APMA1941D Project 1 - Decoding a Substitution Cipher Using Markov Chain Monte Carlo

Alan Chen

February 2022

Contents

1 Introduction

In this report, we detail the procedure of decoding a bijective permutation on a letter state space (substitution cipher) by combining a stationary Markov chain with a transition matrix based on the digram model of English text and the Markov Chain Monte Carlo (MCMC) sampling method. We first discuss some background of MCMC, then the algorithm implemented and the results, and finally conclude with various optimizations and improvements that were implemented and some that are subject to further investigation.

2 Definitions

$\mathcal{S}$ throughout will refer to the state space. For this very simple project, the state space was assumed to be just the 26 lowercase English letters and the space character i.e. $\mathcal{S} = \{a, b, \ldots, z, \ \}$. We let $|\mathcal{S}|$ be the order of $\mathcal{S}$ (in this specific case, 27).

$P_{\text{true}}(c)$ refers to the relative frequency of character c in "true" English (in our case approximated by mining texts). Q will denote the transition matrix of the Markov chain, also derived through data mining.

Finally, we use N to denote the length of the text (both ciphertext and plaintext). This will be used mainly in algorithmic analysis in the Optimization section.

3 Background

3.1 Gibbs Distribution

The Gibbs Distribution is defined by the following pmf:

$$p_\beta = \frac{e^{-\beta E(x)}}{Z_\beta}, Z_\beta = \sum_{y \in \mathcal{S}} e^{-\beta E(y)}, \tag{1}$$

where $E(x)$ is some "energy" function and $\beta > 0$ being a parameter of the model.

Jayne's work concludes that (with proof based in fundamental information theory) this Gibbs distribution is the most unbiased estimate given partial observations (data). So, we would like to be able to sample/work with this distribution, but that is not that easy, as our state space $\mathcal{S}_{27}$ of permutations on $\mathcal{S}$ is huge, so calculating Z_β directly is incredibly difficult, if not impossible. Instead, we will employ MCMC to help us sample from this approximation distribution.

3.2 Bayesian Inference and Maximum Likelihood

The key idea that comes from Bayesian inference is to find σ such that a **posterior** is maximized. In our case, we want to maximize $\mathbb{P}(\sigma | b_1 \ldots b_n)$, or the probability of σ being the permutation given the observed data. Through applications of the definition of conditional probability and a few assumptions about uniformity over all 27! permutations, we can discover the powerful relationship:

$$\mathbb{P}(\sigma|b_1 \dots b_n) \propto \mathbb{P}_{\text{true}}(\sigma^{-1}(b_1))Q(\sigma^{-1}(b_1), \sigma^{-1}(b_2)) \cdots Q(\sigma^{-1}(b_{n-1}), \sigma^{-1}(b_n)) \tag{2}$$

So, the idea is that we define the RHS to simply be the value that we are maximizing i.e. E. Since there's a big product and generally we like to minimize E instead of maximizing it (by convention), we take the negative log on both sides to arrive at the energy function we will attempt to minimize:

$$E(\sigma) = \log \mathbb{P}_{\text{true}}(\sigma^{-1}b_1) + \sum_{i=1}^{n-1} \log Q(\sigma^{-1}(b_i), \sigma^{-1}(b_{i+1})) \tag{3}$$

3.3 Markov Chain Monte Carlo

The previous two sections have essentially outlined the main problem: we would like to maximize some function over an incredibly large state space, and we know that our best estimator is the Gibbs Distribution. Markov Chain Monte Carlo (MCMC) is a probabilistic sampling method that describes a random walk on all possible σ that we will employ to helping to solve this problem of finding $\sigma*$ that maximizes Equation 3.

The algorithm for this context is described in Algorithm 1.

Algorithm 1 MCMC Iteration

```
σ ← random permutation from S
I ← 0
while I < iterations do
    σ' ← random swap on σ
    E_old ← E(σ)
    E_new ← E(σ')
    ΔE ← E_new − E_old
    if ΔE < 0 then σ = σ'
    else
        r ← random from 0 to 1
        if r < e^{−βΔE} then σ = σ'
        end if
    end if
    I ← I + 1
end while
```

In general, MCMC with the Metropolis scheme is not an optimization method: it is really a way to sample from an unknown Gibbs distribution with a pmf on $\mathcal{S}$ as defined in Eq. 1. But, in our specific case of investigating permutations of English text, the algorithm essentially becomes an optimization algorithm because of one key feature of English that significantly alters the shape of E, namely its sensitivity to even just small perturbations from true English.

Intuitively, consider the sentence "the quick brown fox jumps over the lazy dog." We can take one step away from the current state, maybe swapping o for x and x for o. The sentence immediately devolves into chaos: "The quick brxwn fxo jumps xver the lazy dxg," which we immediately recognize as gibberish. Because English text is so sensitive to these small changes, we expect that a well constructed energy function will contain an extremely deep and narrow well around $\sigma*$, as the E for the correct permutation should be significantly smaller than the E at any of the other permutations.

Because of the sampling strategy of MCMC, where we only leave a state for another state with a higher E with a probability ratio, it is extremely unlikely that after we enter this deep well, we will exit it. So, we can expect that after discovering this well, we will just keep sampling that point, rejecting all potential moves to other nodes in the graph, in essence "converging" to an optimal $\sigma*$.

4 Methodology

4.1 Data Mining

In order to have a good representation of the English text to make , we must first mine a corpus, or a large body of text, to determine the relative frequencies of each character and the important transition matrix Q.

In theory, we can just choose any long passage of English text such as War and Peace and mine it to count the values of each individual digram and letter. However, there does arise some issues with using just one text. For example, an important character in War and Peace is named "Dolokhov", and his name appears upwards of 500 times throughout the text. However, the digrams *kh* and *ov* are not actually that common in the majority of English text, potentially inflating the chances we end up converging on some (wrong) permutation that has a lot of *kh* and *ov* patterns in the decoded text.

To help address this, we can simply mine more data from unique sources. I specifically data mined both War and Peace and Crime and Punishment, two long books that have a lot more standard English together that offset and diminish the overrepresentation of certain digrams. Since $\mathcal{S}$ was just restricted to the lowercase letters and space, I also needed to first prune and clean the data of other characters, and convert everything to lowercase.

Since we do eventually take the logarithm of the probabilities, we do need to make sure each probability is nonzero. So, for digrams such as *xx* that might never appear in the digram text, we just need to assign some arbitrarily small probability. In my final program, I just used e^{-20} since it will eventually play well with the logarithm and was sufficiently small for my purposes (it can be adjusted to be smaller if there is a lot of appearances of generally uncommon digrams in the final decoded text). An alternative can also be just assigning the value 1 to the count and computing the probability the same way as other values.

4.2 Algorithm

4.2.1 Relative Frequency and Transition Matrix Construction

To get the relative frequency of each letter, all I did was simply implement a function that iterates through the cleaned text and count each letter, in the end scaling by the sum of all counts. For the digram counts, it was the exact same (first getting the counts), except that because the entire transition matrix is conditional probabilities, we only scale based on the sum of the rows such that the probabilities in each row sum to 1.

4.2.2 MCMC and Convergence

The implementation code of MCMC follows very linearly from the pseudocode in Algorithm 1. The only things we need to consider are: how many iterations in total, how we determine the convergence threshold, and the parameter β in the Gibbs distribution.

Iterations and convergence threshold were empirically tuned. Generally, it was observed that the algorithm converged to the correct solution within a few thousand iterations when it found the correct permutation, and when it didn't it never converged to an permutation. So, for my trials, 25000 total iterations was enough. However, I also added a convergence threshold, in which if the permutation did not change for 5000 iterations, the algorithm would terminate early. These two numbers are pretty arbitrary, and could be more optimally tuned in the future.

β was also mainly an empirically tuned parameter. It needs to be high enough such that we have the ability to escape local minima but low enough such that once we reach a likely solution we won't leave it. For me, any values between $[0.5, 1]$ worked well and converged to the correct solution.

5 Optimizations/Improvements

5.1 Repeated Trials

Due to the overall random and probabilistic nature of the algorithm, it is possible that we simply get "bad luck" and end up converging on a local minimum that is not actually the extremely deep well of the correct $\sigma*$. This is especially potent when the initial σ is particularly unlucky.

A very simple and naive way to address this is simply by running the entire algorithm multiple times, each time storing the permutation. After the iterations are done, we can simply return the permutation that minimized E. Formally, for k repeated applications of the algorithm, and σ_i being the permutation converged on at iteration k, we have:

$$\sigma* = \underset{\sigma_i \in \{\sigma_1,\ldots,\sigma_k\}}{\arg\min} E(\sigma_i) \tag{4}$$

Let's let the probability of failure (i.e. the probability we don't find the true $\sigma*$) be q for each iteration, with $0 \leq q < 1$. Then, as we run the algorithm more times, the probability of failing exponentially decays:

$$\mathbb{P}(\text{failure}) = q^k. \tag{5}$$

To see how well this works, we can try with $q = 0.5$: after repeating just $k = 10$ times, the chance of not finding the best permutation drops to less than $0.001 = 0.1\%$.

q depends mainly on the parameter β, which we can tune to try and minimize, potentially resulting in failure probabilities. Overall, this improvement greatly increased the chances of discovering the true $\sigma*$, albeit slowing down the process quite a bit.

5.2 Invariance Under Ciphertext Length

An important observation we can make is that under the naive framework, we repeatedly iterate over a ciphertext that is **constant over all iterations**, unlike the permutation. This motivates the following optimization: instead of re-iterating over the ciphertext each iteration, we precompute a $|\mathcal{S}| \times |\mathcal{S}|$ **ciphertext matrix C**, defined in the following way:

$$\mathbf{C}_{b_i b_j} = \sum_k \delta_{b_k b_{k+1}}, \text{where} \tag{6}$$

$$\delta_{b_k b_{k+1}} = \begin{cases} 1 & \text{if } b_k b_{k+1} = b_i b_j \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Informally, $\mathbf{C}_{b_i b_j}$ represents the number of times the sequence $b_i b_j$ appears in the ciphertext. With this precomputed matrix, our likelihood function from Equation 3 becomes:

$$E(\sigma) = - \sum_{(b_i, b_j) \in \mathcal{S} \times \mathcal{S}} \mathbf{C}_{b_i b_j} \log Q_{\sigma^{-1}(b_i)\sigma^{-1}(b_j)} \tag{8}$$

Notice the power of this optimization: it changes the complexity per iteration from $O(N)$, where N is the length of the ciphertext, to $O(|\mathcal{S}|^2)$ with a one time $O(N)$ precomputation. While it may not look good on the surface, this optimization actually makes the algorithm's complexity **invariant** under the length of the ciphertext (minus the precomputation). For any $N > |\mathcal{S}|^2$, this algorithm will be strictly faster.

On the relatively short texts for this assignment, the benefit of this trick is less perceivable, but in principle this optimization is very powerful as it makes our algorithm runtime independent of the length of the text.

5.3 Storing the Previous Iteration

We note that throughout the iterations, there could be a lot of recomputation of the exact same value, namely the likelihood of the current permutation. We can remove an extraneous pass by simply storing the energy value of the previous iteration.

Combined with the previous optimization, this greatly speeds up each iteration to one pass of $O(|\mathcal{S}|^2)$ instead of two passes of $O(\text{length of ciphertext})$.

5.4 Exploiting the Difference

Finally, we realize that we don't actually need to recompute for the entire ciphertext, since we only care about ΔE. Thus, in theory, we only need to consider the newly changed values in our computation of E, which only comes from the transposition of two characters. I wasn't able to fully implement this, but instead of iterating over the entire matrix of ciphertext digram counts $\mathcal{C}$, we only need to iterate over the specific columns and rows in the $\mathcal{C}$ with the changed letters. This would reduce a quadratic algorithm into a $O(|\mathcal{S}|)$ (linear) order algorithm.

5.5 Potentially Introducing Bias

An issue I noticed that would sometimes occur is that the permutation would get stuck one or two transpositions away from the true permutation. I believe one way to potentially address this that I did not get the chance to try would be to rule out certain swaps in the permutation to prevent the random number generator from exploring that path after we have tried them once. The implementation details are not immediately obvious to me, but this would hopefully increase the probability that we discover the final swap in the puzzle.

One way to potentially do it is to set some threshold ε. If the probability of visiting that other node is below ε, we can delete it from consideration in the random walk, thus increasing the probability that some other transposition is selected instead. Additionally, it is not clear how much this would impact convergence, but I think using some "warmup" idea could be useful, where said policy is not implemented in the beginning but is slowly introduced as the algorithm approaches the final permutation.

This would not be too difficult or taxing on memory, depending on $|\mathcal{S}|$ size as the number of possible edges out from the current node is only $\binom{|\mathcal{S}|}{2}$. However, it could potentially mess up the ability of the algorithm to exit local minima, as we are cutting off potential out routes, which is why regulating the introduction of the policy would be vital.

6 Appendix

6.1 Decoded Texts

The decoded texts are pasted below. Most took between 2000 to 5000 iterations to find the true permutation (if it ever did).

6.1.1 f.txt (Feynman's Lectures on Computation)

i would now like to take a look at a subject which is extremely interesting but almost entirely academic in nature this is the subject of the energetics of computing we want to address the question how much energy must be used in carrying out a computation this doesnt sound all that academic after all a feature of most modem machines is that their energy consumption when they run very fast is quite considerable and one of the limitations of the fastest machines is the speed at which we can drain off the heat generated in their components such as transistors during operation the reason i have described our subject as academic is because we are actually going to ask another of our fundamental questions what is the minimum energy required to carry out a computation to introduce these more physical aspects of our subject i will return to the field covered in the last chapter namely the theory of information it is possible to treat this subject from a strictly physical viewpoint and it is this that will make the link with the energy of computation to begin with i would like to try to give you an understanding of the physical definition of the information content of a message that physics should get involved in this area is hardly surprising remember shannon was initially interested in sending messages down real wires and we cannot send messages of any kind without some interference from the physical world i am going to illustrate things by concentrating on a particular very basic physical model of a message being sent i want you to visualize the message coming in as a sequence of boxes each of which contains a single atom in each box the atom can be in one of two places on the left or the right side if its on the left that counts as a zero bit if its on the right its a one so the stream of boxes comes past me and by looking to see where each atom is i can work out the corresponding bit to see how this model can help us understand information we have to look at the physics of jiggling atoms around this requires us to consider the physics of gases so i will begin by taking a few things i need from that let us begin by supposing we have a gas containing n atoms or molecules occupying a volume v one we will take this gas to be an exceptionally simple one each atom or molecule within it we take the terms to be interchangeable here is essentially free there are no forces of attraction or repulsion between each constituent this is actually a good approximation at moderately low pressures i am now going to shrink the gas pushing against its volume with a piston compressing it to volume v two i do all this isothermally that is i immerse the whole system in a thermal bath at a fixed temperature t so that the temperature of my apparatus remains constant isnt it wonderful that this has anything to do with what were talking about im going to show you how first we want to know how much work w it takes to compress the gas now a standard result in mechanics has it that if a force f moves through a small distance dx the work done dw is fdx if the pressure of the gas is p and the cross sectional area of the piston is a we can rewrite this using f equals pa and letting the volume change of the gas dv equals adx so that dw is pdv now we draw on a standard result from gas theory for an ideal gas at pressure p volume v and temperature t we have the relation pv equals nkt where n is the number of molecules in the gas and k is boltzmanns constant as t is constant our isothermal assumption we can perform a simple integration to find w since v two is smaller than v one this quantity is negative and this is just a result of the convention that work done on a gas rather than by it has a minus sign now ordinarily when we compress a gas we heat it up this is a result of its constituent atoms speeding up and gaining kinetic energy however in our case if we examine the molecules of the gas before and after compression we find no difference there are the same number and they are jiggling about no more or less energetically than they were before there is no difference between the two at the molecular level so where did the work go we put some in to compress the gas and conservation of energy says it had to go somewhere in fact it was converted into internal gas heat but was promptly drained off into the thermal bath keeping the gas at the same temperature this is actually what we mean by isothermal compression we do the compression slowly ensuring that at all times the gas and the surrounding bath are in thermal equilibrium

6.1.2 h.txt (Harry Potter and the Sorcerer's Stone)

mr and mrs dursley of number four privet drive were proud to say that they were perfectly normal thank you very much they were the last people youd expect to be involved in anything strange or mysterious because they just didnt hold with such nonsense mr dursley was the director of a firm called grunnings which made drills he was a big beefy man with hardly any neck although he did have a very large mustache mrs dursley was thin and blonde and had nearly twice the usual amount of neck which came in very useful as she spent so much of her time craning over garden fences spying on the neighbors the dursleys had a small son called dudley and in their opinion there was no finer boy anywhere the dursleys had everything they wanted but they also had a secret and their greatest fear was that somebody would discover it they didnt think they could bear it if anyone found out about the potters mrs potter was mrs dursleys sister but they hadnt met for several years in fact mrs dursley pretended she didnt have a sister because her sister and her good for nothing husband were as undursleyish as it was possible to be the dursleys shuddered to think what the neighbors would say if the potters arrived in the street the dursleys knew that the potters had a small son too but they had never even seen him this boy was another good reason for keeping the potters away they didnt want dudley mixing with a child like that when mr and mrs dursley woke up on the dull gray tuesday our story starts there was nothing about the cloudy sky outside to suggest that strange and mysterious things would soon be happening all over the country mr dursley hummed as he picked out his most boring tie for work and mrs dursley gossiped away happily as she wrestled a screaming dudley into his high chair none of them noticed a large tawny owl flutter past the window at half past eight mr dursley picked up his briefcase pecked mrs dursley on the cheek and tried to kiss dudley good bye but missed because dudley was now having a tantrum and throwing his cereal at the walls little tyke chortled mr dursley as he left the house he got into his car and backed out of number fours drive it was on the corner of the street that he noticed the first sign of something peculiar a cat reading a map for a second mr dursley didnt realize what he had seen then he jerked his head around to look again there was a tabby cat standing on the corner of privet drive but there wasnt a map in sight what could he have been thinking of it must have been a trick of the light mr dursley blinked and stared at the cat it stared back as mr dursley drove around the corner and up the road he watched the cat in his mirror it was now reading the sign that said privet drive no looking at the sign cats couldnt read maps or signs mr dursley gave himself a little shake and put the cat out of his mind as he drove toward town he thought of nothing except a large order of drills he was hoping to get that day but on the edge of town drills were driven out of his mind by something else as he sat in the usual morning traffic jam he couldnt help noticing that there seemed to be a lot of strangely dressed people about people in cloaks mr dursley couldnt bear people who dressed in funny clothes the getups you saw on young people he supposed this was some stupid new fashion he drummed his fingers on the steering wheel and his eyes fell on a huddle of these weirdos standing quite close by they were whispering excitedly together mr dursley was enraged to see that a couple of them werent young at all why that man had to be older than he was and wearing an emerald green cloak the nerve of him but then it struck mr dursley that this was probably some silly stunt these people were obviously collecting for something yes that would be it the traffic moved on and a few minutes later mr dursley arrived in the grunnings parking lot his mind back on drills mr dursley always sat with his back to the window in his office on the ninth floor if he hadnt he might have found it harder to concentrate on drills that morning he didnt see the owls swooping past in broad daylight though people down in the street did they pointed and gazed open mouthed as owl after owl sped overhead most of them had never seen an owl even at nighttime mr dursley however had a perfectly normal owl free morning he yelled at five different people he made several important telephone

6.1.3 j.txt (Finnegan's Wake)

riverrun past eve and adams from swerve of shore to bend of bay brings us by a commodius vicus of recirculation back to howth castle and environs sir tristram violer damores frover the short sea had passencore rearrived from north armorica on this side the scraggy isthmus of europe minor to wielderfight his penisolate war nor had topsawyers rocks by the stream oconee exaggerated themselse to laurens countys gorgios while they went doublin their mumper all the time nor avoice from afire bellowsed mishe mishe to tauftauf thuartpeatrick not yet though venissoon after had a kidscad buttended a bland old isaac not yet though alls fair in vanessy were sosie sesthers wroth with twone nathandjoe rot a peck of pas malt had jhem or shen brewed by arclight and rory end to the regginbrow was to be seen ringsome on the aquaface the fall bababadalgharaghtakamminarronnkonnbronntonner ronntuonnthunntrovarrhounawnskawntoohoohoordenenthurnuk of a once wallstrait oldparr is retaled early in bed and later on life down through all christian minstrelsy the great fall of the offwall entailed at such short notice the pftjschute of finnegan erse solid man that the humptyhillhead of humself prumptly sends an unquiring one well to the west in quest of his tumptytumtoes and their upturnpikepointandplace is at the knock out in the park where oranges have been laid to rust upon the green since devlinsfirst loved livvy what clashes here of wills gen wonts oystrygods gaggin fishy gods brekkek kekkek kekkek kekkek koaz koaz koaz ualu ualu ualu quaouauh where the baddelaries partisans are still out to mathmaster malachus micgranes and the verdons catapelting the camibalistics out of the whoyteboyce of hoodie head assiegates and boomeringstroms sod s brood be me fear sanglorians save arms apeal with larms appalling killykillkilly a toll a toll what chance cuddleys what cashels aired and ventilated what bidimetoloves sinduced by what tegotetabsolvers what true feeling for their s hayair with what strawng voice of false jiccup o here here how hoth sprowled met the duskt the father of fornicationists but o my shining stars and body how hath fanespanned most high heaven the skysign of soft advertisement but was ix iseut ere were sewers the oaks of ald now they lie in peat yet elms leap where askes lay phall if you but will rise you must and none so soon either shall the pharce for the nunce come to a setdown secular phoenish bygmester finnegan of the stuttering hand freemen s maurer lived in the broadest way immarginable in his rushlit toofarback for messuages before joshuan judges had given us numbers or helviticus committed deuteronomy one yeastyday he sternely struzk his tete in a tub for to watsch the future of his fates but ere he swiftly stook it out again by the might of moses the very water was eviparated and all the guenneses had met their ezodus so that ought to show you what a pentschanjeuchy chap he was and during mighty odd years this man of hod cement and edifices in toper s thorp piled buildung supra buildung pon the banks for the livers by the soangso he addle liddle phifie annie ugged the little craythur wither hayre in honds tuck up your part inher oftwhile balbulous mithre ahead with goodly trowel in grasp and ivoroiled overalls which he habitacularly fondseed like haroun childeric eggeberth he would caligulate by multiplicables the alltitude and malltitude until he seesaw by neatlight of the liquor wheretwin twas born his roundhead staple of other days to rise in undress maisonry upstanded joygrantit a waalworth of a skyerscape of most eyeful hoyth entowerly erigenating from nezt to nothing and celescalating the himals and all hierarchitectitiptitoploftical with a burning bush abob off its baubletop and with larrons o toolers clittering up and tombles a buckets clottering down of the first was he to bare arms and a name wassaily booslaeugh of riesengeborg his crest of huroldry in vert with ancillars troublant argent a hegoak poursuivant horrid horned his scutschum fessed with archers strung helio of the second hootch is for husbandman handling his hoe hohohoho mister finn youre going to be mister finnagain comeday morm and o youre vine senddays eve and ah youre vinegar hahahaha mister funn youre going to be fined again

Why did Joyce write like this?

6.2 Code

The code is linked at this Github repository.