Output text files are all stored in saved_output folder
You can re-run the python files (instructions in following) to generate these output files in server.

Part1:

Python code see learn_motif.py

python3 learn_motif.py --width=6 --positions="example1_positions.txt" --model="example1_model.txt" --subseqs="example1_subseqs.txt" "example1.txt"

It will take 5-10 min to finish running for the given two examples.

The model/PWM will show in example1_model.txt
Start Positions will show in example1_positions.xtx
The subsequences will show in example1_subseqs.txt

Part2

Python code see part2_count_gibbs.py
To run python code, in the working directory with **seq.txt** (with the sequences saved):
----- python part2_count_gibbs.py

Output file is part2_out.txt

(A)
$p_{c,k} = (n_{c,k} + d_c) / (N-1+d_b)$
$p_{c,0} = (n_{c,0} + d_c) / ((N-1)(L-W)+d_b)$

N = 10
L = 8
W = 4

Initial PWM with background, we will fill this prob_matrix based on n_c,k and above formulas:

	0	1	2	3	4
A					
C					
G					
T					

Construct N_table excluding sequence 5:

	0	1	2	3	4
A	**12**	**1**	**5**	**0**	**1**
C	**5**	**5**	**1**	**1**	**3**
G	**12**	**2**	**1**	**7**	**2**
T	**7**	**1**	**2**	**1**	**3**

PWM:

	0	1	2	3	4
A	**0.3250**	**0.1538**	**0.4615**	**0.0769**	**0.1538**
C	**0.1500**	**0.4615**	**0.1538**	**0.1538**	**0.3077**
G	**0.3250**	**0.2308**	**0.1538**	**0.6154**	**0.2308**

T	**0.2000**	**0.1538**	**0.2308**	**0.1538**	**0.3077**

Python output:

updated probability matrix

```
0.325   0.154   0.462   0.077   0.154
0.150   0.462   0.154   0.154   0.308
0.325   0.231   0.154   0.615   0.231
0.200   0.154   0.231   0.154   0.308
```

(B)

For sequence 5:

LR (ai=2) = (pc,1 * pc,2 * pt,3 * pa,4) / (pc,0 * pc,0 * pt,0 * pa,0)
= (0.4615 * 0.1538 * 0.1538 * 0.1538) / (0.1500*0.1500*0.2000*0.3250)
= 1.148

Computer Σ LR (k) = 0.7654 + 1.148 + 1.2925 + 1.7226 + 0.4309 = 5.3594

Probability of (a5 = 2) = 1.148/5.3594 = 0.2142

Python output:

probability of choosing ai=2:

0.216

(C)

Use the same N_table excluding sequence 5:

	0	1	2	3	4
A	**12**	**1**	**5**	**0**	**1**
C	**5**	**5**	**1**	**1**	**3**
G	**12**	**2**	**1**	**7**	**2**
T	**7**	**1**	**2**	**1**	**3**

For palindrome model:

$Pa,1 = Pt,w = (n_{a,1} + n_{t,w} + d_{a,1} + d_{t,w}) / \Sigma ((n_{b,1} + d_{b,1}) \quad + \Sigma (n_{b,1} + d_{b,1}))$

$= (1 + 3 + 1 + 1) / (9 + 4 + 9 + 4) = 0.2308$

	0	1	2	3	4
A	**0.3250**	**0.2308**	**0.3077**	**0.1538**	**0.1538**
C	**0.1500**	**0.3462**	**0.3846**	**0.1538**	**0.2692**
G	**0.3250**	**0.2692**	**0.1538**	**0.3846**	**0.3462**
T	**0.2000**	**0.1538**	**0.1538**	**0.3077**	**0.2308**

Python output:

```
updated palidrome probability matrix
0.325   0.231   0.462   0.077   0.154
0.150   0.346   0.154   0.154   0.269
0.325   0.269   0.154   0.615   0.346
0.200   0.154   0.231   0.154   0.231
```

Part 3

(A)

The log likelihood in the optimal PWM model with width W+1 **will always be more than or equal to** the log likelihood in the optimal model with width W.

Proof

log P(X , Z | p) = $= \sum_i \sum_j Z_{i,j} \log P(X_i \mid Z_{i,j} = 1, p) + n \log \frac{1}{m}$

m = L – W + 1

so with W + 1, we have m updated as m= m-1

(1) i is representing sequence number, no difference between these two models
(2) for j
Initial model: j -> 1,2,….,m
New model: j -> 1,2,…..,m-1
(3) log-likelihood
Initial model: log P_1 = $\Sigma^n \Sigma^m Z_{i,j} \log P(X_i) + n \log (1/m)$
New model: log P_2= $\Sigma^n \Sigma^{m-1} Z_{i,j} \log P(X_i) + n \log(1/ (m-1))$

logP_1 – logP_2
$= \Sigma^n \Sigma^m Z_{i,j} \log P(X_i) + n \log (1/m) - \Sigma^n \Sigma^{m-1} Z_{i,j} \log P(X_i) - n \log(1/ (m-1))$
$= n * (\Sigma^m Z_{i,j} \log P(X_i) - \Sigma^{m-1} Z_{i,j} \log P(X_i) + \log ((m-1)/m))$

Here is the term:

$\log P(Xi \mid Zi,m =1, p)^{Zi,m} + \log (m-1)/m$

both $P(Xi \mid Zi,m =1, p)^{Zi,m}$ and $(m-1)/m <= 1$

so get negative sum or zeor for $\log P(Xi \mid Zi,m =1, p)^{Zi,m} + \log (m-1)/m$
log P_1 – logP_2 <=0
logP_1 <= logP_2

The longer of the motif, the more of the log-likelihood

Part 4

Python code see part4_entropy_cal.py
To run python code, in the working directory with **model2.txt** (from part1):
---- python part4_entropy_cal.py
Output file is part4_logo.txt

(A)

A	0.256	0.104	0.032	0.027	0.057	0.053	0.213	0.076	0.204	0.968	0.964	0.457
C	0.243	0.447	0.019	0.034	0.852	0.396	0.012	0.010	0.010	0.013	0.011	0.039
G	0.257	0.135	0.026	0.338	0.020	0.010	0.365	0.898	0.775	0.010	0.010	0.267
T	0.244	0.314	0.922	0.601	0.071	0.541	0.410	0.016	0.012	0.010	0.015	0.237

$H_{max} = \log_2 N = \log_2 4 = 2$
$H(c) = -\Sigma P(c) \log_2 P(c)$

Height of logo = Hmax – H (c)

[0.21287965 1.58146455 0.73169833 1.23022061 0.73126784 0.39927009
 1.458879 1.09524011 1.73312419 1.73312419 0.33938222]

According to the ratio matrix (entropy at position i for character c) $E_{i,c} / \Sigma E_{i,b}$

entropy_matrix

0.362	0.148	0.144	0.213	0.186	0.474	0.264	0.468	0.068	0.068	0.506
0.523	0.066	0.162	0.205	0.528	0.066	0.066	0.066	0.066	0.066	0.183
0.380	0.100	0.524	0.066	0.066	0.531	0.145	0.304	0.066	0.066	0.497
0.522	0.104	0.437	0.285	0.489	0.530	0.066	0.066	0.066	0.066	0.475

logo matrix

0.213	1.581	0.732	1.230	0.731	0.399	1.459	1.095	1.733	1.733	0.339

individual character matrix

0.043	0.560	0.083	0.341	0.107	0.118	0.711	0.566	0.439	0.439	0.103
0.062	0.251	0.094	0.328	0.304	0.017	0.179	0.080	0.431	0.431	0.037
0.045	0.378	0.302	0.106	0.038	0.132	0.390	0.368	0.431	0.431	0.101
0.062	0.393	0.252	0.455	0.282	0.132	0.179	0.080	0.431	0.431	0.097

(B) see submitted PDF

(C)
According to the logo generated, we can see the positions with higher logo are more likely to be predicted with certain character.
For the same positions, the higher of the character, the probability of the occurrence of that character is higher.
For different positions, the more dispense of the characters, with more equal probability for each character, the lower of the total height of all characters.