Cipher Breaking using Markov Chain Monte Carlo

Thiago R. Bergamaschi
Center for Theoretical Physics, 77 Massachusetts Ave., Cambridge, MA 02139-4307

(Dated: May 9, 2019)

This is my report of Part 2 of the 6.437 final project in Spring 2019. We first implemented a Markov Chain to break a substitution Cipher encryption, and then refined our analysis in part 2. In particular, I propose 2 enhancements: Ensemble MCMC, i.e. running multiple walks on the MC in parallel, and initialization by linear assignment. We also adapt the search for the existence of a breakpoint in the ciphertext.

I. INTRODUCTION

A substitution cipher is a method of encryption by which units of plaintext the original message are replaced with ciphertext the encrypted message according to a ciphering function, e.g. some permutation of the alphabet $A = \{a, b, \cdots, z\} \cup \{\ .,\}$.

Given apriori probabilities for the letters in the english language, $p_a(i) = \mathbb{P}[i]$ for $i \in A$, and bigram transition probabilities, $M_{ij} = \mathbb{P}[x_k = i|x_{k-1} = j]$, there is a closed form expression for the likelihood of the observed ciphertext $\mathbf{y}$ for a given permutation f.

$$p_{\mathbf{y}|f}(\mathbf{y}|f) = p_a(f^{-1}(y_1)) \times \prod_{i=1} \mathbb{P}[f^{-1}(y_{i+1})|f^{-1}(x_i)] \quad (1)$$

For part 1 of this project, we implement a simple Markov Chain on the set of possible permutations. This approach is shown in section 2 of the report, where we discuss and test general implementation enhancements as well. In section 3, we describe the algorithmic enhancements used in part 2. In particular, running multiple ($N = 5$ typically) walks on the Markov Chain and returning the maximum decreases the failure probability considerably, at little to no increase in runtime. Next, we improve on the permutation initialization by linear assignment. We define costs for each swap ij of the permutation, based on the a priori probability of i in the English language, and the empirical probability of j in the ciphertext, and solve the assignment problem with the Hungarian algorithm.

II. MARKOV CHAIN MONTE CARLO

Given these likelihoods, we can construct a Markov Chain on the set of possible permutations. Define edges between permutations that swap a pair of letters. At each vertex f, we pick one of $\binom{28}{2}$ adjacent edges f', and transition there with probability

$$\min\{1, \frac{p_{\mathbf{y}|f}(\mathbf{y}|f')}{p_{\mathbf{y}|f}(\mathbf{y}|f)}\} \quad (2)$$

the algorithmic description follows.

Input: alphabet A, ciphertext $\mathbf{y}$, iteration number T.
Output: most probable permutation f_p

1. initialize f with some random permutation of A, and iteration count $t = 0$. Compute $p_{\mathbf{y}|f}(\mathbf{y}|f)$
2. While $t < T$, construct f' by swapping 2 letters of f, and compute $p_{\mathbf{y}|f}(\mathbf{y}|f')$
3. If $\frac{p_{\mathbf{y}|f}(\mathbf{y}|f')}{p_{\mathbf{y}|f}(\mathbf{y}|f)} > 1, f = f'$.
4. $t \leftarrow t + 1$, repeat step 2.
5. Return f

Clearly, there is immediate space for improvement by speeding up the $p_{\mathbf{y}|f}(\mathbf{y}|f)$ computation. Let us do so by definining a sufficient st

Let us now describe some of the performance enhancements used on the MCMC computations. Specifically, we focus on defining a less memory intensive sufficient statistic, quickly computing loglikelihoods, and perturbing the 0 transition probabilities s.t. the loglikelihoods are strictly finite.

A. Performance Enhancements

Note first that we can compute the log likelihood $\log p_{\mathbf{y}|f}(\mathbf{y}|f)$ from $\log p_a(f^{-1}(y))$, and the count matrix C where c_{ij} is the number of times j is preceded by i in y. It follows

$$\log p_{\mathbf{y}|f}(\mathbf{y}|f) = \log p_a(f^{-1}(y)) + \sum_{ij} c_{ij} \log M_{f(i)f(j)} \quad (3)$$

and thus $C, p_a(\cdot)$ is a sufficient statistic for our model.

Moreover, we can precompute C in $O(\text{text size})$ and compute the loglikelihood as in (3), instead of passing through the text at every vertex.

Finally, note that if we initialize the MCMC at some permutation f that implies $p_{\mathbf{y}|f}(\mathbf{y}|f) = 0$ due to some 0 valued M_{ij}, then the loglikelihood value will be $-\infty$. More importantly, empirical evidence suggests that randomly initializing f sets a considerable probability of failure, due to taking a long time to leave this minima. We easily fix this by assigning large, finite, negative values to the values of $\log M_{ij}$ when $M_{ij} = 0$. In particular, -2000 was found to work well.

B. Results of Part 1

To briefly comment on current performance, a *single pass* of the MCMC when successful runs in $\approx 8s$ and decoding accuracy $\approx .96$. However, due to faulty initialization

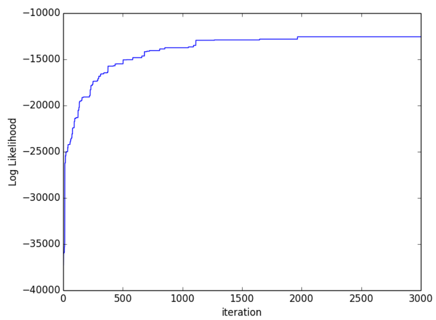

FIG. 1: Log Likelihood as function of number of Iterations