Quiz Section Week 6
May 10, 2016

Viterbi and Forward-Backward Algorithms
Random numbers and simulation
Needleman-Wunsch: determine the best “hidden” evolutionary relationship between two sequences

Observed sequence

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-4</td>
<td>-8</td>
<td>-12</td>
<td>-16</td>
<td>-20</td>
</tr>
<tr>
<td>C</td>
<td>-4</td>
<td>0</td>
<td>-5</td>
<td>-4</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>-8</td>
<td>4</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Alignment score for a position is a function of a previous alignment score and a “transition” score
- Find the path through the matrix that has the best score
Viterbi: determine the likeliest hidden state sequence for an observed sequence

<table>
<thead>
<tr>
<th>Observed sequence</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-rich</td>
<td>0.4</td>
<td>0.9*0.8 = ?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T-rich</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Likelihood for an “alignment” of hidden state to observed sequence is a function of likelihood of previous alignment and transition probability (and also emission probability)
- Find the path through this matrix that has the highest probability
Needleman-Wunsch algorithm and Viterbi Algorithm to fill the matrix of scores and find the best path

**DP in equation form**

- Align sequence \( x \) and \( y \).
- \( F \) is the DP matrix; \( s \) is the substitution matrix; \( d \) is the linear gap penalty.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>-8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
F(0,0) = 0
\]

\[
F(i,j) = \max \left\{ \begin{array}{ll}
F(i-1,j-1) + s(x_i, y_j) \\
F(i-1,j) + d \\
F(i,j-1) + d
\end{array} \right.
\]

- “Align” observed sequence to state sequence

\[
F(i,j) = \max \left\{ \begin{array}{ll}
F(1,j-1)a(\pi_1, \pi_i)e(x_j, \pi_i) \\
F(2,j-1)a(\pi_2, \pi_i)e(x_j, \pi_i)
\end{array} \right.
\]

etc.
Needleman-Wunsch algorithm and Viterbi Algorithm to fill the matrix of scores and find the best path

DP in equation form

- Align sequence $x$ and $y$.
- $F$ is the DP matrix; $s$ is the substitution matrix; $d$ is the linear gap penalty.

\[
F(0,0) = 0
\]
\[
F(i,j) = \max\left\{ F(i-1,j-1) + s(x_i, y_j), F(i-1,j) + d, F(i,j-1) + d \right\}
\]

• “Align” observed sequence to state sequence

\[
F(i,j) = \max\left\{ F(1,j-1)a(\pi_1, \pi_i)e(x_j, \pi_i), F(2,j-1)a(\pi_2, \pi_i)e(x_j, \pi_i) \right. \\
\text{etc.}
\]

\[
\begin{array}{|c|c|c|c|c|} \hline
 & A & A & T & T \hline
\text{A-rich} & 0.4 & 0.288 & & \hline
\text{T-rich} & 0.1 & & & \hline
\end{array}
\]

\[
\pi_i
\]

\[
x_j
\]
Needleman-Wunsch algorithm and Viterbi Algorithm to fill the matrix of scores and find the best path

DP in equation form

- Align sequence $x$ and $y$.
- $F$ is the DP matrix; $s$ is the substitution matrix; $d$ is the linear gap penalty.

\[
F(0,0) = 0 \\
F(i,j) = \max \begin{cases} 
F(i-1,j-1) + s(x_i, y_j) \\
F(i-1,j) + d \\
F(i,j-1) + d
\end{cases}
\]

F(1,j-1)a(\pi_1, \pi_i)e(x_j, \pi_i)
F(2,j-1)a(\pi_2, \pi_i)e(x_j, \pi_i)

etc.
Needleman-Wunsch algorithm and Viterbi Algorithm to fill the matrix of scores and find the best path

DP in equation form

- Align sequence \( x \) and \( y \).
- \( F \) is the DP matrix; \( s \) is the substitution matrix; \( d \) is the linear gap penalty.

\[
F(0,0) = 0 \\
F(i,j) = \begin{cases} 
F(i-1,j-1) + s(x_i, y_j) \\
F(i-1,j) + d \\
F(i,j-1) + d 
\end{cases}
\]

F(1,j-1)a(\( \pi_1, \pi_i \))e(x_j, \( \pi_i \))

F(2,j-1)a(\( \pi_2, \pi_i \))e(x_j, \( \pi_i \))

etc.

"Align" observed sequence to state sequence

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-rich</td>
<td>0.4</td>
<td>.288</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>.00001</td>
</tr>
<tr>
<td>T-rich</td>
<td>0.1</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>.0002</td>
</tr>
</tbody>
</table>
Forward-backward algorithm accounts for all paths instead of best path

- What’s the probability that the T at position 3 was emitted by the T-rich state?
- What’s the probability that any path goes through the T-rich state at the third position?
- Instead of considering the single best path for a position/state pair, it combines all paths that pass through that position/state pair

\[
P(\pi_i = k | x) = \frac{P(x, \pi_i = k)}{P(x)}
\]

\[
P(x, \pi_i = k) = \sum_{\pi_i = k} P(\pi | x)
\]
Random number generation in Python!

What are some situations where you’d want to generate random numbers?
Random number generation in Python!

What are some situations where you’d want to generate random numbers?

In-class examples?
Random number generation in Python!

What are some situations where you’d want to generate random numbers?

In-class examples?

• Generating random sequences to create null distribution for sequence alignment
• Randomly shuffling data/gene annotations for GSEA
random() returns a uniformly distributed random value from [0,1)

• How can you convert this into a random coin flip with heads or tails?

```python
import random
r = random.random()
print(r)
```

0.261256363123
random() returns a uniformly distributed random value from [0,1)

- How can you convert this into a random coin flip with heads or tails?
- Throw a dart, call heads if dart lands between 0 and 0.5, tails if between 0.5 and 1
random() returns a uniformly distributed random value from [0,1)

- How can you convert this into a random coin flip with heads or tails?
  - Throw a dart, call heads if dart lands between 0 and 0.5, tails if between 0.5 and 1
Exercise: write a function to simulate a coin flip using random()

import random
# return 'heads' or 'tails' with 50/50 odds
def coinflip():
Exercise: write a function to simulate a coin flip using random()

```python
import random
# return heads or tails
def coinflip():
    v = random()
    if v > 0.5:
        return 'Tails'
    else:
        return 'Heads'
```
random() returns a uniformly distributed random value from [0,1)

• How can you convert this into a die roll?
Exercise: write a function to simulate a die roll using random()

```python
import random
# return 1,2,3,4,5, or 6 with equal odds
def dieroll():
```
Randomly shuffling a sequence of letters

How would you generate a random permutation of this sequence?

ATCGTCCTTAAGGATTACCATTGGCCTAGA
Randomly shuffling a sequence of letters

How would you generate a random permutation of this sequence?

ATCGTCCTTAAGGATTACCATTTGGCCTAGA