Markov Chains and Cryptography

Maggie Smith, Yuqi Zhang, Nhi Nguyen

Simple substitution cipher

ABCDEFGHIJKLMNOPQRSTUVWXYZ

THIS MESSAGE SHOULD BE SECRET

DURTYVBQASPLCIXWEGJHZKNFMO

HQAJ CYJJDBY JQXZLT UY JTRGTH

Encryption and decryption

Encryption key: substitutes letters in the original message

f = {DURTYVBQASPLCIXWEGJHZKNFMO}

Decryption key: inverse permutation of the encryption key

g = {IGMAQXRTNSVLYWZKHCJDBFPOEU}

These are bijective functions!

f(A) = D, g(D) = A

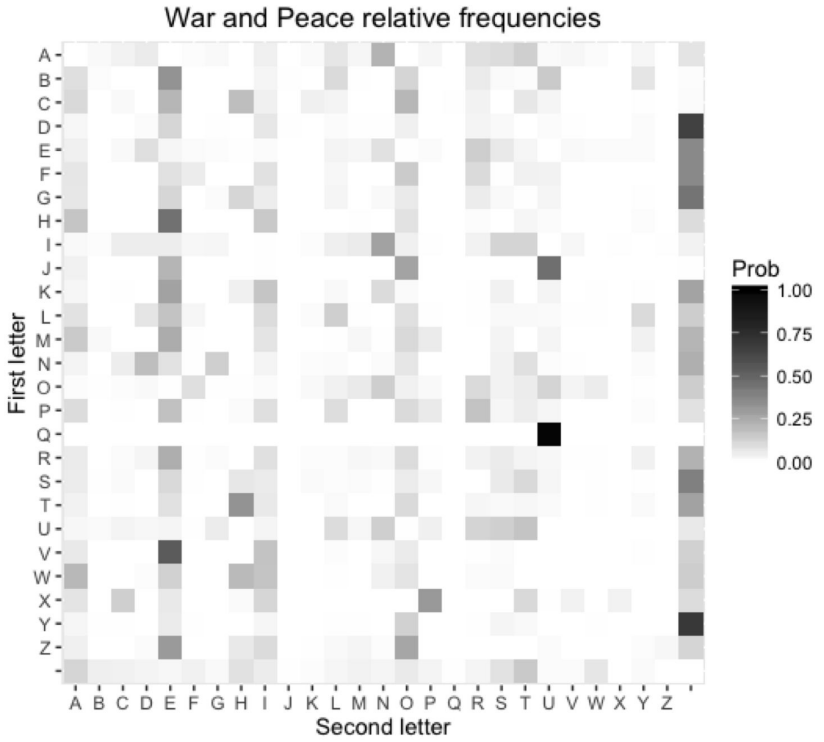
Algorithm introduction

Our goal is to define a probability distribution π on the set of alphabetic permutations S_{26} that gives a high probability to a decryption key *f* if the decrypted key makes sense

This π is the stationary distribution of a Markov chain X_n with state space S_{26} using the Metropolis-Hastings algorithm

We will then run a long-term simulation of this Markov chain to determine which key the chain is most likely or visit, or which state *f* has the highest probability π_f of making sense

Quantification of relative alphabetic frequencies



Score calculation

- How do we determine if the decrypted text makes sense?
- We can compute a score that quantifies how well the consecutive letter pairs in our candidate decrypted text matches the frequencies of that pair in the reference text.

$$\text{score}(f) = \prod_{i=1}^{m-1} M_{f(c_i)f(c_{i+1})};$$

, where m is the number of characters in the encrypted message.

- This value will be high if the consecutive characters f(ci) and f(ci+1) in our decrypted message occur frequently in English.

Score calculation: example

Suppose we have two candidate decrypted texts: “FIHM” and “THIS”. Score calculation tells us which one is a better!

```{r}
source("http://tchumley.mtholyoke.edu/m339sp/project/cryptography-functions.R")
M <- read.table("http://tchumley.mtholyoke.edu/m339sp/project/war-and-peace-matrix.txt")
M["F", "I"] %*% M["I", "H"] %*% M["H", "M"]
M["T", "H"] %*% M["H", "I"] %*% M["I", "S"]

```

```
         [,1]
[1,] 10561278
             [,1]
[1,] 4.022355e+13
```

Score calculation

Our goal is to find the coding function with maximum score!

We let

$$\pi_f = C\text{score}(f).$$

where $C = \left(\sum_{h \in S_{26}} \text{score}(h)\right)^{-1}$ is a normalizing constant that makes π a probability distribution.

- From a Monte Carlo perspective, we want to sample from π, with the idea that a sample is most likely to return a value of high probability.
- By using Metropolis- Hastings algorithm, we do not need to calculate the denominator of πf.

Metropolis-Hastings process for cryptography

1. **Proposal.** Suppose the chain is at state f at time n (X_n = f). Switch 2 random elements of f to get the proposal state f'.
2. **Accept or reject.** Let score(f) and score(f') denote the scores of our current and proposed state.
    a. If score(f') > score(f), accept f' as our new state.
    b. If score(f') < score(f), let p = score(f') / score(f).
        i. Accept f' as our new state with probability p.
        ii. Reject f' as our new state with probability 1-p. In this case the chain stays at the current state f.
3. Repeat the first 2 steps until convergence.

Metropolis-Hastings process for cryptography

Why do we sometimes accept the proposal state f', knowing that it has a lower score than our current state?

By swapping 2 elements, only a small subset of the state space can be reached. If we arrive at a local max and the states surrounding it happen to have lower scores, we will essentially get stuck.

By accepting the proposal state with probability score(f') / score(f), the chain can branch out to a larger number of permutations, which allows the search to be more efficient and thorough.

The acceptance probability is proportional to the score(f') / score(f) ratio, so if f' has a score close to f, it's more likely to be accepted.

R Script results

```
message <- "This is a short message to test the Metropolis Hastings algorithm for cryptography"
```

```
[1] "100000"
[2] "ohis is o shert yassoda te tast tha oatrecenis oostilds onderithy ver bructedrochu"
```

Additional research

Chen and Rosenthal:

- Optimal number of iterations: 10,000
- Cipher text needed: 2,000 characters, though speed is independent
- Trigram attack not as effective

Questions?