Lior Ziv 305742611

Parsimony

1. In the Parsimony model the best tree is the tree with minimum changes. At class we built a score function which is actually a loss function, defining matrix M(MSA matrix) with n rows(number of sequences) and L columns (L is the max of sequences lengths M in our case) .Denoting the A_i as the i'th column of M, hence M = $[A_1,...A_L]$ →Score(M) = $\sum_{i=1}^{L}$ Score(A_i) (we explained in class how a this matrix can be build).

   Now we can define the optimization problem as :

   - Parsimony(M,T) = $\sum_{k}$ $Parsimony(A_k, T)$
   - $Parsimony(A_k, T)$ = minA_k^{N+1}...A_k^{2N-1}(placement for the internal nodes)($\Sigma_{(i,j)\in T} 1\{A_i^j \neq A_i^i\}$)

   This Parsimony loss function uses the minimum necessary changes by choosing an assignment for the inner leaves that minimizes the total changes between adjacent nodes.

2. **Parsimony Algorithm**
   (a) Start from node r
   (b) Go over the tree - post order
   (c) If j is leaf $S_j[a]$= $\left\{ \begin{array}{cc} 0 & A_j = a \\ \\ \infty & else \end{array} \right\}$
   (d) Else we need to compute S_i, S_k the daughter nodes i,k of j :
      i. $S_j[a]$= $\min_{b,c}[S_k[b]+ S_l[c]+1\{a\neq b\}+ 1\{a\neq c\}]$
      -Adds 1 if there were changes between sequences
      -Also since k,l are independent for a given a we get that the above statement equals →$\min_c[S[c]+ 1\{a\neq c\}]$ +$\min_b[S[b]+ 1\{a\neq b\}]$
   (e) Finally for r,j∈ $Neighbor(r)$
      i. Minimal cost of tree = $\min_{a,b}[S_r[a] + S_j[b] + 1\{a\neq b\}]$

   RunTime - time to fill each cell is O($||\Sigma||$) , the amount of cells to fill is $||\Sigma||\cdot O(n)$, number of sequences n, width of alignment is M (no gaps) . We get a total of O($M\cdot n\cdot||\Sigma||^2$).

   Space Complexity - We get an $||\Sigma||\cdot O(n)$ matrix →$O(\ ||\Sigma||\cdot O(n))$

3.

- In order to modify the algorithm I will add edge weights for every transition according to S(a,b) so now instead of adding 1 to the penalty sum, I will add S(a,b)
- In order to retrieve the internal nodes with the minimal penalty I will save two pointers for each internal node, one for each daughter node, $l_k(a)$/ $r_k(a)$ - left/right pointer for the k'th node

(a) Start from node r

(b) Go over the tree - post order

(c) If j is leaf $S_j[a]= \begin{cases} 0 & A_j = a \\ \infty & else \end{cases}$

(d) Else we need to compute S_i, S_k the daughter nodes i,k of j :

i. $S_j[a]= \min_{b,c}[S_k[b]+ S_l[c]+ \mathbf{S(a,b)\{a\neq b\}}+ \mathbf{S(a,c)\{a\neq c\}}] \rightarrow \min_c[S[c]+ \mathbf{S(a,c)\{a\neq c\}}] + \min_b[S[b]+ \mathbf{S(a,b)\{a\neq b\}}]$

(e) **Set** $l_k(a)= \mathbf{argmin}_b(S_i(b)+ \mathbf{S(a,b))}$ **and** $r_k(a)= \mathbf{argmin}_b(s_j(b)+\mathbf{S(a,b))}$

(f) Finally we got to the root, take l_{2N-1}(a), $r_{2N-1}(a)$ and we can start to traceback the entire internal tree.

4. I will suggest a new function f = $-2^{L_{i,j}}\{a\neq b\}$, which obviously gives a larger value to smaller distances, and smaller to bigger distance.

   The function will be added instead of indicator function given in the 1.1($1\{a\neq b\}$).

5. **sources** -

- http://moodle2.cs.huji.ac.il/nu16/pluginfile.php/377947/mod_resource/content/1/Lecture%2017%20-%20David%20Ariel.pdf -David scribe
- http://moodle2.cs.huji.ac.il/nu15/pluginfile.php/224934/course/section/39705/msa.pdf - Scribe from 2015
- page 174- 175 course book.
- http://moodle2.cs.huji.ac.il/nu16/pluginfile.php/383497/mod_resource/content/1/Lecture%2015%20-%20Michal%20Bazir.pdf -Michal Scribe

Segmentation

1. First I will define the probability of each θ_i given it's size ($n_{i+1}-n_i$) , I will set $(\#G+C)_i$ = sum of appearance of G and C in the i'th segment. $P(\theta_i) = (\frac{\theta_i}{2})^{(\#G+C)_i} \cdot (\frac{1-\theta_i}{2})^{(n_{i+1}-n_i-(\#G+C)_i)}$. By this definition we get that the maximum likelihood is $L(\theta_i) = \prod_{i=0}^{k} P(\theta_i) = \prod_{i=0}^{k-1} (\frac{\theta_i}{2})^{(\#G+C)_i}$.

$(\frac{1-\theta_i}{2})^{(n_{i+1}-n_i-(\#G+C)_i)}$, from this we can deduce the sufficient statistics is the $(\#G+C)_i$, and the size of each segment $(n_{i+1}-n_i)$.

Now we need to derive and compare to zero in order to get the MLE.

First I will apply log in order to derive more easily, $\mathrm{LL}(\theta_i) = \log(\prod_{i=0}^{k-1}(\frac{\theta_i}{2})^{(\#G+C)_i}$.

$(\frac{1-\theta_i}{2})^{(n_{i+1}-n_i-(\#G+C)_i)}$) $\rightarrow \sum_{i=0}^{k-1}(\#G+C)_i \cdot log(\frac{\theta_i}{2}) + (n_{i+1}-n_i-(\#G+C)_i)log(\frac{1-\theta_i}{2})$

in order to find specific θ_i i will set all the rest to constants$\rightarrow$

$(\#G+C)_i \cdot \frac{1}{\theta_i} + (n_{i+1}-n_i-(\#G+C)_i)(-\frac{1}{1-\theta_i}) = 0 \rightarrow (\#G+C)_i \cdot (1-\theta_i)$ - $((n_{i+1}-n_i)-(\#G+C)_i)(\theta_i) = 0 \rightarrow (\#G+C)_i - \theta_i(n_{i+1}-n_i) = 0$ $\rightarrow \theta_i = \frac{(\#G+C)_i}{n_{i+1}-n_i} = \rightarrow \boldsymbol{\theta_i} = \frac{(\#G+C)_i}{n_{i+1}-n_i}$.

2. **Definition of the problem**

Given N data points D = {x_1,...x_N}, I will define s_j =(b_j,e_j) - segment ($1 \leq j \leq k$). In order to fit the sequence optimally into k segments I will divide it according to $\overline{s_j}$ which hold the percentage of GC in the segment. The quality of the fit is evaluated by minimizing the total fit error :

- $\overline{s}_j = \frac{(\#G+C)_i}{e_j-b_j} = \sum_{i=b_j}^{e_j} \frac{1_{(x_i=G/C)}}{e_j-b_j}$
- Total Error - $E_s = \Sigma(x_i - \overline{x}_i)^2$ s.t $\overline{x}_i = \overline{s}_j$ for ($b_j \leq j \leq e_j$).

How to fill the table - lets define the structure, we will take k' s.t $(1 \leq k' \leq k)$ the notation E_s[i,k´] represent the segmentation error over the data points {$x_1 \ldots, x_i$} using k´-1 segments, and let E[i,j] be the error in representing the points {$x_i, \ldots, x_j$} using just the mean of the data(one segment).

In order to solve the problem we will work according to the next pseudo code:

- Initiation - initialize an empty table T(all zeros) size - n×k
- k' = 1 fill in T[:,1] = E_s s.t $|s_i| \in \{1, 2, ...n\}$ (fill each cell j with as the segment E_s(1:j)
- update rule - k' >1 T[i,k']= $\min_{1 \leq j \leq i}(E_s[j-1,k'-1]))+E[i,end])$
  - meaning the minimal possible way to select k'-1 segments up the the i'th letter, plus adding a tail of error for the rest of the sequence (total of k' segments)
  - The best fit segmentation score will be at the most right bottom of the table.

Prove of correctness

- By induction :
- **Base :** For k' = 1 trivial since we need to have only one segment (obviously the best fit) the last cell will have the total segment E_s .
- **Inductive step:** Now we can assume that the statement is correct for k' ≤ k-1 segments and prove for k' = k

  If we look at separation into k sequences(last col) T[k,:] , lets assume we don't get to optimal solution for a specific T[k,j] ($1 \leq j \leq n$). By our rule of update we get that the sum of our variables is not minimal-
  - $\underset{1 \leq j \leq i}{min}(E_s[j-1,k'-1]))+E[i,end])$ ($1 \leq j \leq n$) defined as the minimal - we get a contradiction since if there is a better fit which gives a smaller err our update rule will choose that segmentation.
- (I know i'ts not the best solution but I run out of time)

3. **RunTime** - as we can see we have :
   - Table size $n \cdot k$ we fill
   - Each cell takes $n \cdot n$, for every k segments we have to take the min(n options) on and the sum for each option(O(n))
   - Therefore the total runtime is $O(n^3 \cdot k)$

4. The distribution $\underline{N}(\phi_i, s)$ has a probability function $\rightarrow P(x_i:\phi_i, s) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x_i-\phi_i)^2}{2s^2}}$

   Now we can define the likelihood function $\rightarrow \prod_{i=1}^{k}\prod_{j=1}^{n} P(\phi_i) = \prod_{i=1}^{k}\prod_{j=1}^{n} \frac{1}{s\sqrt{2\pi}} e^{\frac{-(x_j-\phi_i)^2}{2s^2}}$

   Next for getting the MLE, I will apply log on $L(\Phi)$, derive LL according to ϕ_i and compare it to zero$\rightarrow \sum_{i=1}^{k}\sum_{i=1}^{n}(log(\frac{1}{s\sqrt{2\pi}}) + \frac{-(x_i-\phi_i)^2}{2s^2}) \overrightarrow{derive}$

   $\sum_{j=1}^{n} \frac{2(x_j-\phi_i)}{2s^2} = 0$

   $\rightarrow \sum_{j=1}^{n} 4x_i s^2 - 4\phi_i s^2 \rightarrow \sum_{j=1}^{n} x_i s^2 = n\phi_i s^2 \rightarrow \phi_i = \frac{\sum_{j=1}^{n} x_i}{n}$

   Sufficient statistic - as we can see in order to get the MLE value we need n, $\sum_{j=1}^{n} x_i$

5. I will take the algorithm I suggested on 2.2 and change the err function, E_s.
   - Total Error - $E_s = \Sigma(x_i - \overline{x}_i)^2$ s.t $x_i \in \widetilde{D}$ and $\overline{x}_i = \overline{s}_j$ for ($b_j \leq j \leq e_j$).
   - $\overline{s}_j = \frac{(\#G+C)_i}{e_j - b_j} = \sum_{i=b_j}^{e_j} \frac{1_{(x_i=G/C)}}{e_j - b_j}$ (just a reminder)
   - The time complexity is as I calculated before at 2.2 $O(n^3 \cdot k)$

6. **sources**

- https://dornsife.usc.edu/assets/sites/516/docs/papers/msw_papers/msw-019.pdf
- http://homepages.spa.umn.edu/~willmert/science/ksegments/#mjx-eqn-eqnrecurse
- http://www.siam.org/meetings/sdm06/proceedings/029terzie.pdf

Transition Rate matrices

1. In order to solve this question I will use the Up-Down algorithm with a few updates in order to take under consideration the fact that we have L_j a distribution over different R matrices which define the rate of change for the nucleotide in the j position for all sequences in the MSE.

   (a) $Pr(a \overset{t_{i,j}}{\rightarrow} x_j)$ when j is a leaf stays the same since we already know the leaf base.

   (b) $\Sigma_{b\in\Sigma} Pr(a \overset{t_{i,j}}{\rightarrow} b) = [e^{\Delta R_{L_j}}]_{a,b} \cdot Pr(R = R_{L_j} | L_{ij}) P(L_{i,j})$
   Here there is a change since we have different $R\{R_1..R_k\}$ matrices , and a distribution $L_{i,j}$ for choosing one.

   Up - Down algorithm

   (a) Input tree T, root r

   (b) Initialize: Post_order = DFS_post, Pre_order = DFS_pre(r)

   (c) **Up**(inward) :

   (d) for i in Post_order do:

      i. for j in N(i) such that j precedes in post_order:do

         A. for u in Σ do :

$$U_{i,j}[a] = \begin{cases} Pr(a \overset{t_{i,j}}{\rightarrow} x_j) & j \quad is \quad a \quad leaf \\ \Sigma_{b\in\Sigma} Pr(a \overset{t_{i,j}}{\rightarrow} b) \prod_{k\in N(j), k\neq i} U_{jk}[b] & j \quad isnot \quad a \quad leaf \end{cases}$$

   (e) Down(outward):

   (f) for i in Pre_order do:

      i. for j in N(i) such that i precedes j in pre_order:do

         A. for a in Σ do

$$U_{j,i}[a] = \Sigma_{b\in\Sigma} Pr(b \overset{t_{i,j}}{\rightarrow} a) \prod_{k'\in N(i), k'\neq j} U_{ik'}[b]$$

   Now in order to get the assignment of the inner nodes i will use posterior calculation for each internal node

   - $\forall i \in$ [n+1,...m] $\Pr(x_L | x_i = a)$

Runtime

- first we set an ordered array (pre/post) O(n)
- Iterate on each node once O(n) and for every node we perform $O(|\Sigma|^2)$ calculations

  As a result the total runtime is O(n)+ $O(n|\Sigma|^2)$ = O(n)

Maximum Likelihood

- $P(k) = \prod_{j \in \text{for each position in seq}} pr(R = R_k|L_j) \cdot pr(L_j)$, for every k $\in [1...k]$
- Now in order to get the MLE for each matrix we can apply log,derive and compare to zero
  - $\prod_{j \in \text{for each position in seq}} pr(R = R_k|L_j) \cdot pr(L_j) = \log(\prod_{j=1}^{n} pr(R = R_k|L_j) \cdot pr(L_j)) = \sum_{j=1}^{n} log(pr(R = R_k|L_j)) + \log(pr(L_j))$ (not sure what to do from here)

2. Not sure how to do this.

3. This question is pretty hard, plus I don't have the previews MLE's results.

   I will define a matrix L size n×k that will contain for each L_j ($1 \leq j \leq n$) k probabilities for choosing matrix R_k. At the beginning I will initialize L with uniform distribution in each cell ($\frac{1}{k}$) With the defined L run the up-down algorithm on our data.

   E step - calculate the sufficient statistics according to the data we get from the up-down algorithm.

   M step - update the L matrix according to our sufficient statistics

   Continue with this process until we get a convergence in the tree LL(a LL which explains the tree nodes assignment), which means we optimized L values up to the point where it explains our tree the best.

4. No time it was hard enough! :)

5. **sources**

- http://www.tau.ac.il/~talp/publications/recombSemphy.pdf
- 'Scribe 10 from previews year.
- Scribe 17 from previews year.
- http://moodle2.cs.huji.ac.il/nu16/pluginfile.php/375468/mod_resource/content/1/Lecture%2016%20-%20Nitzan%20Bodenheimer.pdf (niztan scribe)

<u>הצהרת אמון על עמידה בתנאי מבחן הבית</u>

אני, החתום מטה, מצהיר ומתחייב שפתרתי את מבחן הבית ב"אלגוריתמים בביולוגיה חישובית" בכוחות עצמי, ללא כל עזרה או התייעצות עם אדם אחר, ובכפוף לכל הכללים המפורטים בטופס המבחן.

שם: _____ ת"ז: _____

חתימה: _____ תאריך: _____

-