Quiz Section Week 5
April 26, 2016

Review
Programming: Matrices, files, more on functions, organizing programs
Topics (not guaranteed to be comprehensive!)

• Alignments
  • Reasons to align sequences
  • Needleman-Wunsch algorithm
  • Smith-Waterman algorithm
  • Effects of parameter variation (including gap penalties)
  • Testing for statistical significance of an alignment

• Phylogenetic trees
  • Rooted and unrooted topologies
  • Defining the best tree with UPGMA and Neighbor Joining
  • Concept of parsimony
  • Fitch algorithm: quantifying how parsimonious a tree is, assigning internal states
  • Finding the most parsimonious tree: Hill climbing w/ Nearest-Neighbor interchanges
  • Bootstrapping to quantify confidence in tree partitions

• Clustering
  • Defining a clustering problem
  • Hierarchical clustering
    • Impact of using single/complete/average linkage
  • K-means: Objective and algorithm

• General computation and programming
  • What is an algorithm
  • What is a search heuristic
  • Conceptual definitions of variable and function
  • Algorithm complexity with O(n) notation
  • Data types and converting between them
  • Program flow and control with conditional statements and loops
Phylogenetic trees

**UPGMA/Neighbor Joining**
- Define the best tree: based on distance between leaves
- Find the best tree using: polynomial time algorithm to construct the best tree from a distance matrix

**Parsimony approach**
- Define the best tree: Minimum # of mutations required to traverse tree
- Find the best tree: by enumerating all trees (exhaustive search), or by heuristic approach like Nearest-Neighbor Interchange Hill-Climbing
Tree topologies

Are these the same tree?
Tree topologies

Are these the same tree?

How about these?
Counting tree topologies

For N leaves

# of unrooted topologies = 3*5*7*...*(2N-5)
# of branches = 2N-3

E.g. an unrooted tree with 6 nodes

How many different topologies?
Counting tree topologies

For N leaves

# of unrooted topologies = $3 \times 5 \times 7 \times \cdots \times (2N-5)$
# of branches = $2N-3$

E.g. an unrooted tree with 6 nodes

How many different topologies?
$3 \times 5 \times 7 = \boxed{105}$
Counting tree topologies

For N leaves

# of unrooted topologies = 3*5*7*...*(2N-5)
# of branches = 2N-3

E.g. an unrooted tree with 6 nodes

How many different topologies?
3*5*7 = 105

How many branches?
Counting tree topologies

For $N$ leaves

# of unrooted topologies = $3 \times 5 \times 7 \times \ldots \times (2N-5)$
# of branches = $2N-3$

E.g. an unrooted tree with 6 nodes

How many different topologies?
$3 \times 5 \times 7 = \textbf{105}$

How many branches?
$2N-3 = \textbf{9}$
Counting tree topologies

For N leaves

# of unrooted topologies = 3*5*7*...*(2N-5)
# of branches = 2N-3

E.g. an unrooted tree with 6 nodes

How many different topologies?
3*5*7 = 105

How many branches?
2N-3 = 9
The root could be placed on any branch

E.g. an unrooted tree with 6 nodes

How many different topologies?
3*5*7 = 105

How many branches?
2N-3 = 9
The root could be placed on any branch

E.g. an unrooted tree with 6 nodes

How many different topologies?
3*5*7 = 105

How many branches?
2N-3 = 9

Total options = 105*9
= 945
Nearest Neighbor Interchange trees

For each *internal branch* generate two variant trees that swap the relationships of the four outside branches.
Nearest Neighbor Interchange trees

For each internal branch generate two variant trees that swap the relationships of the four outside branches.
Nearest Neighbor Interchange trees

For each internal branch generate two variant trees that swap the relationships of the four outside branches.
Nearest Neighbor Interchange trees

For each internal branch generate two variant trees that swap the relationships of the four outside branches.
Nearest Neighbor interchange keeps you from stepping too far in hill-climbing

Clades A and B are very closely related

<table>
<thead>
<tr>
<th>Tree</th>
<th>Negative parsimony Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-12</td>
</tr>
<tr>
<td>2</td>
<td>-4</td>
</tr>
<tr>
<td>3</td>
<td>-5</td>
</tr>
<tr>
<td>4</td>
<td>-76</td>
</tr>
<tr>
<td>5</td>
<td>-52</td>
</tr>
<tr>
<td>6</td>
<td>-30</td>
</tr>
</tbody>
</table>

Random designation of neighbors

Tree position on surface

NNI designation
NN Practice: Draw both interchanges from swapping this branch
NN Practice: Draw both interchanges from swapping this branch
Fitch algorithm practice
Fitch algorithm practice: bottom-up
Fitch algorithm practice: top-down
Hierarchical clustering with complete linkage example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A,B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A,B,C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B,C</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
Programming note: 2D matrices in Python

• List of lists!
• Each row is a different list

```
matrix = [ [0, 1, 2, 4], [1, 0, 2, 5], ...
print matrix[0][1]
1
print matrix[0]
[0, 1, 2, 4]
```

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
Reminder: “Big O” notation for complexity
What is the time complexity in $O()$ to compute the sum of a list?

```python
# x is a list of length N
sum = 0
for v in x:
    sum = sum + v
print 'The sum is:', sum
```

Directly proportional to # of items in list!  $O(N)$
How about the time complexity in $O()$ to compute the sum of an $N \times N$ matrix?

```python
# x is a list of N lists
# each list has N elements
sum = 0
for row in x:  # Do this N times
    for v in row:  # N times again
        sum = sum + v
print 'The sum is:', sum
# The answer is $O(N^2)$
```
Given a list of 2D points, compute their center

```python
given a list of 2D points, compute their center

points = [ [1,2], [3,4], [5,6], [7,8] ]
# center point is ( mean_x, mean_y )
mean_x = 0.0
mean_y = 0.0
for i in range(0,len(points)):
    mean_x += points[i][0]
    mean_y += points[i][1]

center = ( mean_x/len(points), mean_y/len(points) )
print center
(4.0, 5.0)
```
Reading data from a file in Python

```python
fin = open('qs5.txt', 'r')  # 'r' stands for 'read'
all_lines = []
for line in fin:  # In a for loop, fin acts like a list of strings
    print line
    all_lines.append(line)
fin.close()  # Lets the computer know it can free up resources used to read the file
print all_lines
```
Alternative file-reading structure

```python
my_open_file = open(sys.argv[1])
s1 = my_open_file.readline().strip()
s2 = my_open_file.readline().strip()
```

Note: if in a different directory, have to supply file path, e.g.:
python myScript.py /Users/cecilia/genome373/dataFile.txt
Writing data to a file

```python
fout = open('output.txt', 'w')  # 'w' stands for 'write
fout.write('Hello! How are you?\nI'm fine.')  # \n' starts a new line
fout.close()
```
Useful function: Split a string into its constituent words

```python
s = 'Wherefore art thou Romeo?
words = s.split() # Returns a list of substrings
print words
['Wherefore', 'art', 'thou', 'Romeo?']

# split() can use any arbitrary string to split by
words = s.split('r')
print words
['Whe', 'efo', 'e a', 't thou Romeo?']
```