Randomized Algorithms and Motif Finding
Outline

• Randomized QuickSort
• Randomized Algorithms
• Greedy Profile Motif Search
• Gibbs Sampler
• Random Projections
Randomized Algorithms

- Randomized algorithms make random rather than deterministic decisions.
- The main advantage is that no input can reliably produce worst-case results because the algorithm runs differently each time.
- These algorithms are commonly used in situations where no exact and fast algorithm is known.
Introduction to QuickSort

- **QuickSort** is a simple and efficient approach to sorting:

- Select an element $m$ from unsorted array $c$ and divide the array into two subarrays: $c_{small}$ - elements smaller than $m$ and $c_{large}$ - elements larger than $m$.

- Recursively sort the subarrays and combine them together in sorted array $c_{sorted}$.
Example of QuickSort

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

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

\[ c = \{ 6, 3, 2, 8, 4, 5, 1, 7, 0, 9 \} \]

Our Selection
Example of QuickSort (cont’d)

**Step 2:** Split the array into $c_{\text{small}}$ and $c_{\text{large}}$

$c_{\text{small}} = \{ 3, 2, 4, 5, 1, 0 \}$

$c_{\text{large}} = \{ 8, 7, 9 \}$

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

**Step 3:** Recursively do the same thing to $c_{small}$ and $c_{large}$ until each subarray has only one element or is empty.

$c_{small} = \{3, 2, 4, 5, 1, 0\}$

$m = 3$

$\{2, 1, 0\} < \{4, 5\}$

$m = 2$

$\{1, 0\} < \{empty\}$

$m = 1$

$\{0\} < \{empty\}$

$c_{large} = \{8, 7, 9\}$

$m = 8$

$\{7\} < \{9\}$

$m = 9$
**Example of QuickSort (cont’d)**

**Step 4:** Combine the two arrays with $m$ working back out of the recursion and as we build together the sorted array.

\[ C_{small} = \{ 0, 1, 2, 3, 4, 5 \} \]

\[ C_{large} = \{ 7, 8, 9 \} \]
Example of QuickSort (cont’d)

Finally we can assemble $c_{small}$ and $c_{large}$ with our original choice of $m$, creating the sorted array.

$c_{small} = \{0, 1, 2, 3, 4, 5\}$  \hspace{1cm}  $m = 6$  \hspace{1cm}  $c_{large} = \{7, 8, 9\}$

$c_{sorted} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
The QuickSort Algorithm

1. QuickSort(c)
2. if c consists of a single element
3. return c
4. $m \in c$
5. Determine the set of elements $c_{small}$ smaller than $m$
6. Determine the set of elements $c_{large}$ larger than $m$
7. QuickSort($c_{small}$)
8. QuickSort($c_{large}$)
9. Combine $c_{small}$, $m$, and $c_{large}$ into a single array, $c_{sorted}$
10. return $c_{sorted}$
QuickSort Analysis: Optimistic Outlook

- Runtime is based on our selection of $m$:
  
  - A good selection will split $c$ evenly such that $|c_{small}| = |c_{large}|$, then the runtime is $O(n \log n)$.
  
  - For a good selection, the recurrence relation is:
    \[ T(n) = 2T(n/2) + \text{const} \cdot n \]

  The time it takes to sort two smaller arrays of size $n/2$  
  Time it takes to split the array into 2 parts where $\text{const}$ is a positive constant
QuickSort Analysis: Pessimistic Outlook

However, a poor selection will split c unevenly and in the worst case, all elements will be greater or less than \( m \) so that one subarray is full and the other is empty. In this case, the runtime is \( O(n^2) \).

For a poor selection, the recurrence relation is:

\[
T(n) = T(n-1) + \text{const} \cdot n
\]

The time it takes to sort one array containing \( n-1 \) elements

Time it takes to split the array into 2 parts where \( \text{const} \) is a positive constant
QuickSort Analysis (cont’d)

• QuickSort seems like an ineffectient MergeSort
• To improve QuickSort, we need to choose $m$ to be a good ‘splitter.’
• It can be proven that to achieve $O(n \log n)$ running time, we don’t need a perfect split, just reasonably good one. In fact, if both subarrays are at least of size $n/4$, then running time will be $O(n \log n)$.
• This implies that half of the choices of $m$ make good splitters.
A Randomized Approach

- To improve QuickSort, *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.
The RandomizedQuickSort Algorithm

1. RandomizedQuickSort(c)
2. if c consists of a single element
   return c
4. Choose element $m$ uniformly at random from c
5. Determine the set of elements $c_{small}$ smaller than $m$
6. Determine the set of elements $c_{large}$ larger than $m$
7. RandomizedQuickSort($c_{small}$)
8. RandomizedQuickSort($c_{large}$)
9. Combine $c_{small}$, $m$, and $c_{large}$ into a single array, $c_{sorted}$
10. return $c_{sorted}$

*Lines Highlighted in red indicate the differences between QuickSort and RandomizedQuickSort*
RandomizedQuickSort Analysis

- Worst case runtime: $O(m^2)$
- *Expected runtime*: $O(m \log m)$.
- Expected runtime is a good measure of the performance of randomized algorithms, often more informative than worst case runtimes.
- RandomizedQuickSort will always return the correct answer, which offers a way to classify Randomized Algorithms.
Two Types of Randomized Algorithms

• **Las Vegas Algorithms** – always produce the correct solution (ie. RandomizedQuickSort)

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

*We make a special note that the profile matrix will be defined in terms of the frequency of letters, and not as the count of letters.
Scoring Strings with a Profile

- $\text{Prob}(a|\mathbf{P})$ is defined as the probability that an $l$-mer $a$ was created by the Profile $\mathbf{P}$.
- If $a$ is very similar to the consensus string of $\mathbf{P}$ then $\text{Prob}(a|\mathbf{P})$ will be high.
- If $a$ is very different, then $\text{Prob}(a|\mathbf{P})$ will be low.

$$\text{Prob}(a|\mathbf{P}) = \prod_{i=1}^{n} p_{a_i}, i$$
Given a profile: $P = \begin{bmatrix}
A & 1/2 & 7/8 & 3/8 & 0 & 1/8 & 0 \\
C & 1/8 & 0 & 1/2 & 5/8 & 3/8 & 0 \\
T & 1/8 & 1/8 & 0 & 0 & 1/4 & 7/8 \\
G & 1/4 & 0 & 1/8 & 3/8 & 1/4 & 1/8
\end{bmatrix}$

The probability of the consensus string:

$$\text{Prob(}\text{aaacct}|P) = \text{???}$$
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>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>

The probability of the consensus string:

\[
\text{Prob}(\text{aacct}|P) = \frac{1}{2} \times \frac{7}{8} \times \frac{3}{8} \times \frac{5}{8} \times \frac{3}{8} \times \frac{7}{8} = 0.033646
\]
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>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>1/4</td>
</tr>
</tbody>
</table>

The probability of the consensus string:

\[
\text{Prob}(\text{aaacct}|P) = \frac{1}{2} \times \frac{7}{8} \times \frac{3}{8} \times \frac{5}{8} \times \frac{3}{8} \times \frac{7}{8} = .033646
\]

Probability of a different string:

\[
\text{Prob}(\text{atacag}|P) = \frac{1}{2} \times \frac{1}{8} \times \frac{3}{8} \times \frac{5}{8} \times \frac{1}{8} \times \frac{1}{8} = .001602
\]
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.

\[
P = \begin{array}{ccccccc}
A & 1/2 & 7/8 & 3/8 & 0 & 1/8 & 0 \\
C & 1/8 & 0 & 1/2 & 5/8 & 3/8 & 0 \\
T & 1/8 & 1/8 & 0 & 0 & 1/4 & 7/8 \\
G & 1/4 & 0 & 1/8 & 3/8 & 1/4 & 1/8 \\
\end{array}
\]

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>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>

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

First try: $\text{ctataaaccctttacatc}$

Second try: $\text{ctataaaccctttacatc}$

Third try: $\text{ctataaaccctttacatc}$

- 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 x 1/8 x 3/8 x 0 x 1/8 x 0          | 0                      |
| ctataaaaccttacat         | 1/2 x 7/8 x 0 x 0 x 1/8 x 0            | 0                      |
| ctataaaccttacat          | 1/2 x 1/8 x 3/8 x 0 x 1/8 x 0          | 0                      |
| ctataaaccttacat          | 1/8 x 7/8 x 3/8 x 0 x 3/8 x 0          | 0                      |
| ctataaaccttacat          | 1/2 x 7/8 x 3/8 x 5/8 x 3/8 x 7/8      | .0336                  |
| ctataaaccttacat          | 1/2 x 7/8 x 1/2 x 5/8 x 1/4 x 7/8      | .0299                  |
| ctataaaccttacat          | 1/2 x 0 x 1/2 x 0 1/4 x 0              | 0                      |
| ctataaaccttacat          | 1/8 x 0 x 0 x 0 x 0 x 1/8 x 0          | 0                      |
| ctataaaccttacat          | 1/8 x 1/8 x 0 x 0 x 3/8 x 0            | 0                      |
| ctataaaccttacat          | 1/8 x 1/8 x 3/8 x 5/8 x 1/8 x 7/8      | .0004                  |
**P-Most Probable \( l \)-mer (cont’d)**

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

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

**aaacct** is the P-most probable 6-mer in:

```
ctataaacctttacatc
```

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

• In our toy example $prob(a|P)=0$ 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 $prob(a|P)=0$, 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.

<table>
<thead>
<tr>
<th></th>
<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>
<td></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>
<td></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>
<td></td>
</tr>
</tbody>
</table>

catakaacgttacatc
cagcccagaacctt
cgggtatacctttacatc
tgcattcaatagctta
tatcctttccactc
tccaaatcctttaca
ggctcatctttatcct
P-Most Probable $l$-mers in Many Sequences (cont’d)

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

<table>
<thead>
<tr>
<th></th>
<th>ctataaa</th>
<th>cgtttacatc</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>atagcg</td>
<td>attcgactg</td>
</tr>
<tr>
<td>3</td>
<td>cagccc</td>
<td>gaacccct</td>
</tr>
<tr>
<td>4</td>
<td>cgggtg</td>
<td>aaccttacatc</td>
</tr>
<tr>
<td>5</td>
<td>tgtcattc</td>
<td>aatagc</td>
</tr>
<tr>
<td>6</td>
<td>tgtc</td>
<td>atcc</td>
</tr>
<tr>
<td>7</td>
<td>tgtccgtt</td>
<td>tccactc</td>
</tr>
<tr>
<td>8</td>
<td>ggtctac</td>
<td>ctttatacct</td>
</tr>
</tbody>
</table>

P-Most Probable $l$-mers form a new profile
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>A</td>
<td>5/8</td>
<td>5/8</td>
<td>4/8</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>4/8</td>
<td>6/8</td>
</tr>
<tr>
<td>T</td>
<td>1/8</td>
<td>3/8</td>
<td>0</td>
<td>3/8</td>
</tr>
<tr>
<td>G</td>
<td>2/8</td>
<td>0</td>
<td>0</td>
<td>2/8</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.

3) Select random starting positions.
4) Create a profile $P$ from the substrings at these starting positions.
5) Find the $P$-most probable $l$-mer $a$ in each sequence and change the starting position to the starting position of $a$.
6) 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 \(s=(s_1, \ldots, s_t)\) from *DNA*
3. \(bestScore \beta 0\)
4. while \(Score(s, DNA) > bestScore\)
5. \(\quad \)Form profile \(P\) from \(s\)
6. \(\quad bestScore \beta Score(s, DNA)\)
7. \(\quad \)for \(i \beta 1\) to \(t\)
8. \(\quad \)Find a \(P\)-most probable \(l\)-mer \(a\) from the \(i^{th}\) sequence
9. \(\quad s_i \beta \) 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:**

\( t = 5 \) sequences, motif length \( l = 8 \)

1. GTAAACAATATTTATAGC
2. AAAATTTACCTCGCAAGG
3. CCGTACTGTCAAGCGTGG
4. TGAGTAAACGACGTCCCA
5. TACTTAACACCCTGTCAAA
Gibbs Sampling: an Example

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

\[
\begin{align*}
  s_1 &= 7 & \text{GTAAACAAATTTATAGC} \\
  s_2 &= 11 & \text{AAAAATTTACCTTAGAAGG} \\
  s_3 &= 9 & \text{CCGTACTGTCAAGCGTG} \\
  s_4 &= 4 & \text{TGAGTAAACGACGTC} \\
  s_5 &= 1 & \text{TACTTTAACACCCTGTCA} 
\end{align*}
\]
Gibbs Sampling: an Example

2) Choose one of the sequences at random:

**Sequence 2:** AAAATTTTACCTTTAGAAGG

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$=7</td>
<td>GTAAAC</td>
<td>AATTTTATAGC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_2$=11</td>
<td>AAAATTTTACCTTTAGAAGG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_3$=9</td>
<td>CCGTACTGTCGAACGTG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_4$=4</td>
<td>TGAGTAAACGACGTCG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_5$=1</td>
<td>TACTTTAACCCTGTCA</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Gibbs Sampling: an Example

2) Choose one of the sequences at random:

Sequence 2: AAAATTTTACCTTTAGAAGG

\[ s_1 = 7 \]
GTAAAC\textcolor{green}{AATATTTTATAGC}\\

\[ s_3 = 9 \]
CCGT\textcolor{green}{ACTG}TCAAGCGTGG\\

\[ s_4 = 4 \]
TGAG\textcolor{green}{TAAACG}ACGTCCC\textcolor{green}{A}\\

\[ s_5 = 1 \]
\textcolor{green}{TACTTAAACACCCTGTG}TCAA
Gibbs Sampling: an Example

3) Create profile $P$ from $l$-mers in remaining 4 sequences:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<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>T</td>
<td>A</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>A</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>
<td></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>
<td></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>
<td></td>
</tr>
<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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
<tr>
<td>Consensus String</td>
<td>T</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td>G</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>
Gibbs Sampling: an Example

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

| Strings Highlighted in Red                                      | $\text{prob}(a|P)$ |
|-----------------------------------------------------------------|--------------------|
| AAAATTTACCTTAGAAGGG                                              | 0.000732           |
| AAAATTTACCTTAGAAGGG                                              | 0.000122           |
| AAAATTTACCTTAGAAGGG                                              | 0.000183           |
| AAAATTTACCTTAGAAGGG                                              | 0.000183           |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
| AAAATTTACCTTAGAAGGG                                              | 0                  |
Gibbs Sampling: an Example

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 probability:

   - Starting Position 1: $\text{prob}(\text{AAAATTTA} | P) = 0.000732 / 0.000122 = 6$
   - Starting Position 2: $\text{prob}(\text{AAATTTC}A | P) = 0.000122 / 0.000122 = 1$
   - Starting Position 8: $\text{prob}(\text{ACCTTTAGA} | P) = 0.000183 / 0.000122 = 1.5$

   Ratio = 6 : 1 : 1.5
Turning Ratios into Probabilities

b) Define probabilities of starting positions according to 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): $\frac{1.5}{6+1+1.5} = 0.176$
Gibbs Sampling: an Example

c) Select the start position according to computed ratios:

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

\[
\begin{align*}
s_1 &= 7 & \text{GTAAAC} & \text{AATATTATAGC} \\
s_2 &= 1 & \text{AAAA} & \text{TATTACCGCAAGG} \\
s_3 &= 9 & \text{CCGT} & \text{TACTGTAAGCGTGG} \\
s_4 &= 5 & \text{TGA} & \text{GTAATCGACGTCCCA} \\
s_5 &= 1 & \text{TACTTC} & \text{ACACCCTGTCAA}
\end{align*}
\]
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 unequal 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**: Some instances of a motif agree on 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.

$l = 15 \quad \text{Projection} \quad k = 7$

ATGGCATTCA$\text{GATTC}$ $\quad \rightarrow \quad \text{TGCTGAT}$

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 bucket that contain many $l$-tuples.

Input sequence:

```
...TCAAATGCACCTAT...
```

Bucket TGCT
Random Projections Algorithm (cont’d)

• Some projections will fail to detect motifs but if we try many of them the probability that one of the buckets fills in is increasing.
• In the example below, the bucket **GC*AC is “bad” while the bucket AT**G*C is “good”

```
...ccATCCGACca...
...ttATGAGGCTc...
...ctATAAAGTCgc...
...tcATGTGACac...
```

ATGC CGTC
(7,2) motif
Example

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

\[\ldots TAGAC ATCCGAC TTTG C C TTA C TAC \ldots\]

Buckets

- ATGC
- GCTC
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 $s$.

ATGC  |  GCTC  |  CATC  |  ATTC
Motif Refinement

- How do we recover the motif from the sequences in the enriched buckets?
- $k$ nucleotides are from hash value of bucket.
- Use information in other $l-k$ positions as starting point for local refinement scheme, e.g. Gibbs sampler.

![Diagram showing local refinement algorithm from ATGC to ATGCAGC]

Local refinement algorithm

ATGC

Candidate motif

ATGCAGC
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

Profile P

Refined profile P*

Gibbs sampler
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 $\mathbf{P}$ 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:

$$4^k >> t (n - l + 1)$$
How Many Iterations?

- *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.
Expectation Maximization (EM)

- \( S = \{ x(1), \ldots x(t) \} \) : set of input sequences
- Given: A probabilistic motif model \( W(Q) \) depending on unknown parameters \( Q \), and a background probability distribution \( P \).
- Find value \( Q_{\text{max}} \) that maximizes likelihood ratio:
  \[
  \frac{\Pr(S \mid W(Q_{\text{max}}), P)}{\Pr(S \mid P)}
  \]
- EM is local optimization scheme. Requires starting value \( Q_0 \).
EM Motif Refinement (cont’d)

- For each input sequence, $x(i)$, return $l$-tuple $y(i)$ which maximizes likelihood ratio:

$$\frac{\Pr(y(i) | W(Q h^*))}{\Pr(y(i) | P)}$$

- $T = \{ y(1), y(2), \ldots, y(t) \}$
- $C(T) =$ consensus string