Lecture 24: Randomized Algorithms

Chapter 12
Randomized Algorithms

- Randomized algorithms incorporate random, rather than deterministic, decisions
- Commonly used in situations where no exact and/or fast algorithm is known
- Main advantage is that no input can reliably produce worst-case results because the algorithm runs differently each time.
Select

- **Select**(L, k) finds the k\(^{th}\) smallest element in L
- **Select**(L, 1) find the smallest...
  - Well known O(n) algorithm
    
    $$\text{minv} = \text{HUGE}$$
    
    $$\text{for v in L:}$$
    
    $$\text{if (v < minv):}$$
    
    $$\text{minv} = v$$

- **Select**(L, len(L)/2) find the median...
  - How?
    - $$\text{median} = \text{sorted}(L)[\text{len}(L)/2] \rightarrow O(n \log n)$$
- Can we find medians, or 1\(^{st}\) quartiles in O(n)?
Select Recursion

- **Select(L, k)** finds the $k^{th}$ smallest element in $L$
  - Select an element $m$ from unsorted list $L$ and partition $L$ the array into two smaller lists:
    
    $L_{lo}$ - elements smaller than $m$
    and
    $L_{hi}$ - elements larger than $m$.

- If $\text{len}(L_{lo}) > k$ then
  Select($L_{lo}$, $k$)

- else if $k > \text{len}(L_{lo}) + 1$ then
  Select($L_{hi}$, $k - \text{len}(L_{lo}) - 1$)

- else $m$ is the $k^{th}$ smallest element
Example of Select(L, 5)

Given an array: \( L = \{ 6, 3, 2, 8, 4, 5, 1, 7, 0, 9 \} \)

**Step 1:** Choose the first element as \( m \)

\[ L = \{ 6, 3, 2, 8, 4, 5, 1, 7, 0, 9 \} \]
Example of Select (cont’d)

**Step 2:** Split the array into $L_{lo}$ and $L_{hi}$

$L_{lo} = \{3, 2, 4, 5, 1, 0\}$

$L_{hi} = \{8, 7, 9\}$

$L = \{6, 3, 2, 8, 4, 5, 1, 7, 0, 9\}$
Example of Select (cont’d)

**Step 3:** Recursively call Select on either $L_{lo}$ or $L_{hi}$ until $\text{len}(L_{lo}) = k$, then return $m$.

- $\text{len}(L_{lo}) > k = 5 \rightarrow \text{Select}([3, 2, 4, 5, 1, 0], 5)$
  - $m = 3$
  - $L_{lo} = \{2, 1, 0\}$, $L_{hi} = \{4, 5\}$

- $k = 5 > \text{len}(L_{lo}) + 1 \rightarrow \text{Select}([4, 5], 5 - 3 - 1)$
  - $m = 4$
  - $L_{lo} = \{\text{empty}\}$, $L_{hi} = \{5\}$

- $k = 1 = \text{len}(L_{lo}) + 1 \rightarrow \text{return } 4$
def select(L, k):
    value = L[0]
    Llo = [t for t in data if t < value]
    Lhi = [t for t in data if t > value]
    below = len(Llo) + 1
    if (k < len(Llo)):
        return select(Llo, k)
    elif (k > below):
        return select(Lhi, k - below)
    else:
        return value
Select Analysis with Good Splits

- Runtime depends on our selection of $m$:

  - A good selection will split $L$ evenly such that

    \[ |L_{lo}| = |L_{hi}| = |L|/2 \]

  - The recurrence relation is:

    \[ T(n) = T(n/2) \]

  - $n + n/2 + n/4 + n/8 + n/16 + \ldots = 2n \Rightarrow O(n)$

Same as search for minimum
Select Analysis with Bad Splits

However, a poor selection will split $L$ unevenly and in the worst case, all elements will be greater or less than $m$ so that one Sublist is full and the other is empty.

For a poor selection, the recurrence relation is

$$T(n) = T(n-1)$$

In this case, the runtime is $O(n^2)$.

Our dilemma:

$O(n)$ or $O(n^2)$,

depending on the list… or $O(n \log n)$ independent of it
Select Analysis (cont’d)

- Select seems risky compared to sort
- To improve Select, we need to choose $m$ to give good ‘splits’
- It can be proven that to achieve $O(n)$ running time, we don’t need a perfect splits, just reasonably good ones.
- In fact, if both subarrays are at least of size $n/4$, then running time will be $O(n)$.
- This implies that half of the choices of $m$ make good splitters.
A Randomized Approach

• To improve Select, *randomly* select $m$.
• Since half of the elements will be good splitters, if we choose $m$ at random we will get a 50% chance that $m$ will be a good choice.
• This approach will make sure that no matter what input is received, the expected running time is small.
def randomizedSelect(L, k):
    value = random.choice(L)
    Llo = [t for t in data if t < value]
    Lhi = [t for t in data if t > value]
    below = len(Llo) + 1
    if (k < len(Llo)):
        return randomizedSelect(Llo, k)
    elif (k > below):
        return randomizedSelect(Lhi, k-below)
    else:
        return value
RandomizedSelect Analysis

- Worst case runtime: $O(n^2)$
- **Expected runtime**: $O(n)$.
- Expected runtime is a good measure of the performance of randomized algorithms, often more informative than worst case runtimes.
- Worst case runtimes are rarely repeated
- RandomizedSelect always returns the correct answer, which offers a way to classify Randomized Algorithms.
Two Types of Randomized Algorithms

• **Las Vegas Algorithms** – always produce the correct solution (i.e. randomizedSelect)

• **Monte Carlo Algorithms** – do not always return the correct solution.

• Las Vegas Algorithms are always preferred, but they are often hard to come by.
The Motif Finding Problem

Motif Finding Problem: Given a list of \( t \) sequences each of length \( n \), find the “best” pattern of length \( l \) that appears in each of the \( t \) sequences.

\[
l = 8
\]

\[
t = 5
\]

\[
n = 69
\]

\[
cctgatagacgctatctggctatccaggtacTTaggtcctctgtgcgaatctatgcttttcaaccat
\]

\[
agtactgggtcatatttgatCcAtacgtacaccggaacctgaaacaaacgctcagaaccagaagtgc
\]

\[
aacgtTAGtgcaccccttttcttcggtctctgccaacgagggctgtatgtataagacgaaatatttt
\]

\[
agcctccgatgtaagtcagctgtaactattacctgcccaccctattacatcttacgtCcAtataca
\]

\[
ctgttatacaacgcgtcatggcgggtatgcgttatttggtcgtcgtacgctcgcgttacgtacgGc
\]
A New Motif Finding Approach

• **Motif Finding Problem:** Given a list of \( t \) sequences each of length \( n \), find the “best” pattern of length \( l \) that appears in each of the \( t \) sequences.

• **Previously:** we solved the Motif Finding Problem using a Branch and Bound or a Greedy technique.

• **Now:** randomly select possible locations and find a way to greedily change those locations until we have converged to the hidden motif.
Profiles Revisited

- Let $s = (s_1, ..., s_t)$ be the starting positions for $l$-mers in our $t$ sequences.
- The substrings corresponding to these starting positions will form:
  - $t \times l$ alignment matrix
  - $4 \times l$ profile matrix*

* Note that we now define the profile matrix in terms of frequency, not counts as in Lecture 5.

$$P(X|profile) = 0.6 \times 0.8 \times 0.8 \times 1.0 \times 0.6 \times 0.8 \times 0.6 \times 0.8 = 0.0885$$
Scoring Strings with a Profile

• Let l-mer \( a = a_1, a_2, a_3, \ldots a_l \)

• \( P(a \mid P) \) is defined as the probability that an \( l \)-mer \( a \) was created by the Profile \( P \).

• If \( a \) is very similar to the consensus string of \( P \) then \( P(a \mid P) \) will be high

• If \( a \) is very different, then \( P(a \mid P) \) will be low.

\[
Prob(a \mid P) = \prod_{i=1}^{l} p(a_i, i)
\]
Scoring Strings with a Profile (cont’d)

Given a profile: \( P = \)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1/2</td>
<td>0</td>
<td>1/8</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1/8</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>1/8</td>
<td>1/8</td>
<td>0</td>
<td>7/8</td>
</tr>
<tr>
<td>G</td>
<td>1/4</td>
<td>0</td>
<td>1/8</td>
<td>1/8</td>
</tr>
</tbody>
</table>

The probability of the consensus string: 
\( \text{Prob}(\text{aaacct}|P) = ??? \)
Given a profile: $P = \begin{bmatrix} 1/2 & 7/8 & 3/8 & 0 & 1/8 & 0 \\ 1/8 & 0 & 1/2 & 5/8 & 3/8 & 0 \\ 1/8 & 1/8 & 0 & 0 & 1/4 & 7/8 \\ 1/4 & 0 & 1/8 & 3/8 & 1/4 & 1/8 \end{bmatrix}$

The probability of the consensus string:

$Prob(\text{aaacct}|P) = \frac{1}{2} \times \frac{7}{8} \times \frac{3}{8} \times \frac{5}{8} \times \frac{3}{8} \times \frac{7}{8} = .033646$
Given a profile: \( P = \)

<table>
<thead>
<tr>
<th>A</th>
<th>1/2</th>
<th>7/8</th>
<th>3/8</th>
<th>0</th>
<th>1/8</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1/8</td>
<td>0</td>
<td>1/2</td>
<td>5/8</td>
<td>3/8</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>1/8</td>
<td>1/8</td>
<td>0</td>
<td>0</td>
<td>1/4</td>
<td>7/8</td>
</tr>
<tr>
<td>G</td>
<td>1/4</td>
<td>0</td>
<td>1/8</td>
<td>3/8</td>
<td>1/4</td>
<td>1/8</td>
</tr>
</tbody>
</table>

The probability of the consensus string:
\[
Prob(\text{aacct}|P) = 1/2 \times 7/8 \times 3/8 \times 5/8 \times 3/8 \times 7/8 = .033646
\]

Probability of a different string:
\[
Prob(\text{atacag}|P) = 1/2 \times 1/8 \times 3/8 \times 5/8 \times 1/8 \times 1/8 = .001602
\]
P-Most Probable $l$-mer

- Define the P-most probable $l$-mer from a sequence as an $l$-mer in that sequence which has the highest probability of being created from the profile $P$.

$$\begin{array}{c|cccccc}
A & 1/2 & 7/8 & 3/8 & 0 & 1/8 & 0 \\
C & 1/8 & 0 & 1/2 & 5/8 & 3/8 & 0 \\
T & 1/8 & 1/8 & 0 & 0 & 1/4 & 7/8 \\
G & 1/4 & 0 & 1/8 & 3/8 & 1/4 & 1/8 \\
\end{array}$$

Given a sequence = ctataaacctttacatc, find the P-most probable $l$-mer
P-Most Probable l-mer (cont’d)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1/2</td>
<td>7/8</td>
<td>3/8</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1/8</td>
<td>0</td>
<td>1/2</td>
<td>5/8</td>
</tr>
<tr>
<td>c</td>
<td>1/8</td>
<td>1/8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>1/4</td>
<td>0</td>
<td>1/8</td>
<td>3/8</td>
</tr>
<tr>
<td>e</td>
<td>1/4</td>
<td>0</td>
<td>1/8</td>
<td>1/8</td>
</tr>
</tbody>
</table>

Find the $\text{Prob}(a|\text{P})$ of every possible 6-mer:

- First try: \textbf{c t a t a a a c c t t a c a t c}
- Second try: \textbf{c t a t a a a c c t t a c a t c}
- Third try: \textbf{c t a t a a a c c t t a c a t c}

-Continue this process to evaluate every possible 6-mer
P-Most Probable l-mer (cont’d)

Compute $\text{prob}(a|P)$ for every possible 6-mer:

| String, Highlighted in Red | Calculations | $\text{prob}(a|P)$ |
|-----------------------------|--------------|---------------------|
| ctataaaccttacat             | $1/8 \times 1/8 \times 3/8 \times 0 \times 1/8 \times 0$ | 0                   |
| ctataaaccttacat             | $1/2 \times 7/8 \times 0 \times 0 \times 1/8 \times 0$ | 0                   |
| ctataaaccttacat             | $1/2 \times 1/8 \times 3/8 \times 0 \times 1/8 \times 0$ | 0                   |
| ctataaaccttacat             | $1/8 \times 7/8 \times 3/8 \times 0 \times 3/8 \times 0$ | 0                   |
| ctataaaccttacat             | $1/2 \times 7/8 \times 3/8 \times 5/8 \times 3/8 \times 7/8$ | .0336               |
| ctataaaccttacat             | $1/2 \times 7/8 \times 1/2 \times 5/8 \times 1/4 \times 7/8$ | .0299               |
| ctataaaccttacat             | $1/2 \times 0 \times 1/2 \times 0 \times 1/4 \times 0$ | 0                   |
| ctataaaccttacat             | $1/8 \times 0 \times 0 \times 0 \times 1/8 \times 0$ | 0                   |
| ctataaaccttacat             | $1/8 \times 1/8 \times 0 \times 0 \times 3/8 \times 0$ | 0                   |
| ctataaaccttacat             | $1/8 \times 1/8 \times 3/8 \times 5/8 \times 1/8 \times 7/8$ | .0004               |
P-Most Probable l-mer (cont’d)

P-Most Probable 6-mer in the sequence is aaacct:

| String, Highlighted in Red | Calculations                      | Prob(a | P) |
|----------------------------|-----------------------------------|-------|
| ctataaacctttacat          | $1/8 \times 1/8 \times 3/8 \times 0 \times 1/8 \times 0$ | 0     |
| ctataaacctttacat          | $1/2 \times 7/8 \times 0 \times 0 \times 1/8 \times 0$ | 0     |
| ctataaacctttacat          | $1/2 \times 1/8 \times 3/8 \times 0 \times 1/8 \times 0$ | 0     |
| ctataaacctttacat          | $1/8 \times 7/8 \times 3/8 \times 0 \times 3/8 \times 0$ | 0     |
| ctataaacctttacat          | $1/2 \times 7/8 \times 3/8 \times 5/8 \times 3/8 \times 7/8$ | .0336 |
| ctataaacctttacat          | $1/2 \times 7/8 \times 1/2 \times 5/8 \times 1/4 \times 7/8$ | .0299 |
| ctataaacctttacat          | $1/2 \times 0 \times 1/2 \times 0 \times 1/4 \times 0$ | 0     |
| ctataaacctttacat          | $1/8 \times 0 \times 0 \times 0 \times 1/8 \times 0$ | 0     |
| ctataaacctttacat          | $1/8 \times 1/8 \times 0 \times 0 \times 3/8 \times 0$ | 0     |
| ctataaacctttacat          | $1/8 \times 1/8 \times 3/8 \times 5/8 \times 1/8 \times 7/8$ | .0004 |
aaacct is the P-most probable 6-mer in:

ctataaacctttacatc

because $Prob(\text{aaacct}|P) = 0.0336$ is greater than the $Prob(a|P)$ of any other 6-mer in the sequence.
Dealing with Zeroes

• In our toy example $\text{prob}(a | P) = 0$ in many cases. In practice, there will be enough sequences so that the number of elements in the profile with a frequency of zero is small.

• To avoid many entries with $\text{prob}(a | P) = 0$, there exist techniques to equate zero to a very small number so that one zero does not make the entire probability of a string zero (we will not address these techniques here).
P-Most Probable $l$-mers in Many Sequences

- Find the $P$-most probable $l$-mer in each of the sequences.

$$
\begin{array}{c|cccccc}
\text{ } & A & C & T & G \\
\hline
A & \frac{1}{2} & \frac{7}{8} & \frac{3}{8} & 0 & \frac{1}{8} & 0 \\
C & \frac{1}{8} & 0 & \frac{1}{2} & \frac{5}{8} & \frac{3}{8} & 0 \\
T & \frac{1}{8} & \frac{1}{8} & 0 & 0 & \frac{1}{4} & \frac{7}{8} \\
G & \frac{1}{4} & 0 & \frac{1}{8} & \frac{3}{8} & \frac{1}{4} & \frac{1}{8} \\
\end{array}
$$
**P-Most Probable l-mers in Many Sequences (cont’d)**

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>g</td>
<td>t</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>t</td>
<td>a</td>
<td>g</td>
<td>c</td>
<td>g</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>t</td>
</tr>
<tr>
<td>4</td>
<td>g</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>t</td>
</tr>
<tr>
<td>5</td>
<td>a</td>
<td>t</td>
<td>a</td>
<td>g</td>
<td>c</td>
<td>t</td>
</tr>
<tr>
<td>6</td>
<td>g</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>t</td>
<td>g</td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>t</td>
<td>c</td>
<td>c</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>8</td>
<td>t</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

A | 5/8 | 5/8 | 4/8 | 0 | 0 | 0 |
---|---|---|---|---|---|---|
C | 0 | 0 | 4/8 | 6/8 | 4/8 | 0 |
---|---|---|---|---|---|---|
T | 1/8 | 3/8 | 0 | 0 | 3/8 | 6/8 |
---|---|---|---|---|---|---|
G | 2/8 | 0 | 0 | 2/8 | 1/8 | 2/8 |

**P-Most Probable l-mers form a new profile**

cataaacgtttacatc
ataagcgaattcgactg
cagcccacaacccct
cggtaacctttacatct

tgcattcaataagctta
tgctcctgtccactcactc
tctccaaatccttttaca
ggtctaccttttatcct
Comparing New and Old Profiles

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>c</th>
<th>g</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>g</td>
<td>t</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>t</td>
<td>a</td>
<td>g</td>
<td>c</td>
<td>g</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>t</td>
</tr>
<tr>
<td>4</td>
<td>g</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>t</td>
</tr>
<tr>
<td>5</td>
<td>a</td>
<td>t</td>
<td>a</td>
<td>g</td>
<td>c</td>
<td>t</td>
</tr>
<tr>
<td>6</td>
<td>g</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>t</td>
<td>g</td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>t</td>
<td>c</td>
<td>c</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>8</td>
<td>t</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5/8</td>
<td>0</td>
<td>1/8</td>
<td>2/8</td>
</tr>
<tr>
<td>2</td>
<td>5/8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>4/8</td>
<td>4/8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>6/8</td>
<td>4/8</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>4/8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>3/8</td>
<td>0</td>
<td>3/8</td>
<td>6/8</td>
</tr>
<tr>
<td>7</td>
<td>1/8</td>
<td>1/8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>2/8</td>
<td>0</td>
<td>2/8</td>
<td>1/8</td>
</tr>
<tr>
<td>A</td>
<td>1/2</td>
<td>7/8</td>
<td>3/8</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1/8</td>
<td>0</td>
<td>1/2</td>
<td>5/8</td>
</tr>
<tr>
<td>T</td>
<td>1/8</td>
<td>1/8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1/4</td>
<td>0</td>
<td>1/8</td>
<td>3/8</td>
</tr>
</tbody>
</table>

Red – frequency increased, Blue – frequency decreased
Greedy Profile Motif Search

Use P-Most probable l-mers to adjust start positions until we reach a “best” profile; this is the motif.

1) Select random starting positions.
3) Create a profile $\mathbf{P}$ from the substrings at these starting positions.
4) Find the $\mathbf{P}$-most probable l-mer $\mathbf{a}$ in each sequence and change the starting position to the starting position of $\mathbf{a}$.
5) Compute a new profile based on the new starting positions after each iteration and proceed until we cannot increase the score anymore.
GreedyProfileMotifSearch Algorithm

1. **GreedyProfileMotifSearch**(DNA, t, n, l)
2. Randomly select starting positions s=(s_1,…,s_t) from DNA
3. \( bestScore \leftarrow 0 \)
4. while \( \text{Score}(s, DNA) > bestScore \)
5. \( \text{form profile } P \text{ from } s \)
6. \( bestScore \leftarrow \text{Score}(s, DNA) \)
7. for \( i \leftarrow 1 \) to \( t \)
8. \( \text{Find a } P\text{-most probable } l\text{-mer } a \text{ from the } i^{th} \text{ sequence} \)
9. \( s_i \leftarrow \text{starting position of } a \)
10. return \( bestScore \)
GreedyProfileMotifSearch Analysis

- Since we choose starting positions randomly, there is little chance that our guess will be close to an optimal motif, meaning it will take a very long time to find the optimal motif.
- It is unlikely that the random starting positions will lead us to the correct solution at all.
- In practice, this algorithm is run many times with the hope that random starting positions will be close to the optimum solution simply by chance.
**Gibbs Sampling**

- GreedyProfileMotifSearch is probably not the best way to find motifs.
- However, we can improve the algorithm by introducing **Gibbs Sampling**, an iterative procedure that discards one $l$-mer after each iteration and replaces it with a new one.
- Gibbs Sampling proceeds more slowly and chooses new $l$-mers at random increasing the odds that it will converge to the correct solution.
How Gibbs Sampling Works

1) Randomly choose starting positions \( s = (s_1, ..., s_t) \) and form the set of \( l \)-mers associated with these starting positions.

2) Randomly choose one of the \( t \) sequences.

3) Create a profile \( P \) from the other \( t - 1 \) sequences.

4) For each position in the removed sequence, calculate the probability that the \( l \)-mer starting at that position was generated by \( P \).

5) Choose a new starting position for the removed sequence at random based on the probabilities calculated in step 4.

6) Repeat steps 2-5 until there is no improvement.
Gibbs Sampling: an Example

Input:

\[ t = 5 \text{ sequences, motif length } l = 8 \]

1. GTAAACAATATTTTATAGC
2. AAAATTTACCTCGCAAGG
3. CCGTACTGTCAAGCGTGG
4. TGAGTAAACGACGTCCCA
5. TACTTAACACCCTGTCACAA
**Gibbs Sampling: an Example**

1) Randomly choose starting positions, \( s = (s_1, s_2, s_3, s_4, s_5) \) in the 5 sequences:

<table>
<thead>
<tr>
<th></th>
<th>Sequence 1</th>
<th>Sequence 2</th>
<th>Sequence 3</th>
<th>Sequence 4</th>
<th>Sequence 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
<td>11</td>
<td>9</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>GTAAACAAATTTTATAGC</td>
<td>AAAATTACCTTAGAAGG</td>
<td>CCGTACTGTCAAGCGTG</td>
<td>TGAGTAACCAGCGTCCA</td>
<td>TACTTAACCCTGTCGA</td>
</tr>
</tbody>
</table>
Gibbs Sampling: an Example

2) Choose one of the sequences at random:

**Sequence 2:** AAAATTTTACCTTTAGAAGG

\[ s_1 = 7 \quad \text{GTAAACAATATTTTAGC} \]
\[ s_2 = 11 \quad \text{AAAATTTTACCCTTAGAAGG} \]
\[ s_3 = 9 \quad \text{CCGTACTGTCAGCGTGG} \]
\[ s_4 = 4 \quad \text{TGAATAAACGACGTCCCA} \]
\[ s_5 = 1 \quad \text{TACTTTAACACCCTGTCAA} \]
Gibbs Sampling: an Example

2) Choose one of the sequences at random:

**Sequence 2**: AAAATTTACCTTTAGAAGG

\[ s_1 = 7 \quad \text{GTAAAC} \text{AATATTTTAGC} \]
\[ s_3 = 9 \quad \text{CCGTACTGTCAAGCGTGG} \]
\[ s_4 = 4 \quad \text{TGA} \text{GTAACGACGTCCCA} \]
\[ s_5 = 1 \quad \text{TACTTTAACACCCTGTCAA} \]
3) Create profile $P$ from $l$-mers in remaining 4 sequences:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>G</td>
<td>C</td>
<td>G</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>G</td>
<td>T</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>C</td>
<td>G</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>T</td>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1/4</td>
<td>2/4</td>
<td>2/4</td>
<td>3/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>2/4</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1/4</td>
<td>1/4</td>
<td>0</td>
<td>0</td>
<td>2/4</td>
<td>0</td>
<td>1/4</td>
</tr>
<tr>
<td>T</td>
<td>2/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>2/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>G</td>
<td>1/4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/4</td>
<td>0</td>
<td>3/4</td>
<td>0</td>
</tr>
</tbody>
</table>

Consensus String: TAAATCAGA
4) Calculate the $\text{prob}(a | P)$ for every possible 8-mer in the removed sequence:

| String                  | $\text{prob}(a | P)$ |
|-------------------------|----------------------|
| AAAATTTACCTTAGAAGGG     | 0.00732              |
| AAAATTTACCTTAGAAGGG     | 0.000122             |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0.000183             |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |
| AAAATTTACCTTAGAAGGG     | 0                    |

Strings Highlighted in Red
5) Create a distribution of probabilities of \( l \)-mers \( \text{prob}(a | P) \), and randomly select a new starting position based on this distribution.

A) To create this distribution, divide each probability \( \text{prob}(a | P) \) by the lowest one:

Starting Position 1: \( \text{prob}(\text{AAAATTTA} | P) = 0.000732 / 0.000122 = 6 \)
Starting Position 2: \( \text{prob}(\text{AAATTTAC} | P) = 0.000122 / 0.000122 = 1 \)
Starting Position 8: \( \text{prob}(\text{ACCTTAGA} | P) = 0.000183 / 0.000122 = 1.5 \)

Ratio = 6 : 1 : 1.5
B) Define probabilities of starting positions according to the computed ratios

Probability (Selecting Starting Position 1): $\frac{6}{6+1+1.5} = 0.706$
Probability (Selecting Starting Position 2): $\frac{1}{6+1+1.5} = 0.118$
Probability (Selecting Starting Position 8): $\frac{1.5}{6+1+1.5} = 0.176$
C) Select a new starting position at random according to computed distribution:

\[
\begin{align*}
P(\text{selecting starting position 1}) & : 0.706 \\
P(\text{selecting starting position 2}) & : 0.118 \\
P(\text{selecting starting position 8}) & : 0.176
\end{align*}
\]
Gibbs Sampling: an Example

Assume we select the substring with the highest probability – then we are left with the following new substrings and starting positions.

\[ s_1 = 7 \quad \text{GTAAAC} \text{AATATTTATAGC} \]
\[ s_2 = 1 \quad \text{AAAATTTA} \text{CCTCGCAAGG} \]
\[ s_3 = 9 \quad \text{CCGTACTG} \text{TCAAGCGTGG} \]
\[ s_4 = 5 \quad \text{TGAGTA} \text{ATCGACGTCCCA} \]
\[ s_5 = 1 \quad \text{TACTTC} \text{ACACCCTGTCAA} \]
Gibbs Sampling: an Example

6) We iterate the procedure again with the above starting positions until we cannot improve the score any more.
Gibbs Sampler in Practice

- Gibbs sampling needs to be modified when applied to samples with biased distributions of nucleotides (relative entropy approach).
- Gibbs sampling often converges to locally optimal motifs rather than globally optimal motifs.
- Needs to be run with many randomly chosen seeds to achieve good results.
Another Randomized Approach

- **Random Projection Algorithm** is a different way to solve the Motif Finding Problem.
- **Guiding principle:** Instances of a motif agree at a subset of positions.
- However, it is unclear how to find these “non-mutated” positions.
- To bypass the effect of mutations within a motif, we randomly select a subset of positions in the pattern creating a **projection** of the pattern.
- Search for that projection in a hope that the selected positions are not affected by mutations in most instances of the motif.
Projections

- Choose $k$ positions in string of length $l$.
- Concatenate nucleotides at chosen $k$ positions to form $k$-tuple.
- This can be viewed as a projection of $l$-dimensional space onto $k$-dimensional subspace.

$l = 15$  

taxonomy

Projection $k = 7$

ATGGCATTCAGATTC $\rightarrow$ TGCTGAT

Projection $= (2, 4, 5, 7, 11, 12, 13)$
Random Projections Algorithm

- Select $k$ out of $l$ positions uniformly at random.

- For each $l$-tuple in input sequences, hash into bucket based on letters at $k$ selected positions.

- Recover motif from enriched buckets that contain many $l$-tuples.

Input sequence:

```
...T C A A T G C A C C T A T...
```

Bucket TGCT

```
TG
CA
CC
CT
```

```
TGCACCT
```
Some projections will fail to detect motifs but if we try many of them the probability that one of the buckets fills increases.

In the example below, the bucket **GC*AC is “bad” while the bucket  AT**G*C is “good”
Example

- $l = 7$ (motif size), $k = 4$ (projection size)
- Choose projection (1,2,5,7)

...TAGAC**ATCCGAC**TTGCCTTAC**TAC**...
Hashing and Buckets

- Hash function $h(x)$ obtained from $k$ positions of projection.
- Buckets are labeled by values of $h(x)$.
- *Enriched buckets*: contain more than $s$ $l$-tuples, for some parameter $s$. 

```
ATTC
CATC
GCTC
ATGC
```
Motif Refinement

• How do we recover the motif from the sequences in enriched buckets?
• $k$ nucleotides are exact matches, (hash key of bucket).
• Use information in other $l-k$ positions as starting point for local refinement scheme, e.g. Gibbs sampler.
Synergy between Random Projection and Gibbs Sampler

- Random Projection is a procedure for finding good starting points: every enriched bucket is a potential starting point.
- Feeding these starting points into existing algorithms (like Gibbs sampler) provides good local search in vicinity of every starting point.
- These algorithms work particularly well for “good” starting points.
Building Profiles from Buckets

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>0</th>
<th>.25</th>
<th>.50</th>
<th>0</th>
<th>.50</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>.25</td>
<td>.50</td>
<td>0</td>
<td>.50</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>.25</td>
<td>.25</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>.50</td>
<td>0</td>
<td>1</td>
<td>.25</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>.25</td>
<td>0</td>
<td>.25</td>
<td>0</td>
</tr>
</tbody>
</table>

Profile P

Gibbs sampler

Refined profile P*
Motif Refinement

• For each bucket $h$ containing more than $s$ sequences, form profile $P(h)$

• Use Gibbs sampler algorithm with starting point $P(h)$ to obtain refined profile $P^*$
Random Projection Algorithm: A Single Iteration

- Choose a random \( k \)-projection.
- Hash each \( l \)-mer \( x \) in input sequence into bucket labeled by \( h(x) \)
- From each enriched bucket (e.g., a bucket with more than \( s \) sequences), form profile \( P \) and perform Gibbs sampler motif refinement
- Candidate motif is best found by selecting the best motif among refinements of all enriched buckets.
Choosing Projection Size

- Projection size $k$
  - choose $k$ small enough so that several motif instances hash to the same bucket.
  - choose $k$ large enough to avoid contamination by spurious $l$-mers:
    \[4^k >> t \left(n - l + 1\right)\]
• **Planted bucket**: bucket with hash value $h(M)$, where $M$ is the motif.

• Choose $m =$ number of iterations, such that

$$\Pr(\text{planted bucket contains at least } s \text{ sequences in at least one of } m \text{ iterations}) = 0.95$$

• Probability is readily computable since iterations form a sequence of independent Bernoulli trials