Arshia Singh
ANLY601: Advanced Machine Learning

Evolutionary Markov Chain Monte Carlo in Decryption: Benchmarks and Comparison to Single-chain Markov Chain

1. Abstract

I applied both single chain and evolutionary Markov Chain Monte Carlo methods to a piece of substitution ciphered text to try to recover the mappings, and the original text. I measured the efficiency of each method via the computational time and total steps travelled needed to achieve recovery of the original text. My results did not indicate any clear advantages to using evolutionary methods over a single chain Markov Chain Monte Carlo approach, but I think it would be fruitful to continue exploring further applications of evolutionary methods to this problem, particularly via the parallelization of a population of chains.

2. Introduction

Cryptography is an interesting space to apply evolutionary Markov Chain Monte Carlo methods. Although research shows that Markov Chain Monte Carlo can recover text given its transformation, there hasn't been any work done to see if the addition of evolutionary methods could further increase the efficiency of this process. Cryptography is a compelling space to conduct testing in because it is possible to create a transformation for a given text and then validate if it is correctly recovered by an algorithm. Even though simple substitution ciphers are not frequently used for encryption, in this setting they serve as a useful benchmarking problem to ascertain if evolutionary MCMC can be leveraged in modelling other problems involving stochastic simulation; problems like these such as simulating crowd movements and the spread of a disease do not typically have a point of model convergence, which can make it difficult to compare the efficacy of different simulation methods.

3. Related Work

System Identification (2001) appears to be the foundational paper that in the development of eMCMC. The authors describe creating the algorithm as a combination of evolutionary methods and Markov Chain Monte Carlo. Unlike traditional evolutionary algorithms which create members of a population at random, eMCMC draws from a known probability distribution. Additionally, eMCMC leverages search points (or fitness tests) to find a solution more efficiently than traditional MCMC can alone. In this paper the authors apply this method to find the architecture of a system given its inputs. This is an important theoretical problem to test the robustness of this solution relative to MCMC or evolutionary algorithms alone, but I think it would be interesting to see if this increased efficiency translates to real-world problems and simulations as well.

Drugan and Thierens (2003) summarize further research on this topic in their paper Evolutionary Markov Chain Monte Carlo; they classify development in this space into two categories: algorithmsthat are family-competitive and algorithms that are population-driven. Family-competitive algorithms iterate via acceptance rules that are set at the family level. This means that chains from a population can exchange information which informs the sample space of the next step in the chain. The acceptance rules in this space are considered coupled. In one example of this method, you would start with a “family” of four chains, which split in to two pairs that each share information to determine the proposal, recombining in the process. On the other hand, population-driven algorithms shift their proposal distributions based on the current state of the entire population. Instead of two chains exchanging information, proposals and offspring are created by sampling from the distribution of the entire population. They propose a new algorithm that uses fitness ordered temperature assignment and elitist coupled acceptance rules to reach convergence, and they show that it performs similarly to algorithms that employ parallel tempering and parallel tempering with recombination in terms of recovering solutions to a deceptive trap function.

Although Drugan and Thierens (2003) discuss many of these algorithms at length, and provide a comparison of how they perform in exploring a solution space, there hasn’t been a lot of work done on the computational or step length efficiency of these methods. Additionally, a survey of documentation indicates that there isn’t any current application of these techniques that has been packaged for use in R. I think it would be worth developing a methodology for even the simpler evolutionary Markov Chain Monte Carlo algorithms to be applied in R, as well as an experimental test that will allow for evaluation of this efficiency.

Diaconis (2009) wrote a paper which discusses an interesting real-world application of Markov Chain Monte Carlo methods: in the space of cryptography. Given a text that has been encrypted using a series of substitution codes, Markov Chain Monte Carlo can be used to switch a letter in the text. The proposal is developed based on a sample from a probability distribution modeled on letter frequency in a reference text. The acceptance criteria is based on the frequency of letter pairings in the reference text (for example, the pairing “AS” is more probable than “ZK”). Typically, a short piece of text such as a few sentences can reach convergence in only a few thousand steps.

Chen and Rosenthal (2010) go even further than substation-based encryption to show that Markov Chain Monte Carlo can be used to successfully decrypt transposition ciphers and even substitution-plus-transposition ciphers. To do this they combine independent runs of chains and vary their proposal methods (in this case referred to as “cipher attacks”). This seems like a particularly fruitful space for the addition of evolutionary methods to Markov Chain Monte Carlo since they similarly allow for additional chains and different proposal mechanisms that standard Markov Chain Monte Carlo.

From the perspective of evolutionary algorithms, Alsharafat (2015) explores the use of evolutionary algorithms to create strong encryption methods through crossover and mutation-based ciphers. However, the paper doesn’t explore the use of these methods to decrypt an encrypted text with an unknown key. Garg (2010) does look at the theoretical application of evolutionary algorithms as a mechanism for cryptanalysis, or decryption for text with an unknown key (also known as breaking a cipher). However, despite outlining the methodology for application of these methods like genetic algorithms and simulated annealing, the author does not apply them to actual examples, and doesn’t evaluate their effectiveness at decryption versus other methods.

4. Datasets

My dataset consists of two pieces of text, which I downloaded from Project Gutenberg: the full text of War and Peace by Leo Tolstoy, and a paragraph of text from Great Expectations by Charles

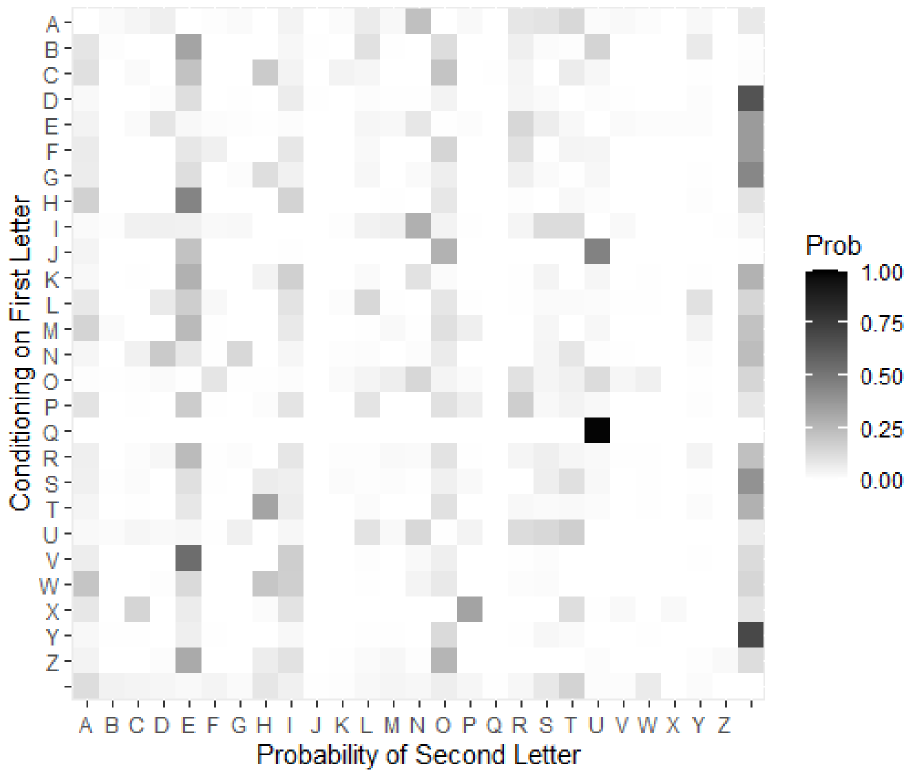
Dickens. The full text of War and Peace served as my reference text, which allowed me to create a transition matrix and calculate the log likelihood of my proposed decryption at each step. Separately, the paragraph from Great Expectations served as the test text. I downloaded both texts from the Gutenberg Project website and read them into R, removing punctuation and converting them to uppercase passages. In the case of the passage from Great Expectations, I also encrypted the text via a series of substitution transformations.

War and Peace was a useful reference text because it is quite long (8.2 megabytes, or about 65,219 lines of text) and can therefore be used to establish common letter occurrence probabilities for the English language with a large variety of words. I similarly chose another passage from a book written by a different author in the 1860s, Great Expectations, to serve as the test passage for decryption. This was a passage of only 7 lines, and was the paragraph at the end of Chapter 1.

An interesting aspect of using a Markov Chain Monte Carlo method for decryption is that unlike some NLP models that require lengthy dictionaries or packages tailored to work in only specific languages, it is relatively easy in this case to swap out a different lengthy reference text in another language with a test set from that language, and still achieve similar rates of convergence.

5. Methods

First, I read in the reference text, War and Peace. I created a 27x27 matrix to find the transition probability matrix for the co-occurrence of letters in the reference text. This tells me both how likely a given letter is to follow another letter, and the 27th column also provides information on how common the letters are generally. I used ggplot to display these probabilities:



There is already a lot of information present in this chart – we can see that “U” is very likely to follow the letter “Q” in a piece of text, for example. The last column also shows us that the letters “Y” and “D” are more likely to appear at the end of a word than other letters, while the last row shows that the letters “A” and “T” are marginally more likely to appear at the beginning of a word than other letters.

Next, I begin to look at the test data, the paragraph from Great Expectations. I encrypt the text with a series of substitutions. To implement a standard Markov Chain Monte Carlo technique, I initialize a random mapping of letters and then use the transition matrix above to calculate the initial log likelihood of the text decoded using the map. I then create a proposal by sampling two letters to swap in the mapping, and then calculate the decoded text’s log likelihood. If the log likelihood meets the acceptance criteria, which favors maximizing likelihood with a degree of random rejection, I accept the swap, otherwise I retain the mapping from the previous step. I run a few hundred iterations and time the full process as well as the time taken in each step.

I then apply a version of evolutionary MCMC to try to recover the original text in order to compare its effectiveness to the standard MCMC. In this application I use a population-based eMCMC method, initially starting with three different random mappings and calculating their associated likelihoods. Then, for each of the 3 chains, I make ten proposals by sampling ten sets of two letters to swap. I continue to accept the swaps into the population of 30 if they meet the acceptance criteria outlined above. Then, the population is culled, with only the three highest likelihood mappings surviving, or recombining into the next iteration, or generation. Again, I run a few hundred iterations and time the full process and the time taken in each step.

6. Results

I ran the above procedure 5 times using each method to determine its average efficacy. The results of each run in number of steps taken, total time taken, and average step time are displayed in the following table:

Run	**MCMC steps**	**eMCMC steps**	**MCMC total time (secs)**	**eMCMC total time (secs)**	**MCMC Average step time (secs)**	**eMCMC average step time (secs)**
1	114	93	145.96	96.38	1.29	1.02
2	53	103	60.71	109.69	1.15	1.05
3	98	108	106.19	113.23	1.09	1.02
4	75	86	65.73	88.26	0.88	1.03
5	79	73	95.70	93.87	1.22	1.28
Avg	**83.8**	**92.6**	**94.86**	**100.29**	**1.12**	**1.08**

In some cases, the algorithm did not converge - this occurred three times with the single-chain Markov Chain Monte Carlo method, and once with the evolutionary Markov Chain Monte Carlo method. These runs were not included above but I think they should be considered in analyzing the results.

7. Discussion of Results

In this experiment, it looks like the evolutionary Markov Chain Monte Carlo method did not outperform the traditional single chain Markov Chain Monte Carlo method. I think that this occurred for a number of different reasons. First, the single chain Markov Chain Monte Carlo computations were already using a lot of my machine's system resources, although they were relatively fast; when I tried to expand this to a population-based approach like evolutionary Markov Chain Monte Carlo, it was unable to successfully perform operations over a lot of chains or with a large "population" set. I had to optimize the parameters based on a heuristic approach of what was feasible to compute on my machine, which isn't optimal for determining the best parameters for the algorithm itself.

As a result of my evolutionary Markov Chain Monte Carlo approach having such a small population (30 members composed of 3 chains with 10 proposals), I didn't see a large increase in step level computation over the single chain Markov Chain Monte Carlo approach. Generally, the two methods seemed to produce about the same results, with the evolutionary approach generally taking slightly longer and more steps to reach convergence. I think this is likely due to the fact that I only included 5 runs in my analysis and convergence is frequently related more to the initial sample maps than to the algorithms themselves. This is also evident in my note under the results - there were 3 cases where the single chain Markov Chain Monte Carlo method did not converge, and one case where the evolutionary Markov Chain Monte Carlo did not converge, even after 300 iterations. I think this is one area that evolutionary Markov Chain Monte Carlo methods show a lot of promise - with a singe Markov Chain Monte Carlo approach, most researchers recommend running a few chains before trying to assess the results; in this case, with the evolutionary Markov Chain Monte Carlo approach it seems less likely that the algorithm will not reach convergence within a few hundred steps.

8. Conclusions

Overall, my results were not indicative of any clear efficiencies in using an evolutionary Markov Chain Monte Carlo approach relative to a single chain Markov Chain Monte Carlo approach in decrypting text. Still, I think my initial research does indicate that this is a fruitful area of research in stochastic simulations. Particularly in applying these methods, it would have been useful to test under more controlled computing conditions, such as in the cloud, rather than on a single machine. I also chose parameters for evolutionary model’s number of chains and proposals based on heuristic sampling and the limits of my own machine, but further work could be done to tune these parameters for increased recovery in a variety of settings. Furthermore, one of the huge advantages in incorporating an evolutionary approach to Markov Chain Monte Carlo methods, particularly with population-based methods, is the ability to easily parallelize chains, which I was unable to take advantage of working on a single machine. I believe this would only increase the efficiency of these methods, although it would require a tradeoff of more computing power than single-chain Markov Chain Monte Carlo techniques.

Bibliography

Agarwal, Rahul. (2019). Applications of MCMC for Cryptography and Optimization. (https://towardsdatascience.com/applications-of-mcmc-for-cryptography-and-optimization-1f99222b7132)

Alsharafat, Wafa. (2015). Evolutionary genetic algorithm for encryption. 10.1109/ICCIC.2014.7238559. (https://www.researchgate.net/publication/285612552_Evolutionary_genetic_algorithm_for_encryption)

Byoung-Tak Zhang, Dong-Yeon Cho, System identification using evolutionary Markov chain Monte Carlo, Journal of Systems Architecture, Volume 47, Issue 7, 2001, Pages 587-599, ISSN 1383-7621, https://doi.org/10.1016/S1383-7621(01)00017-0. (http://www.sciencedirect.com/science/article/pii/S1383762101000170)

Chen, Jian & Rosenthal, Jeffrey. (2012). Decrypting classical cipher text using Markov chain Monte Carlo. Statistics and Computing. 22. 397-413. 10.1007/s11222-011-9232-5. (http://probability.ca/jeff/ftpdir/decipherart.pdf)

Diaconis, Persi. (2009). The Markov Chain Monte Carlo Revolution. Bulletin of the American Mathematical Society. 46. 179-205. 10.1090/S0273-0979-08-01238-X. (https://www.statweb.stanford.edu/~cgates/PERSI/papers/MCMCRev.pdf)

Dickens, C., & Mitchell, C. (2003). *Great Expectations*. (https://www.gutenberg.org/files/1400/1400-h/1400-h.htm)

Drugan, Madalina & Thierens, Dirk. (2003). Evolutionary Markov Chain Monte Carlo. 63-76. (https://dspace.library.uu.nl/bitstream/handle/1874/24000/drugan_03_evolutionarymarkov.pdf?sequence=1)

Garg, Poonam. (2010). Evolutionary Computation Algorithms for Cryptanalysis: A Study. International Journal of Computer Science and Information Security, Vol. 7, No. 1. (https://arxiv.org/pdf/1006.5745)

Landgraf, Andrew J. (2013). Text Decryption Using MCMC. (https://alandgraf.blogspot.com/2013/01/text-decryption-using-mcmc.html)

Tolstoy, L., Pevear, R., & Volokhonsky, L. (2007). *War and Peace*. (http://www.gutenberg.org/files/2600/2600-h/2600-h.htm)