HW4 Extra Credit: Motif Finding

EN.600.438 Comptutational Genomics: Data Analysis

Richard Chen

Worked with: Steven Chen, Yunfan Fan

0. Dependencies

In [40]:
```
import numpy as np
import re
import math
```

1. E Step: Estimate γ(Zij) from P

In [41]:
```
def E_Step(W, read_list, read_length, PFM, nucleotides_dict):
    for pos in range(read_length):
        all_curr_pos = "".join([line[pos] for line in read_list])
        for base in range(len(nucleotides_dict.keys())):
            PFM[base][pos] = all_curr_pos.count(nucleotides_dict.ke
ys()[base])
    PFM = PFM/len(read_list)
    PFM_init = PFM[0:4,:W]
    return PFM_init
```

2. M Step: Estimate Pck and Bc using γ(Z)

In [42]:
```
def M_Step(W, read_list, read_length, PFM, nucleotides_dict, nucleo
tides_prob, Pck):
    gamma = np.zeros((len(read_list),read_length))
    for ind in range(len(read_list)):
        for pos in range(read_length-W+1):
            curr_motif = list(read_list[ind][pos:pos+W]); else_seq
= list(read_list[ind][:0] + read_list[ind][W:])
            curr_motif_prob = reduce(lambda x,y: x*y, [Pck[:,i][[nu
cleotides_dict[base] for base in list(curr_motif)][i]] for i in ran
ge(W)])
            else_seq_prob = reduce(lambda x,y: x*y, [nucleotides_pr
ob[base] for base in else_seq])
            gamma[ind,pos] = curr_motif_prob*else_seq_prob
    rowsums = np.sum(gamma, axis = 1)
    gamma = gamma/rowsums[:,None]

    nck = np.zeros((4,W))
    for i in range(len(nucleotides_dict.keys())):
        for ind in range(len(read_list)):
            indices = [m.start() for m in re.finditer(nucleotides_d
ict.keys()[i], read_list[ind][:read_length-W+1])]
            for j in range(W):
                nck[i][j] = nck[i][j] + sum([gamma[ind][index-j] fo
r index in indices])
    Pck = (nck + 1)/(np.sum(nck, axis = 0) + 4)

    B = []
    for ind in range(len(nucleotides_dict.keys())):
        mc = 0
        for read in read_list:
            mc += read.count(nucleotides_dict.keys()[ind])
        gc = mc - sum(nck[ind]) + 1
        B.append(gc)
    denom = sum(B)
    B = B/denom

    return gamma, Pck, B
```

3. Compute log likelihood of the data

In [43]:
```
def loglikelihood(gamma):
    return sum([math.log10(p) for p in np.sum(gamma, axis = 1)])
```

4a. Implement a function findmotif(file name, motif width, iterations) using the EM algo- rithm described above.

In [44]:
```
def findmotif(file_name, motif_width, iterations):
    read_list = [line.rstrip() for line in open(file_name).readline
s()]
    read_length = len(read_list[0]); W = motif_width; iterations =
100; loglik_prev = 0; itr = 1; conv = False

    PFM = np.zeros((4,read_length))
    nucleotides_dict = {'A':0, 'C':1, 'T':2, 'G':3}
    nucleotides_prob = {'A':0.25, 'C':0.25, 'T':0.25, 'G':0.25}

    Pck = E_Step(W, read_list, read_length, PFM, nucleotides_dict)

    while (itr < 100 and conv == False):
        gamma, Pck, B = M_Step(W, read_list, read_length, PFM, nucl
eotides_dict, nucleotides_prob, Pck)

        nucleotides_prob['A'] = B[0]
        nucleotides_prob['C'] = B[1]
        nucleotides_prob['T'] = B[2]
        nucleotides_prob['G'] = B[3]

        loglik_curr = loglikelihood(gamma)
        if (abs(loglik_curr) - loglik_prev < 0.001):
            conv = True
        loglik_prev = loglik_curr
        itr += 1
    return Pck
```

4b. Test your function using sequences in file shortmotif.txt.

In [45]:
```
Pck = findmotif('shortmotif.txt', 5, 100)
np.savetxt("shortmotif_PWM.csv", Pck, delimiter="\t")
print(Pck)
```

```
[[ 0.33490318  0.25533093  0.20480038  0.16002505  0.24113299]
 [ 0.04166667  0.24724517  0.20941226  0.42815773  0.20676955]
 [ 0.20187415  0.41667375  0.24011027  0.28505357  0.21109404]
 [ 0.421556    0.08075015  0.34567709  0.12676365  0.34100342]]
```

In [46]:
```
max_indices = [list(Pck[:,i]).index(max(Pck[:,i])) for i in range
(5)]
motif = "".join([['A', 'C', 'T', 'G'][i] for i in max_indices])
print(motif)
```

```
GTGCG
```

4c. Test your function using sequences in file longmotif.txt.

In [47]:
```
Pck = findmotif('longmotif.txt', 8, 50)
np.savetxt("longmotif_PWM.csv", Pck, delimiter="\t")
print(Pck)
```

```
[[ 0.23960764  0.28939401  0.19443794  0.28393938  0.2819804   0.1
625083
   0.43414859  0.34312959]
 [ 0.34224468  0.20034018  0.33260079  0.27406113  0.22183682  0.1
1344038
   0.14031175  0.24296603]
 [ 0.35932415  0.25926189  0.19296801  0.26577793  0.1288782   0.5
3703908
   0.19124647  0.20724608]
 [ 0.05882353  0.25100392  0.27999326  0.17622156  0.36730457  0.1
8701225
   0.23429319  0.20665829]]
```

In [48]:
```
max_indices = [list(Pck[:,i]).index(max(Pck[:,i])) for i in range
(8)]
motif = "".join([['A', 'C', 'T', 'G'][i] for i in max_indices])
print(motif)
```

```
TACAGTAA
```