Problem 1

MEME Implementation

Write a program, learn_motif.py, that takes as input a set of DNA sequences and an integer width W and learns an OOPS model for a motif of width W.

function OOPS_EXHAUSIVE($input\_seqs, W, \pi_{init}$)
  $bestProb = -\infty$
  for W-mer in input_seqs **do**
    $pwm \leftarrow get\_init\_pwm$(W-mer)
    $Z, prob \leftarrow E\_step\_get\_Z(pwm, sequences)$
    if $prob > bestProb$ **then**
      $best\_Z, best\_prob \leftarrow Z, prob$
      $best\_pwm \leftarrow M\_step\_get\_pwm(Z, sequences)$
    end if
    $pwm, Z \leftarrow best\_pwm, best\_Z$
  end for
  while not Converged **do**
    $Z \leftarrow E\_step\_get\_Z(pwm, sequences)$
    $pwm' \leftarrow M\_step\_get\_pwm(Z, sequences)$
    $Converged \leftarrow |pwm' - pwm| < tol$
    $pwm \leftarrow pwm'$
  end while
  Write $pwm, Z_{i,max}, ...$
end function

Algorithm 1: Exhaustive Start OOPS Algorithm

Problem 2

Motif Discovery

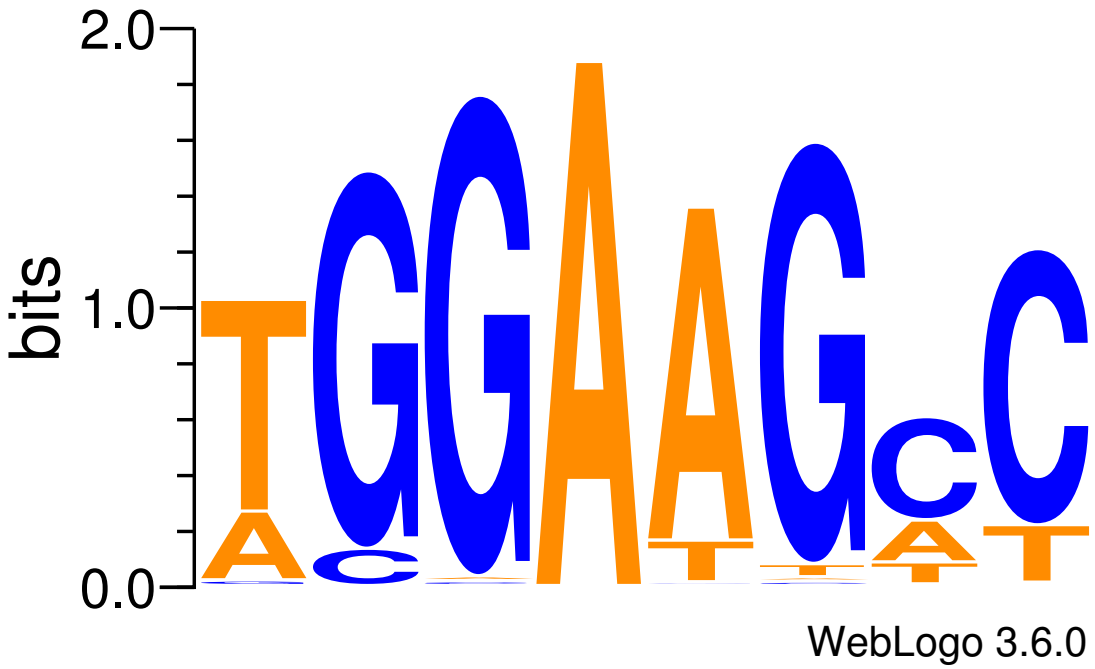
We suppose the input sequences in the file hw1_hidden.txt come from the promoter regions of 100 yeast genes that are differentially expressed in a stress response experiment. We would like to use motif finding to learn more about a candidate regulator (transcription factor) of this expression response. Using our program from part I, we find the best OOPS motif of width 8:

ch	BG	Pos1	Pos2	Pos3	Pos4	Pos5	Pos6	Pos7	Pos8
C	0.256	0.010	0.094	0.019	0.010	0.010	0.019	0.595	0.797
A	0.241	0.240	0.010	0.022	0.970	0.856	0.020	0.251	0.010
G	0.250	0.029	0.886	0.948	0.011	0.020	0.922	0.020	0.020
T	0.252	0.721	0.010	0.011	0.010	0.114	0.040	0.135	0.174

Hidden Motif PWM

Problem 3

Sequence Logo
Express the PWM as a sequence Logo.



Problem 4

Transcription Factor Search
To find a candidate match for our transcription factor, we go to JASPAR, "an open-access database of curated, non-redundant transcription factor (TF) binding profiles." By following the instructions on the alignment page, we see that the top hit, MA0304.1 (GCR1) is a very close match:

Others in the top 5 include MA0377.1 (SFL1), MA0305.1 (GCR2), MA0323.1 (IXR1), and MA0418.1 (YAP6), but they aren't nearly as close. The maximum possible score is two times the length of the two sequences, and the top hit was over 15, compared with 13 for second place.

Problem 4 continued on next page...

There are slight differences. As the Jaspar description is "Finalized", it doesn't need to use pseudo counts, and thus the informational content is marginally higher at every location.
Using the www.yeastgenome.org website, we find GCR1 is a "transcriptional activator of genes involved in glycolysis." From Wikipedia, glycolysis is the metabolic pathway that breaks down glucose into pyruvate, releasing the free energy needed to create the universal fuel ATP. GCR1 is known to regulate 86 genes.

Problem 5

PWM Parameters
We are given the following motif solution, using no pseudocounts:

C**ATGT**GAA
CAG**CAGG**G
A**CCTC**TTC
CAGACATG
AC**CTAT**CG
G**CGGC**AGT
GTGTAGTT
C**CAGG**AAG
AT**GACC**GG
GGA**TAGT**A

	i=1	i=2	i=3	i=4	Background
A	0.1	0.5	0.1	0.1	0.325
C	0.6	0.1	0.1	0.3	0.175
G	0.2	0.1	0.7	0.2	0.325
T	0.1	0.3	0.1	0.4	0.175

A) Extend existing solution to W = 5

	i=1	i=2	i=3	i=4	i=5	Background
A	0.100	0.500	0.100	0.100	0.400	0.300
C	0.600	0.100	0.100	0.300	0.200	0.167
G	0.200	0.100	0.700	0.200	0.300	0.333
T	0.100	0.300	0.100	0.400	0.100	0.200

B) Compare Log Likelihoods

$\log_{10} P(X\|W=4)$	-42.73
$\log_{10} P(X\|W=5)$	-42.57
$\log_{10} P(X\|W=5) - \log_{10} P(X\|W=4)$	0.168

Problem 5 continued on next page...

C) Discussion

Extending the width will always give a likelihood which is greater than or equal to the shorter solution.

To see why, we can first look at the case in which $W << L$, so background probabilities are unchanged by the single extension. Then the probability of each character in each sequence will be unchanged, except for the terms in the $w+1$ motifs position. In the W_0 case, the background probabilities p_{bg_c}, are used, and in the W_0+1 case, the probabilities used are the Dirichlet parameters, p_{D_c}, calculated just for this position. The Dirichlet distribution was been shown to be a maximum likelihood estimator. Thus

$$\prod_{c \in seqs_{W_0+1}} p_{D_c} \geq \prod_{c \in seqs_{W_0+1}} p_{bg_c}$$

because it's greater than or equal for *all* distributions. Therefore

$$\frac{P(X|W = W_0 + 1)}{P(X|W = W_0)} = \prod_{c \in seqs_{W_0+1}} \frac{p_{D_c}}{p_{bg_c}} \geq 1$$

For the case in which the background changes, we note that the new background probabilities, $p_{bg'_c}$ are again MLE estimators of all non-motif characters for the case in which the $W_0 + 1$ characters *are excluded*. So their ratio is again greater than one:

$$\prod_{c \notin \text{motif}_{W_0+1}} \frac{p_{bg'_c}}{p_{bg_c}} \geq 1$$

To complete the argument, we note when two terms that are each ≥ 1 are multiplied, their product is as well. This implies that while maximizing likelihood is able to find the best motif solution for a given width, it is not able to choose the *best* width.

Problem 5 continued on next page...

Listing 1: Problem 5 Code

```
import numpy as np
import itertools

ch_to_n = {el:i for i,el in enumerate('ACGT')}

def get_pwm(seqs,z1,W):
    Nc = {el:0 for el in 'ACGT'}
    for seq in seqs:
        for ch in seq:
            Nc[ch] += 1

    pwm = np.zeros((4,W+1))
    for i, seq in enumerate(seqs):
        for j in range(W):
            pwm[ch_to_n[seq[j + z1[i]]],j+1] += 1

    for ch in ch_to_n:
        pwm[ch_to_n[ch],0] = Nc[ch] - sum(pwm[ch_to_n[ch],1:W+1])

    pwm = pwm/pwm.sum(axis=0)
    return (pwm)

def get_log_prob(seqs,pwm,z1,W):
    log_pwm = np.log10(pwm)
    prob = 0
    L = len(seqs[0])
    for i,seq in enumerate(seqs):
        motif_pos = range(W)
        non_motif_range = itertools.chain(range(z1[i]),range(z1[i]+W,L))
        for j in non_motif_range:
            prob += log_pwm[ch_to_n[seq[j]],0]
        for j in motif_pos:
            prob += log_pwm[ch_to_n[seq[j+z1[i]]],j+1]
    return prob



seqs = [
        'CATGTGAA',
        'CAGCAGGG',
        'ACCTCTTC',
        'CAGACATG',
        'ACCTATCG',
        'GCGGCAGT',
        'GTGTAGTT',
        'CCAGGAAG',
        'ATGACCGG',
        'GGATAGTA']
```

```
z1 = [1,3,1,0,2,1,0,1,2,3]

print (get_pwm(seqs,z1,4))
pwm = get_pwm(seqs,z1,5)
for i, ch in enumerate('ACGT'):
    print (' & '.join(
            [ch]
            + ["{:.3f}".format(el) for el in pwm[i,1:]]
            + ["{:.3f}".format(pwm[i,0])]))
p4 =  get_log_prob(seqs, get_pwm(seqs,z1,4), z1, 4)
p5 =  get_log_prob(seqs, get_pwm(seqs,z1,5), z1, 5)
print("The log likelihood for W=4 solution is {:.2f}".format(p4))
print("The log likelihood for W=5 solution is {:.2f}".format(p5))
print("The difference is {:.3f}".format(p5-p4))
```