2. **[65 points] Decrypting Messages with MCMC**. You are given a passage of English text that has been encrypted by remapping each symbol to a (usually) different one. For example,

$$\begin{aligned} a &\to s \\ b &\to ! \\ \langle\text{space}\rangle &\to v \\ &\vdots \end{aligned}$$

Thus a text like 'a boy...' might be encrypted by 'sv!op...'. Assume that the mapping between symbols is one-to-one. The file **symbols.txt** gives the list of symbols, one per line (the second line is ⟨space⟩). The file **message.txt** gives the encrypted message.

Decoding the message by brute force is impossible, since there are 53 symbols and thus 53! possible permutations to try. Instead we will set up a Markov chain Monte Carlo sampler to find modes in the space of permutations.

We model English text, say $s_1 s_2 \cdots s_n$ where s_i are symbols, as a Markov chain, so that each symbol is independent of the preceding text given only the symbol before:

$$p(s_1 s_2 \cdots s_n) = p(s_1) \prod_{i=2}^{n} p(s_i | s_{i-1})$$

ψ : transition
ϕ : equilibrium

(a) Learn the transition statistics of letters and punctuation in English by downloading a large text [say *War and Peace* (in translation!)] from the web and estimating the transition probabilities $p(s_i = \alpha | s_{i-1} = \beta) \equiv \psi(\alpha, \beta)$ as well as the stationary distribution $\lim_{i\to\infty} p(s_i = \gamma) \equiv \phi(\gamma)$. Assume that the first letter of your text (and also that of the encrypted text provided) is itself sampled from the stationary distribution.

Give formulae for the ML estimates of these probabilities as functions of the counts of numbers of occurrences of symbols and pairs of symbols.

Compute the estimated probabilities. You may report the values using a table, Hinton diagram or other method. [6 marks]

(b) The state variable for our MCMC sampler will be the symbol permutation. Let $\sigma(s)$ be the symbol that stands for symbol s in the encrypted text, e.g., $\sigma(a) = s$ and $\sigma(b) =!$ above. Assume a uniform prior distribution over permutations.

Are the latent variables $\sigma(s)$ for different symbols s independent?

Let $e_1 e_2 \cdots e_n$ be an encrypted English text. Write down the joint probability of $e_1 e_2 \cdots e_n$ given σ. [6 marks]

(c) We use a Metropolis-Hastings (MH) chain, with the proposal given by choosing two symbols s and s' at random and swapping the corresponding encrypted symbols $\sigma(s)$ and $\sigma(s')$.

How does the proposal probability $S(\sigma \to \sigma')$ depend on the permutations σ and σ'? What is the MH acceptance probability for a given proposal? [10 marks]

(d) Implement the MH sampler, and run it on the provided encrypted text. Report the current decryption of the first 60 symbols after every 100 iterations. Your Markov chain should converge to give you a fairly sensible message. (Hint: it may help to initialize your chain intelligently and to try multiple times; in any case, please describe what you did). [30 marks]

(e) Note that some $\psi(\alpha, \beta)$ values may be zero. Does this affect the ergodicity of the chain? If the chain remains ergodic, give a proof; if not, explain and describe how you can restore ergodicity. [5 marks]

(f) Analyse this approach to decoding. For instance, would symbol probabilities alone (rather than transitions) be sufficient? If we used a second order Markov chain for English text, what problems might we encounter? Will it work if the encryption scheme allows two symbols to be mapped to the same encrypted value? Would it work for Chinese with > 10000 symbols? [8 marks]

$P(X_n | X_{n-1})$
$P(X_n | X_{n-1}, X_{n-2})$

(a) Model: $P(S_1 S_2 \cdots S_n) = P(S_1) \prod_{i=2}^{n} P(S_i | S_{i-1})$

Let $\tilde{S}$ be the set of all symbols. $k = |\tilde{S}|$ is the cardinality of $\tilde{S}$, i.e. k is the number of symbols.

Let $N : \tilde{S} \to \mathbb{N}^+$ be the number of appearance of symbols in "near-and-dear".

The transition matrix $\underline{\underline{\Psi}} \in \mathbb{R}^{k \times k}$ satisfies:

$$P(S_i = \alpha | S_{i-1} = \beta) = \psi(\alpha, \beta) = \frac{N(\alpha\beta)}{\sum_{\beta \in \tilde{S}} N(\alpha\beta)} = \frac{N(\alpha\beta)}{N(\alpha)}.$$

The equilibrium distribution $\underline{\phi}$ satisfies $\underline{\phi}\,\underline{\underline{\Psi}} = \underline{\phi}$

(b) The latent variable $\sigma(s)$ for different s are not independent.

$S_1 S_2 \cdots S_n$: decrypted text. $\quad \sigma : e_i \mapsto S_i \quad \forall i \in \{1, \cdots, n\}$.

$e_1 e_2 \cdots e_n$: encrypted text.

"likelihood of σ" $P(e_1 e_2 \cdots e_n | \sigma) = P(e_1 | \sigma) \prod_{i=2}^{n} P(e_i | e_{i-1}, \sigma)$. ---- ✡

(c) $S(\sigma \to \sigma') = \begin{cases} \frac{1}{\binom{n}{2}} & \text{if } \sigma \leftrightarrow \sigma' \\ 0 & \text{otherwise} \end{cases}$, where σ' is the given proposal

Note that: $S(\sigma \to \sigma') = S(\sigma' \to \sigma)$.

Then consider the "likelihood" of a key σ:

$$\phi(\sigma) = P(\sigma(e_1)) \cdot \prod_{i=2}^{n} P(\sigma(e_i) | \sigma(e_{i-1}))$$

$$= P(S_1) \cdot \prod_{i=2}^{n} P(S_i | S_{i-1}).$$

Accept probability $= \min\left(1, \frac{\phi(\sigma) S(\sigma \to \sigma')}{\phi(\sigma') S(\sigma' \to \sigma)}\right)$.

where $\frac{\phi(\sigma) S(\sigma \to \sigma')}{\phi(\sigma') S(\sigma' \to \sigma)} = \frac{\phi(\sigma)}{\phi(\sigma')} = \frac{P(\sigma(e_1))}{P(\sigma'(e_1))} \prod_{i=2}^{n} \frac{P(\sigma(e_i) | \sigma(e_{i-1}))}{P(\sigma'(e_i) | \sigma'(e_{i-1}))}$

$(S(\sigma \to \sigma') = S(\sigma' \to \sigma))$ $= \frac{P(e_1 | \sigma)}{P(e_1 | \sigma')} \prod_{i=2}^{n} \frac{P(e_i | e_{i-1}, \sigma)}{P(e_i | e_{i-1}, \sigma')}$.

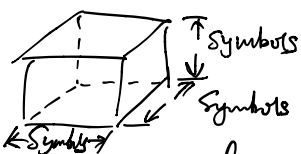
Note: $\phi(\sigma) \equiv P(e_1 e_2 \cdots e_n | \sigma)$.

(e) Zeros in the transition matrix do affect ergodicity of the chain.
The chain becomes non-ergodic with those zeros.
Solution: After we learnt transition matrix $\underline{\Psi}$ from "war-and-peace",
we add 10^{-3} to each element in the transition matrix.

python code: `equ_mtx = equ_mtx + 0.0001`.

This restores ergodicity.

(f) * If we just use symbols probability alone, this is not enough.
The English language rule heavily depends on the combinations
of letters. If we just use symbols probability, we lost information
compared with using symbol combinations.
The decryption result will be meaningless

* First order MC: $\mathcal{P}(X_t | X_{t-1}, \cdots X_1) = \mathcal{P}(X_t | X_{t-1})$.
Second order MC: $\mathcal{P}(X_t | X_{t-1}, \cdots, X_1) = \mathcal{P}(X_t | X_{t-1}, X_{t-2})$.
If we use Second order MC, then transition matrix will
be a tensor. 

Then, the computation will be more complex,

* Allow two symbols map to the same encrypted value?
NO, it doesn't work. Equation ⑥ depends on the fact that
$\mathcal{G}$ is a bijection. (one-to-one + onto). If we allow multiple map,
Equation ⑥ will fail, so as the following steps in Metropolis-Hasting.

* Work for Chinese with > 10000 characters?
No, there are too many characters, so the transition matrix
will be extremely large and sparse.
This will cause computational problems.