6.7800 Final Project

Annie Feng `azf@mit.edu`

Project Part I Annie Feng

1. a). $x_j = f^{-1}(y_i)$. f is one-to-one. y is length n.

$$P_{Y|f}(y|f) = \mathbb{P}(x_1 = f^{-1}(y_1)) \cdot \mathbb{P}(x_2 = f^{-1}(y_2) | y_1) \cdots \mathbb{P}(x_n = f^{-1}(y_n) | y_{n-1})$$

$$= P_{f^{-1}(y_1)} \cdot \prod_{j=2}^{n} M_{f^{-1}(y_j), f^{-1}(y_{j-1})}$$

where $P_i = \mathbb{P}(x_k = i)$, and $[M_{ij}] = M$ is the transition matrix.

b) $$P_{f|y}(f|y) = \frac{P_{Y|f}(y|f) \cdot P_f(f)}{\sum_f P_{Y|f}(y|f) \cdot P_f(f)} = \frac{\frac{1}{m!} P_{Y|f}(y|f)}{\frac{1}{m!} \sum_{i=1}^{m!} P_{Y|f}(y|f_i)}$$

$\sim U(\frac{1}{m!})$

$$= \frac{P_{Y|f}(y|f)}{\sum_{i=1}^{m!} P_{Y|f}(y|f_i)}$$, where $P_{Y|f}(y|f)$ is defined in part a.

$$\hat{f}_{MAP}(y) = \underset{f \in \mathcal{F}}{\arg\max}\, P_{f|y}(f|y) = \underset{f \in \mathcal{F}}{\arg\max} \frac{P_{Y|f}(y|f)}{\sum_{i=1}^{m} P_{Y|f}(y|f_i)}$$

$\mathcal{F} = \{ f_i,\ i = 1 \ldots m! \}$, each f_i is a permutation of the m symbols in alphabet A.

c) To get $P_{Y|f}$, this is $O(n)$. Since $|\mathcal{F}| = m!$, the denominator of MAP estimator takes $O(nm!)$, which is computationally infeasible for even moderate-size alphabets A ($28! \approx 10^{29}$), and English texts can be long in n, the character-length.

2.

a) $\dfrac{\binom{m}{2}}{m!}$

$\sum_{i=1}^{m!} \mathbb{P}(f_1 = f_i) \cdot \mathbb{P}(f_2 \text{ differs from } f_i \text{ in 2 places})$

$= \sum_{i=1}^{m!} \dfrac{1}{m!} \dfrac{\binom{m}{2}}{m!} = \dfrac{\binom{m}{2}}{m!}$

b) $v(f_2 \mid f_1) = \begin{cases} \dfrac{1}{\binom{m}{2}} & \text{if } f_1 \text{ and } f_2 \text{ differ in only 2 symbol assignments} \\ 0 & \text{else} \end{cases}$

proposal distr. satisfies Corollaries 20.1 - 20.3 to converge correctly:

1. $v(\cdot|\cdot)$ is fully comm.
2. $v(x|x') = 0$ iff $v(x'|x) = 0$
3. $a(x \to x') < 1$ for some distinct x, x':

   consider $\hat{f}_{MAP}$ and a f that differs in 2 symbols.

   $p(\hat{f}_{MAP}|y) > p(f|y)$ so $a(f_{MAP} \to f) < 1$.

Markov Chain: $m!$ nodes: $f_1 \ldots f_{m!}$

edges between f_i and f_j if they differ in exactly 2 symbol assignments.

If $p_{Y|F}(y|f_j) < p_{Y|F}(y|f_i)$, then

$f_i \to f_j$ has transition prob $\dfrac{p_{Y|F}(y|f_j)}{p_{Y|F}(y|f_i)} \cdot \dfrac{1}{\binom{m}{2}} = \omega(f_j|f_i)$

$f_j \to f_i$ has transition prob $\dfrac{1}{\binom{m}{2}} = \omega(f_i|f_j)$

and to normalize, $f_i \to f_i$ has transition prob
(add self-loops)

$$w(f_i \mid f_i) = 1 - \sum_{f_k \neq f_i \in \mathcal{F}} w(f_k \mid f_i)$$

and similarly for $f_j \to f_j$,

$$w(f_j \mid f_j) = 1 - \sum_{f_k \neq f_j} w(f_k \mid f_j).$$

In the MH alg, $p_0(f \mid y) = P_{Y|F}(y \mid f)$.

since $$P_{F|Y}(\cdot \mid y) = \frac{P_{Y|F}(y \mid f)}{\sum_{i=1}^{m!} P_{Y|F}(y \mid f_i)} \propto p_0(f \mid y).$$

Part I

Problem 2

c)

Algorithm 1 Decoding with MCMC

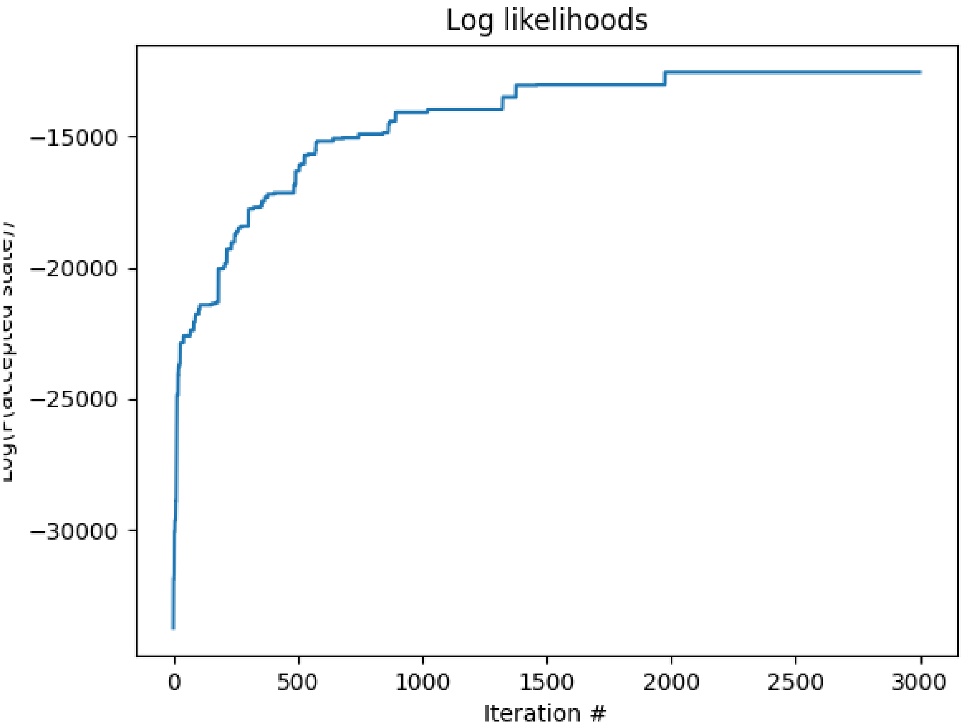
```
procedure DECODE_MAIN(ciphertext)
    f ← randomly init a permutation of length 28 numpy.ndarray
    for n iterations do
        x' ← sample_from_proposal(x)
        u ← compute_acceptance(x, x')
        a ← sample_from_uniform(0, 1)
        if a ≤ u then
            x ← x'
        else
            x ← x
    return get_plaintext(x, ciphertext)
```

Algorithm 2 Helper methods

```
procedure SAMPLE_FROM_PROPOSAL(x)
    (i, j) ← uniformly sample 2 indices from range(28) without replacement
    x' ← x.copy()
    x'[i] ← x[j]
    x'[j] ← x[i] return x'
procedure GET_PLAINTEXT(f, ciphertext)
    return [inverse(f, c) for c in ciphertext ].to_string()
procedure COMPUTE_ACCEPTANCE(f,f')
    p_{y|f} ← 0                                          ▷ Compute products in log-space
    p_{y|f'} ← 0
    for i in range(len(ciphertext)) do                   ▷ Possible to vectorize instead of for-loop
        if i == 0 then
            p_{y|f} ← p_{y|f} + log(P[inverse(f, ciphertext[i])])
            p_{y|f'} ← p_{y|f'} + log(P[inverse(f', ciphertext[i])])
        else
            p_{y|f} ← p_{y|f} + log(M[inverse(f, ciphertext[i]), inverse(f, ciphertext[i − 1])])
            p_{y|f'} ← p_{y|f'} + log(M[inverse(f', ciphertext[i]), inverse(f', ciphertext[i − 1])])
    a ← min(1, exp(p_{y|f'} − p_{y|f}))
    return a
```

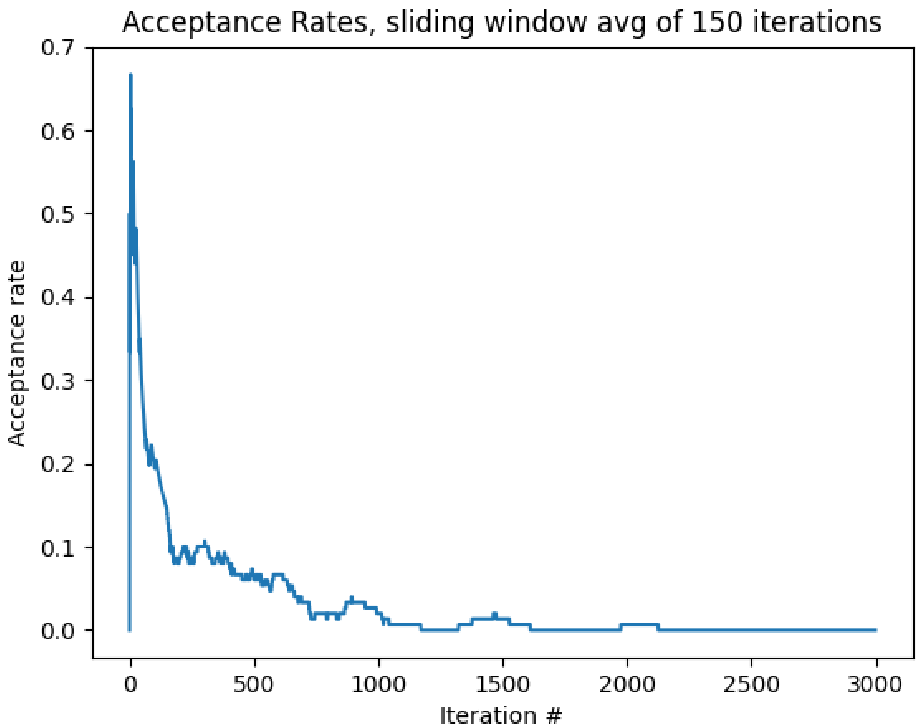
Problem 3

a)

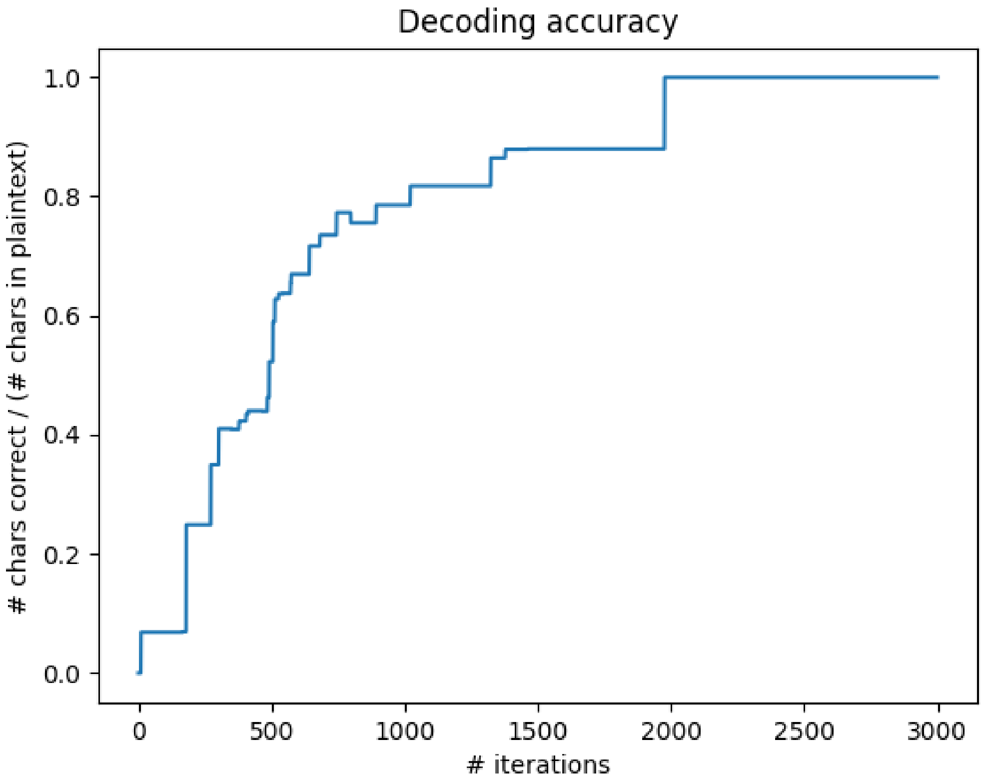


b)

T=150 in the plot below.

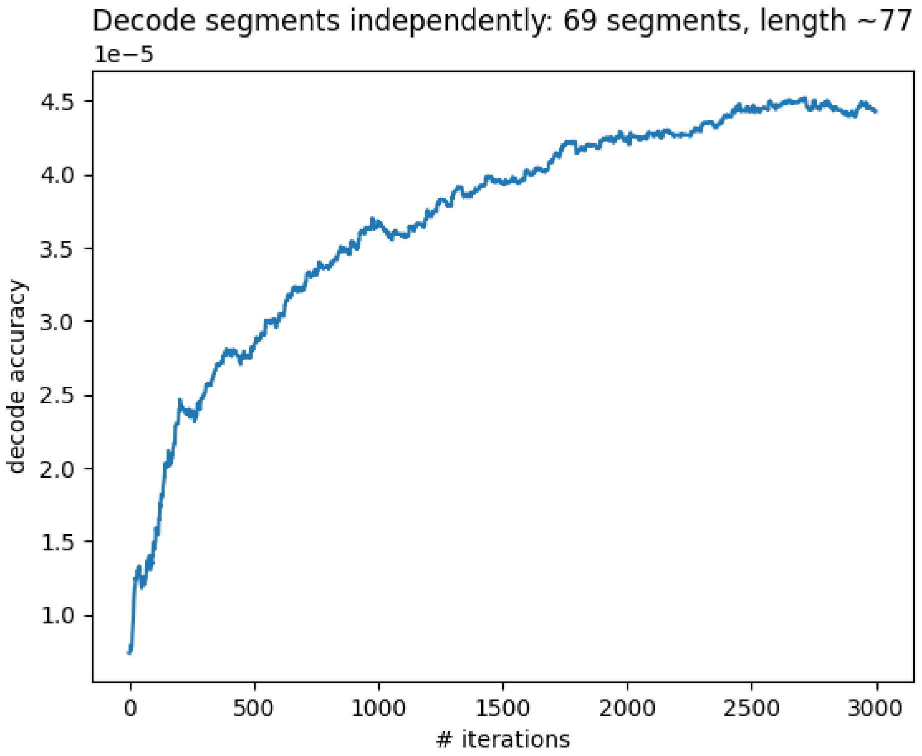


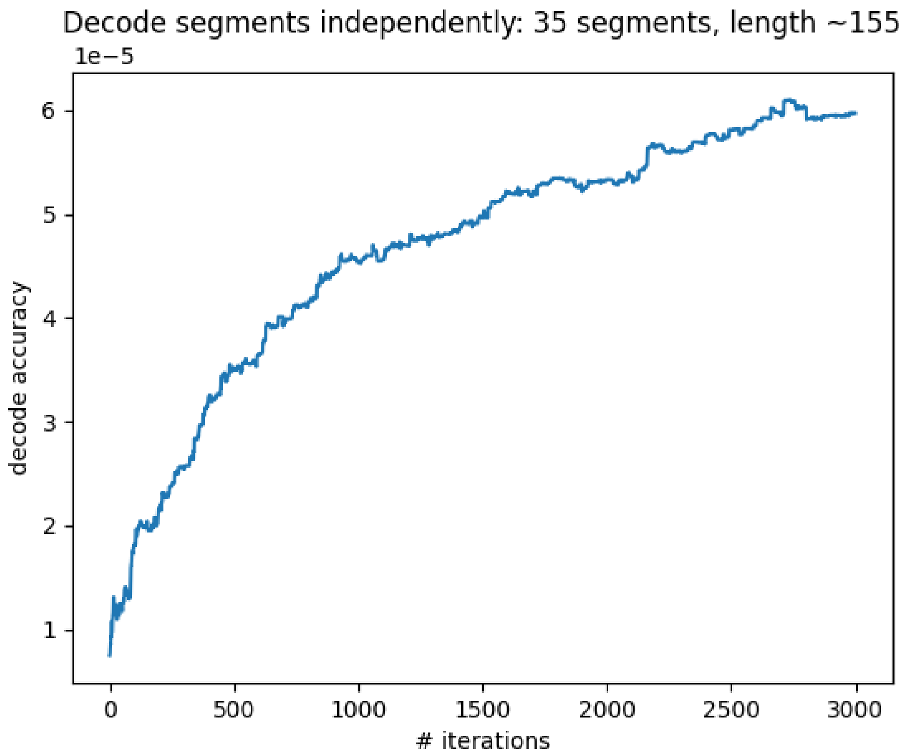
c)

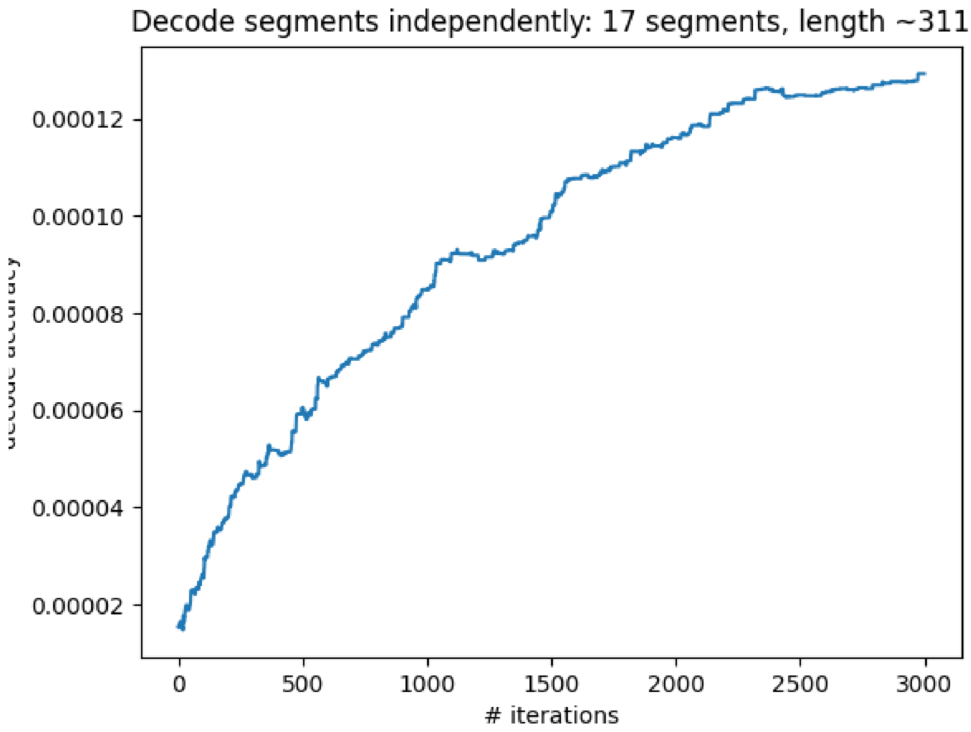


d)

As seen in the plots below, as the segment size increases, the decoding accuracy increases. There are more "English rules" encoded in increasing number of character occurrences; as there are more characters, more constraints appear. Thus, since locally there may be several permutations (f) that achieve higher likelihood, maximizing the likelihood globally is better than maximizing likelihood locally because there will be fewer permutations that maximize the objective, and greater chance to pick the correct decoder.


Decode segments independently: 69 segments, length ~77
1e−5
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
decode accuracy
0
500
1000
1500
2000
2500
3000
iterations

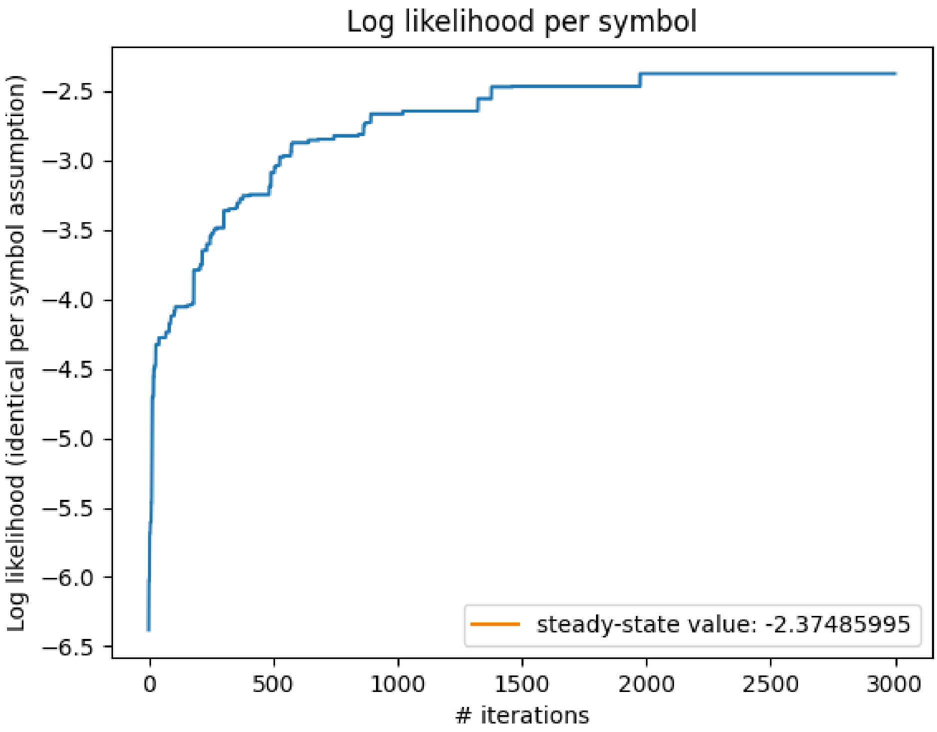

Decode segments independently: 35 segments, length ~155
1e−5
6
5
4
3
2
1
decode accuracy
0
500
1000
1500
2000
2500
3000
iterations



e)

With an i.i.d. assumption of the 28 symbols (letters, space, and period), the log likelihood should be $log_2(\frac{1}{28}) \approx -4.8$. However, as seen in the attached plot, the steady-state value for the log-likelihood is about -2.37 experimentally on the "ciphertext.txt" corpus. As the letters become less uniformly distributed (less random), the log-likelihood increases, so this log-likelihood comparison makes sense, since English characters are not uniformly distributed (there are grammar constraints, for example).

Compared to the entropy of English by Claude Shannon, this experimental value is also reasonable. In Shannon's paper, "Prediction and Entropy of Printed English", he estimated the 27-letter alphabet (with a space) to have entropy of 2.14, and the 26-letter alphabet to have entropy of 2.62. The negative log likelihood in this experiment with a 28-letter alphabet was 2.37. So, Shannon's claim that the redundancy of English is about 50 percent is verified in this experiment (-4.8 to -2.37 reduction in log-likelihood).



Log likelihood per symbol
Log likelihood (identical per symbol assumption)
−2.5
−3.0
−3.5
−4.0
−4.5
−5.0
−5.5
−6.0
−6.5
0
500
1000
1500
2000
2500
3000
iterations
steady-state value: -2.37485995