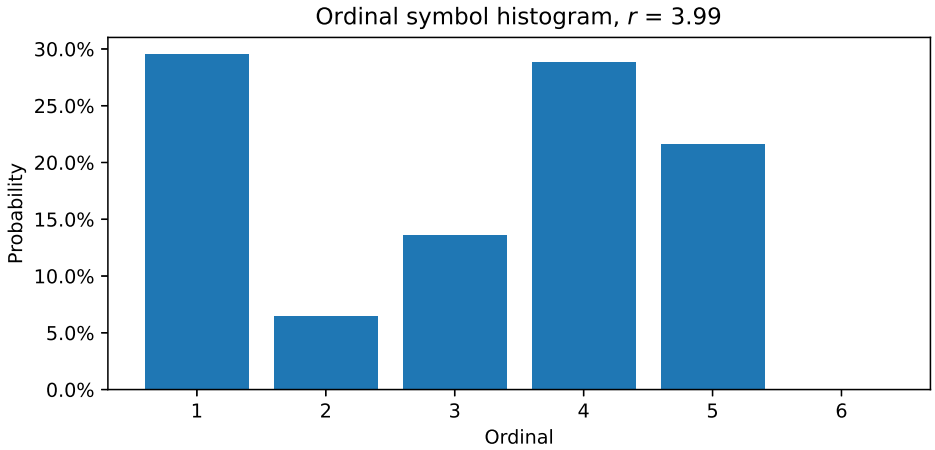


Figure 3: Histogram showing the distribution of probabilities for a growth rate of 3.99. Pattern six is completely absent.

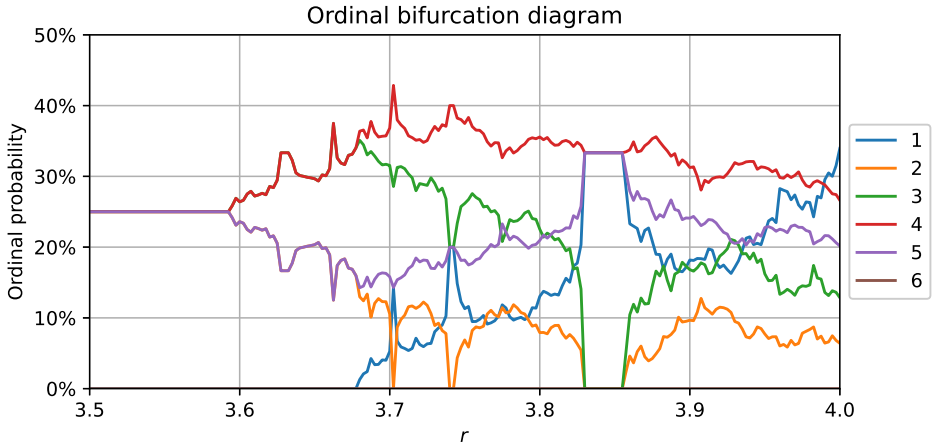


Figure 4: Graph showing, for varying growth rate values, the probability of a certain pattern anywhere in the transient-removed logistic map time series.

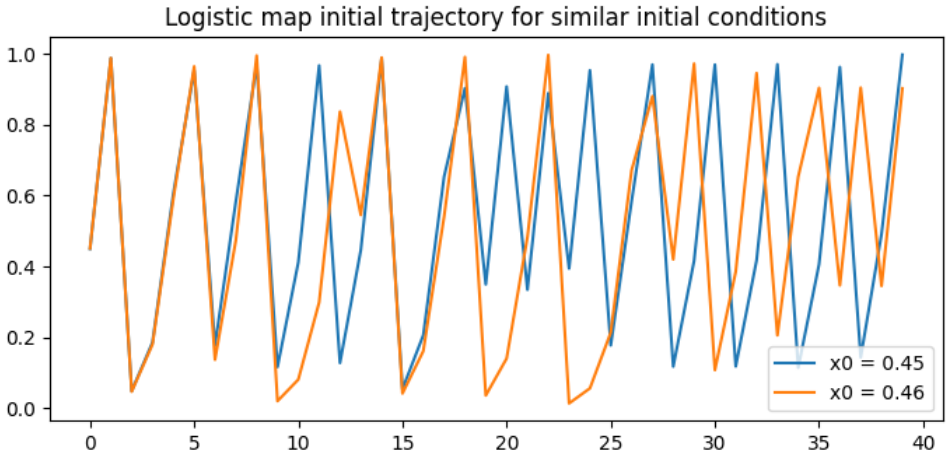


Figure 5: Graph showing the initial trajectories of the logistic map time series for two similar initial conditions. The two trajectories are initially similar but over time diverge.

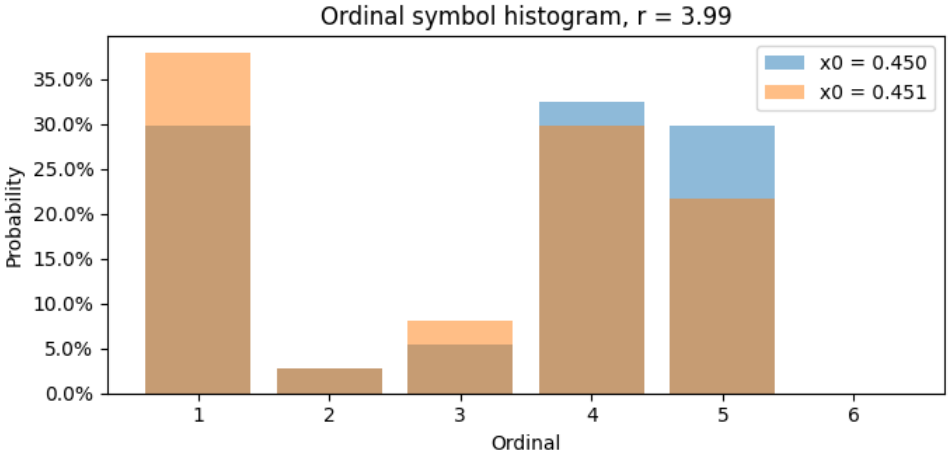
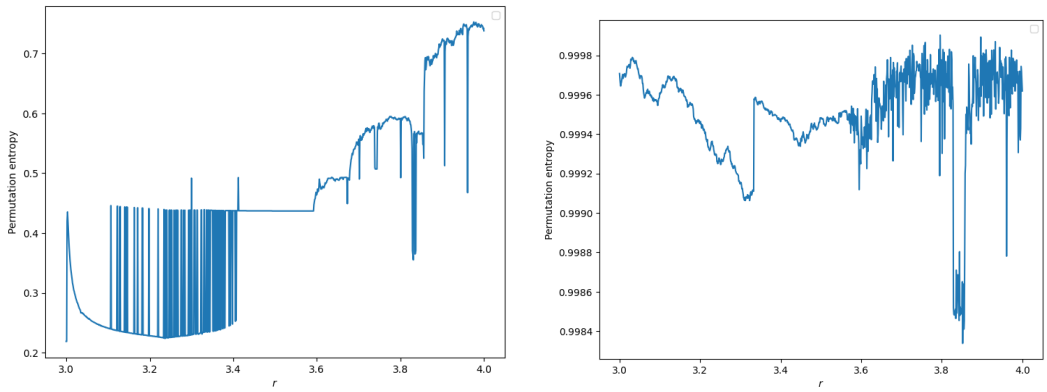


Figure 6: Graph showing the initial trajectories of the logistic map time series for two similar initial conditions. The two trajectories are initially similar but over time diverge.

In Figure 7, in the region r = [3.5, 4], it is apparent that some of the patterns present in the noiseless sample, even though there is less y-axis variation.



(a) Permutational entropy as a function of r, without noise.

(b) Permutational entropy as a function of r, with noise.

Figure 7: Permutational entropy with and without noise.

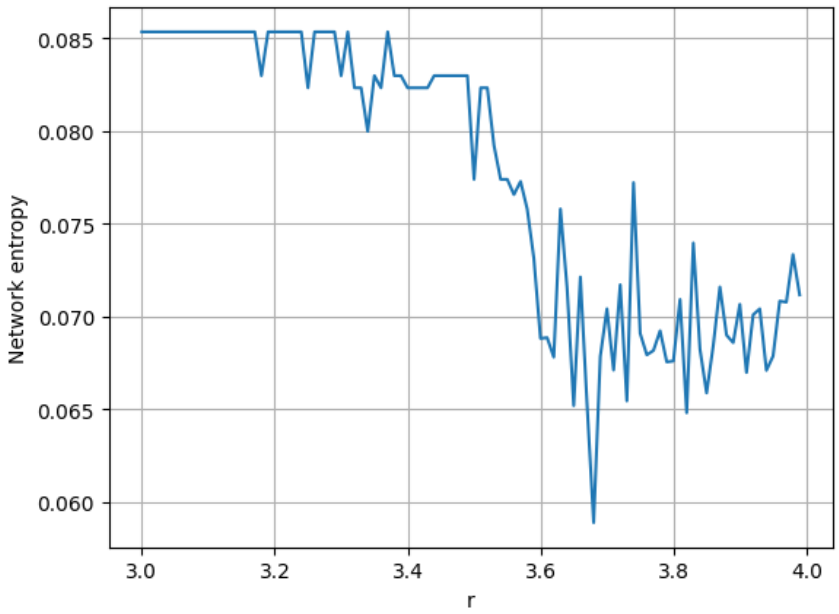


Figure 8: Network entropy as a function of r

Contrary to the permutation entropy in Figure 7, the permutation entropy seems to decrease as the chaos increases.

5 Discussion

The Feigenbaum constant was difficult to calculate without a large error. The algorithm for bifurcation detection needed a lot of work and still did not really produce a fully satisfactory result. The main reason for the difficulty in this calculation was that the raw data in the bifurcation diagram had some "bumps" around every bifurcation that got more severe for every r.

The ordinal analysis tells us a lot about the nature of the chaotic section. One unexpected finding was how far from random the patterns were. One peculiar finding was that two consecutively descending (pattern six) values in a row do not appear any time, anywhere in the logistic map data for any growth rate parameter studied. Much time was spent to formulate a proof for why or develop some intuition about it, but without success.

The comparison of the permutational entropy with and without noise are interesting. As the range of the logistic map is only one, and the variance of the Gaussian noise is also one, the level of noise could be considered quite extreme. Yet, the pattern in the range [3.5, 4] shows remarkable persistence. The reason for why the permutational entropy in the lower range shows no persistence at all is likely due to what can be observed in Figure 4. In the lower range of r, four patterns are

split evenly. This is due to the non-chaotic periodic undulation for the logistic map equation in this range of r. If the variation in the time series is little, and the values are close, the noise will be close to being the only characteristic factor.

When it comes to the network entropy, there were many difficulties. Small networks (n=50) had to be constructed, as the horizontal visibility algorithm has a very high time complexity. Yet, the simulations were time-consuming enough that the practicality of experimentation suffered. Attempts with much smaller networks were made but with poor results. The concept's level of abstractness and lack of standardized methods of implementation led to additional difficulties. The end result seems counter-intuitive, as the entropy decrease somewhat as the bifurcations take place. One thing that is however consistent with the theory is the sharp spikes in the network entropy that coincide exactly with the bifurcation points. Based on the data, the permutational entropy seems to be more informative, which seems to be consistent with Figure 4.

6 Summary

This report provides an analysis of a logistic map, a dynamical system that represents the population size at each time step as a function of its previous size. The logistic map exhibits complex behavior, including periodic oscillations, chaotic dynamics, and the emergence of stable and unstable fixed points. The report explores the use of bifurcation diagrams to represent the behavior of the system as the groth rate r is varied, and how the Feigenbaum constant, which describes the period-doubling behavior in complex dynamical systems, can be calculated from the bifurcation points. Additionally, ordinal pattern analysis and permutation entropy are used to investigate patterns in the chaotic portions of the time series data. The main finding is that the time series of the chaotic sections of the diagram are indeed chaotic, but not random, as several patterns and curiosities emerge.