6.437

Evan Tey

April 2020

Problem 1: Bayesian framework

(a)

$$\begin{aligned} p_{\mathbf{Y}|\mathsf{f}}(\boldsymbol{y}|f) &= p_{\mathsf{Y}_{1\dots N-1}|\mathsf{f}}(y_{1\dots N-1}|f)\; p_{\mathsf{Y}_N|\mathsf{Y}_{N-1},\mathsf{f}}(y_N|y_{N-1},f) \\ &= p_{\mathsf{Y}_1|\mathsf{f}}(y_1|f)\prod_{i=2}^{n} p_{\mathsf{Y}_i|\mathsf{Y}_{i-1},\mathsf{f}}(y_i|y_{i-1},f) \\ &= P_{f^{-1}(y_1)}\prod_{i=2}^{n} M_{f^{-1}(y_i),f^{-1}(y_{i-1})} \end{aligned}$$

where the conditioning on $\mathsf{Y}_{1\dots N-2}$ disappears in the first line because the letters were generated from a Markov chain. The second line comes from the expansion of $p_{\mathsf{Y}_{1\dots N-1}|\mathsf{f}}$, and the third line comes from rewriting

$$\begin{aligned} p_{\mathsf{Y}_i|\mathsf{Y}_{i-1},\mathsf{f}}(y_i|y_{i-1},f) &= P(\mathsf{x}_k = f^{-1}(y_i)|\mathsf{x}_{k-1} = f^{-1}(y_{i-1})) = M_{f^{-1}(y_i),f^{-1}(y_{i-1})} \\ p_{\mathsf{Y}_i|\mathsf{f}}(y_i|f) &= P(\mathsf{x}_i = f^{-1}(y_i)) = P_{f^{-1}(y_i)} \end{aligned}$$

(b)

$$\begin{aligned} p_{\mathsf{f}|\mathbf{Y}}(f|\boldsymbol{y}) &= \frac{p_{\mathbf{Y}|\mathsf{f}}(\boldsymbol{y}|f)\; p_{\mathsf{f}}(f)}{p_{\mathbf{Y}}(\boldsymbol{y})} \\ &= \frac{p_{\mathbf{Y}|\mathsf{f}}(\boldsymbol{y}|f)\,\frac{1}{|A|!}}{\sum_{f'} p_{\mathbf{Y}|\mathsf{f}}(\boldsymbol{y}|f')\;\frac{1}{|A|!}} \\ &\propto p_{\mathbf{Y}|\mathsf{f}}(\boldsymbol{y}|f) \end{aligned}$$

The prior on f becomes a constant $\frac{1}{|A|!}$ because we assume a uniform distribution over permutations. Since the posterior is just proportional to the likelihood, the MAP estimate, $\hat{\mathsf{f}}_{MAP}(\boldsymbol{y}) = \hat{\mathsf{f}}_{ML}(\boldsymbol{y}) = \operatorname{argmax}_f p_{\mathbf{Y}|\mathsf{f}}(\boldsymbol{y}|f)$.

(c) Direct evaluation of $\hat{\mathsf{f}}_{MAP}(\boldsymbol{y})$ is computationally infeasible because the denominator requires a sum over all possible permutations f'. Since $|A| = 28$, we'd have to evaluate the likelihood 28! times.

Problem 2: Markov chain Monte Carlo method

(a) Given permutation function f, we can generate a permutation function differing in exactly 2 symbols by swapping the outputs of two symbols from A. Since there are $\binom{|A|}{2}$ ways to select which symbols to swap, there are $\binom{|A|}{2}$ permutation functions that differ from f in exactly 2 symbols. Since there are $|A|!$ total functions,

$$P(\text{sampling a function that differs in two symbols}) = \frac{\binom{|A|}{2}}{|A|!} \approx 1.24 \times 10^{-27}$$

(b) Define a proposal distribution $V(x'|x)$ as follows:

$$V(f'|f) = \begin{cases} \frac{1}{\binom{28}{2}} & \text{if f' and f differ in exactly 2 symbols} \\ 0 & \text{otherwise} \end{cases}$$

Then, the acceptance probability is:

$$a(f \to f') = \min\left(1, \frac{p(f')V(f|f')}{p(f)V(f'|f)}\right) = \min\left(1, \frac{p(f')}{p(f)}\right)$$

where $p(f)$ is the posterior $p_{\mathbf{f}|\mathbf{Y}}(f|\boldsymbol{y})$. Together, this gives the effective transition distribution:

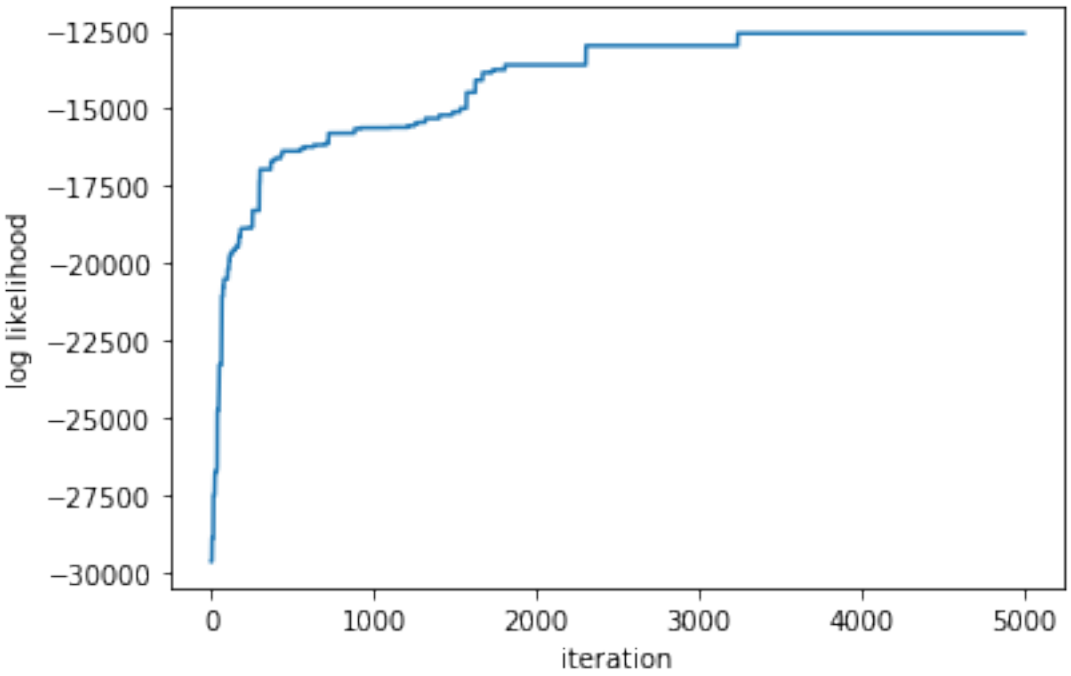
$$W(f'|f) = V(f'|f)a(f \to f')$$

which defines a Markov chain over permutation functions with stationary distribution $p_{\mathbf{f}|\mathbf{Y}}$.

(c) An MCMC algorithm would proceed as follows:

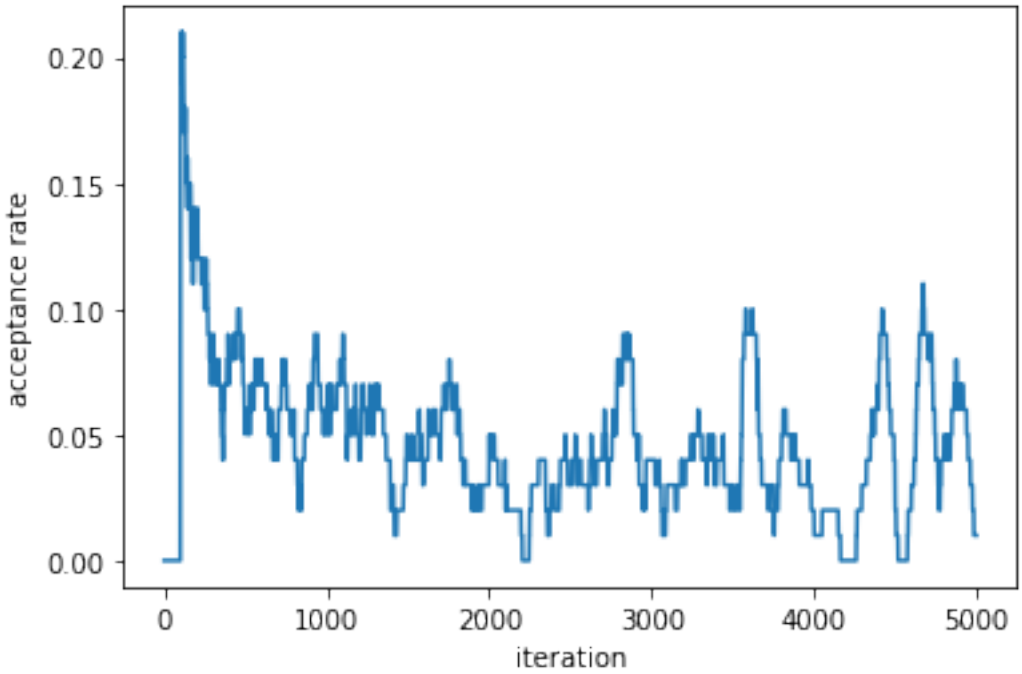
1. Choose an arbitrary initial permutation function f_0
2. For n in $[0, \ldots, T]$: (for some number of iterations T)
   (i) Generate a f' according to $V(f'|f_n)$ by sampling uniformly over the $\binom{28}{2}$ functions that differ from f_n in exactly two positions.
   (ii) Compute $a(f_n \to f')$ according to Problem 2(b) and Problem 1(a).
   (iii) Sample a $u \sim \text{Unif}(0, 1)$.
   (iv) If $u \leq a(f_n \to f')$, set $f_{n+1} = f'$. Otherwise reject f'.
3. Choose the most frequently occurring f_i as $\hat{f}_{MAP}$

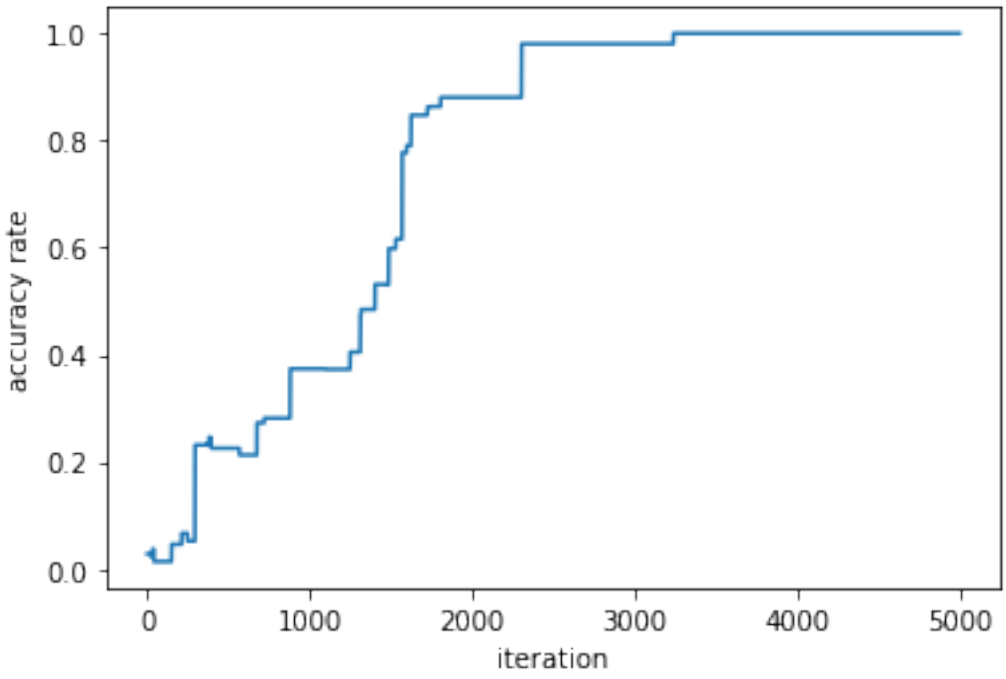
Problem 3: Implementation



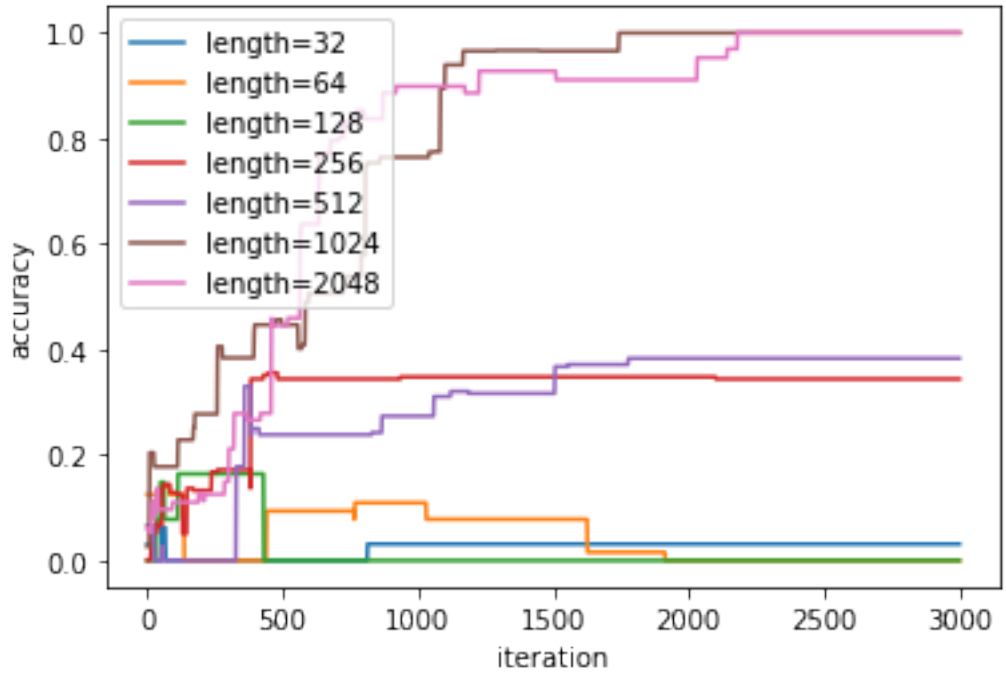
(a)

(b) I use a window size of $T = 100$.



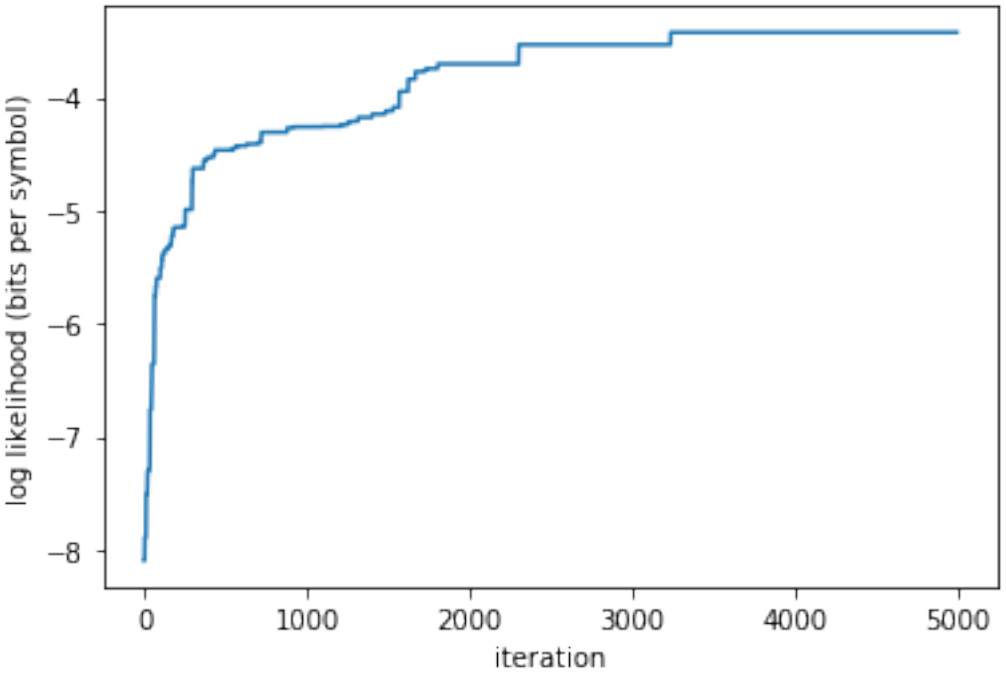


(c)



(d)

I tried partitioning the ciphertext into segments with lengths going up by powers of two. Roughly, it appears that accuracy increases faster when ciphertexts are longer. This makes sense because it's easier for shorter texts to settle into some local maxima. Another way of seeing this is that longer ciphertexts (with more digrams) give a more accurate depiction of the likelihood (it's less dependent on the particular digrams that appear in the sequence).



(e)

When converted into units of bits per symbol, our log likelihood graph appears to approach an approximate value of -3.5. This appears to match the negative F_2 value in Shannon's "Prediction and Entropy of Printed English." This makes sense because our likelihood function essentially takes the likelihood of each *digram* in our text independently. If our text is long enough (if we have enough

digrams), we're essentially evaluating the expected log probability over all digrams. This is negative the digram entropy as described in F_2. (Note: The cited value is for a 26-letter alphabet and we use a 28-letter alphabet, so we'd probably expect the true entropy to be slightly higher. We're also only using a shorter ciphertext, so we'd expect our measured entropy to be slightly lower.)