582670 Algorithms for Bioinformatics

Lecture 2: Exhaustive search and randomized algorithms for motif discovery

10.9.2015

These slides are based on previous years' slides by Alexandru Tomescu, Leena Salmela and Veli Mäkinen

These slides use material from http://bix.ucsd.edu/bioalgorithms/slides.php
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Greedy Motif Search

Randomized Algorithms
Biological Motivation

3’ gene X gene gene gene gene
Microarray experiment
5’
Biological Motivation (cont’d)

Microarray experiment

3’ gene X gene gene gene gene 5’

Microarray experiment
Gene Regulation

- Microarray experiments can be used to measure gene activity
- A gene can be knocked out to see what effect that has on gene activity
- An experiment can show that when one gene (gene X) is knocked out, 20 other genes stop being expressed.
- How can one gene have such a drastic effect?
Gene X encodes a regulatory protein, a.k.a. a transcription factor (TF)

The 20 unexpressed genes rely on gene X’s TF to induce transcription

A single TF may regulate multiple genes
Every gene contains a regulatory region (RR) typically stretching 100-1000 bp upstream of the transcriptional start site. Located within the RR are Transcription Factor Binding Sites (TFBS), also known as motifs, specific for a given transcription factor. TFs influence gene expression by binding to a specific location in the respective gene’s regulatory region - TFBS.
Transcription Factor Binding Sites

- A TFBS can be located anywhere within the regulatory region
- TFBS may vary slightly across different regulatory regions since non-essential bases could mutate
Motifs and Transcriptional Starting Sites

ATCCCG
gene

TTCC G G
gene

ATCCCG
gene

AT G CCG
gene

AT G CC C
gene
Motif Logo

- Motifs can mutate on non important bases
- The five motifs in five different genes have mutations in positions 3 and 5
- Representations called motif logos illustrate the conserved and variable regions of a motif

```
TGGGGGA
TGAGAGA
TGGGGGA
TGAGAGA
TGAGGGGA
TGAGAGA
TGAGGGGA
```
Identifying Motifs

- Genes are turned on or off by regulatory proteins
- These proteins bind to upstream regulatory regions of genes to either attract or block an RNA polymerase
- Regulatory protein (TF) binds to a short DNA sequence called a motif (TFBS)
- Genes regulated by the same TF share a motif
- Given the regulatory regions of co-expressed genes we want to identify the common motif
Identifying Motifs: Complications

- We do not know the motif sequence
- We do not know where it is located within the regulatory region of each gene
- Motifs can differ slightly from one gene to the next
- How to discern it from random “motifs”? 
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Greedy Motif Search

Randomized Algorithms
Random Sequences

atgaccgggatactgataccgtatatttgccctaggcggtacacacattagataaaacgtatgaagtacgttagactcgccgccc
acccctatttttttgacgacagatatttagtgacctggaaaaaaaaatttgagtacaaaaacttttccccgaatactgggcataaaggtac
tgagtatccctgggatgactttttgggaacactatatagtgtcctcctccgatttttggaatattgtaggtatcattccgcaggggtccg
gctgagaattggatgacctctgtaagttttccacgcaatcgccaacacccggaaccacaggcaagaccgataaaggag
tcccttttgcgtaatggtgcggggggagctggtacctagggaagccttaacgcaacttaatggccacacttaagttcatttaa
gtcaatcagtttttttgatgaggttaacctgagccccttgcggctgggtacgtgaggggcatagccccgcttgccgcacccaaaccttttcagttggtggtggcccttgtttagaggcccccgtactgatggaaactttcaattatgagagagcataatctatcgcgtgtgttca
aaccttggacctggggcgacctcataataagggagacccattcttcctgttaatcgtgatgtgacactatgtttgccccattggctaaaagcccaacttgacaaatggaagatagaatccttgcaatttcaacgtatgccgaaccgaaagggactggtgagcaacgacagattcttacgcccccttctgagttctcgcttagggtctcagttttttttctggtgatgagc
ctggtgagcagcagatattcttacgctaggcattagccttcggggatcataatagcacaagctttctggtactgatagc
Implanting Motif AAAAAAAAAGGGGGGGGGG

atgaccgggatactgat[AAAAAAAAGGGGGGGG]ggcgtacacattgataaagtgatagctacggtgacgc
accctattttttagcagatttagtgacctggaaccaatatgagtcataaacttttcgaata[AAAAAAAAGGGGGGGG]
tgagtatccctgggatgacctt[AAAAAAAAGGGGGGGG]tgcctctcccgatttttgaatatgtgtaggatcatccgcagcggtgcc
gctgagaattggatgt[AAAAAAAAGGGGGGGG]tccacgcaatcgcgaaccaacggacgaccaaggcaagaccgataaaggag
tcccttttgcgtaatgtgcgggaggctgtttacgtaggaagccctaacggacatta[AAAAAAAAGGGGGGGG]cttata
gtcaatcatgtttcttgatgatggatttt[AAAAAAAAGGGGGGGG]ggccgctttggccgcaccaaatcagtggtggcagcgc
cggttttgcctttttctagggcccctctgt[AAAAAAAAGGGGGGGG]caattatgagagagctaattctatcgcgtctgtcttaa
aacttgagtt[AAAAAAAAGGGGGGGG]ctggtgccacatacaagaggagtctttcttcttacgtaatgtctgtatgacactatgt
ttgcccccattgctaaagcccaacctgagcataagacacacattttgccattgctcttggaatgtagacactatt[AAAAAAAAGGGGGGGG]acccgaagggaactgtgagcaacgacagatttttagctgcattccggggaatcataaagcagaggtc[AAAAAAAAGGGGGGGG]
Where are the implanted motifs?

atgaccggggatactgataaaaaaaaaaagggggggggtacacattagataaaagtatgaagtacgttagactcgccgccccgccccccccctatatttttgagcagatttagtgacctggaaaaaaaaatttgagtacaaaaacttttccgaataaaaaaaaaaggggggggtgagatatctccggtgtctctcccccaatttttggaatatgttaggatcattcgccaggttccggtcgagaatgggatgaaaaaaggggggggtccacgcaatcgccgaccaacgcggaccCAAaggcaagaccgataaaggggtccctttgcaatgtctgtgaatggatttaaaaaaaaggggggggaccgcggacccaaattcagtgggcgagcgacggttttggcccttgttagaggcccccttttattttataatttttctggtaatgtgcgggaggcttgtactgagggaaggcctaaacgcgttaaaaatttttttttacgctttgctttgcccccttttttttttcttttccccccctttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttaaaa
Implanting Motif AAAAAAAAGGGGGGGG with four mutations
Where are the implanted motifs???
Why finding (15,4)-motifs is hard?

Aligning two first occurrences of the motif
The Implanted Motif Problem

Finding a motif in a sample of

- 20 random sequences (e.g. 600 nt long)
- Each sequence containing an implanted pattern of length 15 at random position
- Each pattern appearing with 4 random mismatches as (15,4)-motif
The Implanted Motif Problem

Common benchmark problem for algorithms
  ▶ Difficult but not impossible

Real data is noisy
  ▶ Some input sequence might not contain a motif
  ▶ Algorithm searching only motifs appearing in all sequences could fail
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Greedy Motif Search

Randomized Algorithms
The Motif Finding Problem

▶ Given a random sample of DNA sequences:

cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgttttccaaccatatagtacctgggtgtacatttgatacgtagtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtacgtgcaccctcttttctttcgttgctctggccaaacgagggctgtatgtataagacgaaaatttt
agcctccgatgtaagtcataagctgtaactattacctgccacccctattacatcttcgtacgtataaca
cgtttatacaacgcgtcatggcgggtatgcgttttgggtcgctgctacgctcgatcgtaacgtacgtc

▶ Find the pattern appearing in each of the individual sequences, the shared motif
The Motif Finding Problem

- Given a random sample of DNA sequences:

  ```
cctgatagacgctatctggctatccacgtacgttaggtcctctgtgcgaatctatgctgttccccaccata
tagactgggtgtacattttgataacgtacgtagcaacggcaacctgaaacacgcggccagaaccagaagtgc
aaacgtacgtgcacccctctttcttcgtgctctggccacggggtgatgtatagacgaaaatcccc
agcctccgatgtaagtcataagctgtatcactaactattacctgcccacccttttaactacactcgtacgtataaca
tgttatacaacgcgtcatggccgggtatgcgatggggttctcgtcgtacgtcgtgatcgtaacgtacgtc
  ```

- Find the pattern appearing in each of the individual sequences, the shared motif

- Additional information:
  - The hidden sequence is of length 8
  - The pattern is not exactly the same in each sequence because random point mutations may occur in the sequences
The Motif Finding Problem (cont’d)

The motifs revealed with no mutations:

cctgatagacgctatctggctatccagtacgtaggtcctcttgtgcgaatctatgcttcccaccat
agtactgggtacatattgatacgtaacaccggcaacctgaacaaacgctcagaaccagaagtgc
aacgtacgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaaatttt
agcctccgatgtaaagtcatagctgtaactattacctgcaccccattacatctttacgtacgtataaca
tgttatacaacgcgtcatggcggggtatgcgttgggtcgtacgcctcgatcgtaacgtacgtc
The Motif Finding Problem (cont’d)

The motifs revealed with 2 mutations:

cctgatagacgctatctggctatccacGgtacTtaggtcctctgtgcgaatctatgcgttttccaaccat
agtactgggtgtacatttgatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtTAgtgcaccctctttcttcgctgctctggcccaacgagggctgtatgataagacgaaatttt
agcctccgatgtagtagctagctgtaactattacctgcaccctttattacatctttacgtCcAtatatca
ctgtttatacaacgcgtcatggcgggtatgctttttgggtcgtcgctcgtacgctcgtatcgtaaCgtagGc
The Motif Finding Problem (cont’d)

The motifs revealed with 2 mutations:

```plaintext
cctgatagacgctatctggctatcc\[\text{AGgtacTt}\]aggtcctctgtgcgaatctatgcgttttccaaccat
agtactgggtgtacattttgat\[\text{CcAtacgt}\]acaccggcaacctgaaacaagcgcagaaccagaagtgca
aa\[\text{acgtTAg}t\]gcacccctctttcttcgctggctctggccaacggacctgtatataagacgaaat
agccctccgatgtaagtcataagctgttaacctattactccctgaccctcttcattacatcttt\[\text{acgtCcAt}\]atataca
tgttatacaacgcgtcatggcgggggtatgcgtttttggtgcgtacgcctcgatgtta\[\text{Ccgtacgc}\]
```

Can we still find the motifs now that we have 2 mutations?
### Motif Matrix

<table>
<thead>
<tr>
<th>Motifs</th>
<th>Count(Motifs)</th>
<th>Consensus(Motifs)</th>
<th>Score(Motifs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a G g t a c T t</td>
<td>A 3 0 1 0 3 1 1 1 0</td>
<td>A C G T A C G T</td>
<td>2+1+1+0+2+1+2+1=10</td>
</tr>
<tr>
<td>C c A t a c g t</td>
<td>C 2 4 0 0 1 4 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a c g t T A g t</td>
<td>G 0 1 4 0 0 0 3 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a c g t C c A t</td>
<td>T 0 0 0 5 1 0 1 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C c g t a c g G</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **t motifs (k-mers)**, one from each sequence
- Count symbols in each column
- Consensus formed by most frequent symbols
- Score is the number of mismatching symbols
The Motif Finding Problem: Formulation

- **Goal**: Given a set of DNA sequences, find a set of $k$-mers, one from each sequence, that minimizes the consensus score.
- **Input**: A collection of strings $DNA$, and an integer $k$
- **Output**: A collection $Motifs$ of $k$-mers, one from each string in $DNA$, minimizing $\text{Score}(Motifs)$
Parameters

\[ k = 8 \]

\[ t = 5 \]

\[ n = 69 \]

\[ \text{DNA} \]

\[ \text{cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcaatctatgcgttttccaaccat} \]
\[ \text{agtactggtgtacattttgtatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc} \]
\[ \text{aaacgTAgtgtcaccccttttcttctgtgcttggctctggccaacgagggctgatgtataagacgaaat} \]
\[ \text{agcctccgatgtaagtcatgctgtaactattacctgccaccctattacatctttacgtCcAtataca} \]
\[ \text{ctgttatataacgcgtcatgccccgggtatgcgttttggctgtcgtacgctcgtgatcgttaCcgtagGc} \]
BruteForceMotifSearch

- Compute the score for every possible combination of motifs
- Output the set of motifs with the smallest score
Running Time of BruteForceMotifSearch

- \((n - k + 1)\) different \(k\)-mers in each sequence
- \((n - k + 1)^t\) different combinations of motifs
- \(kt\) time to compute score for one set of motifs
- \(kt(n - k + 1)^t = O(ktn^t)\) time in total
- E.g. for \(t = 20\), \(n = 600\), \(k = 15\) we must perform approximately \(10^{58}\) computations — it would take billions of years
The Median String Problem

- Given a set of $t$ DNA sequences find a pattern that appears in all $t$ sequences with the minimum number of total mismatches.
- This pattern will be the shared motif.
The Hamming distance $d(v, w)$ is the number of mismatches between two $k$-mers $v$ and $w$.

For example:

$$d(\text{AAAAAAA}, \text{ACAAAA}C) = 2$$
Computing Score

Motifs

\[
\begin{align*}
\text{a G g t a c T t} & \quad 2 \\
\text{C c A t a c g t} & \quad 2 \\
\text{a c g t T A g t} & \quad 2 \\
\text{a c g t C c A t} & \quad 2 \\
\text{C c g t a c g G} & \quad 2 \\
\end{align*}
\]

Score(Motifs) \[2+1+1+0+2+1+2+1=10\]

Consensus(Motifs) \[A C G T A C G T\]

- Score is the number of mismatching symbols
- Can be computed column by column or row by row
- Row sums are Hamming distances
Computing Score

Define

- \( \text{Motifs} = \{\text{Motif}_1, \text{Motif}_2, \ldots, \text{Motif}_t\} \)
- \( d(\text{Pattern}, \text{Motifs}) = \sum_{i=1}^{t} d(\text{Pattern}, \text{Motif}_i) \)

Then

- \( \text{Score}(\text{Motifs}) = d(\text{Consensus}(\text{Motifs}), \text{Motifs}) \)
Best Match Distance

- Assume $|String| > |Pattern| = k$
- The best match distance $d(Pattern, String)$ is the smallest Hamming distance $d(Pattern, Motif)$ between $Pattern$ and any $k$-mer $Motif$ in $String$
- Example: $d(ACGTACGT, gcaaaAGGTACTTccaa) = 2$

Generalize for a set of strings
- $Dna = \{Dna_1, Dna_2, \ldots, Dna_t\}$
- $d(Pattern, Dna) = \sum_{i=1}^{t} d(Pattern, Dna_i)$
The Median String Problem

- **Goal**: Given a set of DNA sequences, find a median string
- **Input**: A collection of strings $DNA$ and an integer $k$
- **Output**: A $k$-mer $Pattern$ minimizing $d(Pattern, Dna)$ among all $k$-mers $Pattern$
Motif Finding Problem = Median String Problem

- **Motifs**: output of Motif Finding
- **Pattern**: output of Median String
- **Score**(Motifs) = d(Pattern, Dna)
Motif Finding Problem = Median String Problem

- **Motifs**: output of Motif Finding
- **Pattern**: output of Median String
- \[ \text{Score}(\text{Motifs}) = d(\text{Pattern}, \text{Dna}) \]

Why?

- If \[ \text{Score}(\text{Motifs}) < d(\text{Pattern}, \text{Dna}) \], we could choose \[ \text{Consensus}(\text{Motifs}) \] as a better **Pattern**
- If \[ \text{Score}(\text{Motifs}) > d(\text{Pattern}, \text{Dna}) \], we could choose the best match occurrences of **Pattern** as better **Motifs**
Median String Algorithm

MedianString($DNA, k$)

1: $BestPattern \leftarrow AAA\ldots A$
2: for each k-mer $Pattern$ from AAA\ldots A to TTT\ldots T do
3:    if $d(Pattern, DNA) < d(BestPattern, DNA)$ then
4:       $BestPattern \leftarrow Pattern$
5: return $BestPattern$
Running Time of MedianString

- $4^k$ different $k$-mers
- $O(k \cdot n)$ time to compute the best match distance to one string
- $O(knt4^k)$ time in total
- E.g. for $t = 20$, $n = 600$, $k = 15$ this is about about $10^{13}$
  — still a lot but much less than $10^{58}$
Running Time of MedianString

- $4^k$ different $k$-mers
- $O(k \cdot n)$ time to compute the best match distance to one string
- $O(knt4^k)$ time in total
- E.g. for $t = 20$, $n = 600$, $k = 15$ this is about $10^{13}$
  — still a lot but much less than $10^{58}$

- Reformulating a problem can help!
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Greedy Motif Search

Randomized Algorithms
Search Space

- BruteForceMotifSearch and MedianString algorithms have exponential running time
- This is because the search space, the set of possible solutions, is exponential
  - $n^t$ different ways to choose Motifs
  - $4^k$ different ways to choose Pattern
Exploring Only Part of Search Space

Branch and bound algorithms (covered in study groups)

- Avoid regions that cannot improve solution
- Still exponential in the worst case

Greedy algorithms

- Search the most promising directions
- No guarantee of finding an optimal solution

Randomized algorithms

- Add randomness to greedy search
- Avoids getting stuck in a dead end
Profile Matrix

Motifs

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

Count(Motifs)

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

Profile(Motifs)

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

Consensus(Motifs)  A C G T A C G T

Profile represents the probability of each nucleotide in each position

More detailed summary of the set of motifs than consensus
**k-Mer Probabilities**

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

The probability of a *k*-mer given a profile

\[
\text{Pr}(\text{AGGTACTT} \mid \text{Profile}) = 0.6 \cdot 0.2 \cdot 0.8 \cdot 1 \cdot 0.6 \cdot 0.8 \cdot 0.2 \cdot 0.8 = 0.0073728
\]

- Measure how well the *k*-mer matches the motif
- Does 0.0073728 imply a good match?
Profile-Most Probable $k$-mer

- The $k$-mer with the highest probability in a string
- Considered the best matching motif
- Example: The *Profile*-most probable 8-mer in `gcaaaAGGTACTTccaa` is `AGGTACTT`
  - $\Pr(\text{AGGTACTT} \mid \text{Profile}) = 0.0073728$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th></th>
<th></th>
<th>C</th>
<th></th>
<th>G</th>
<th></th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.6</td>
<td>0</td>
<td>.2</td>
<td>.6</td>
<td>.2</td>
<td>.2</td>
<td>.2</td>
<td>0</td>
</tr>
<tr>
<td><em>Profile</em></td>
<td>.4</td>
<td>.8</td>
<td>0</td>
<td>0</td>
<td>.2</td>
<td>.8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>.2</td>
<td>.8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.6</td>
<td>.2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>.1</td>
<td>.2</td>
<td>0</td>
<td>.2</td>
<td>.8</td>
<td></td>
</tr>
</tbody>
</table>
Problem: Zero Probabilities

\[
\begin{array}{cccccccc}
A & .6 & 0 & .2 & 0 & .6 & .2 & .2 & 0 \\
C & .4 & .8 & 0 & 0 & .2 & .8 & 0 & 0 \\
G & 0 & .2 & .8 & 0 & 0 & 0 & .6 & .2 \\
T & 0 & 0 & 0 & 1 & .2 & 0 & .2 & .8 \\
\end{array}
\]

Profile

Consensus

\[ \Pr(\text{TCGTACGT} \mid \text{Profile}) = 0 \cdot .8 \cdot .8 \cdot 1 \cdot .6 \cdot .8 \cdot .6 \cdot .8 = 0 \]

- Only one mismatch compared to consensus
- Should this probability really be 0?
### Pseudocounts

- Add one to all counts
- Avoids zero counts

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

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

47 / 64
Laplace’s Rule of Succession

- Use pseudocounts instead of counts to compute probabilities
- As if we had seen one occurrence of each symbol before the main data

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PseudoCount</td>
<td>C</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</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>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4/6</td>
<td>1/6</td>
<td>2/6</td>
<td>1/6</td>
<td>4/6</td>
<td>2/6</td>
<td>2/6</td>
<td>1/6</td>
<td></td>
</tr>
<tr>
<td>Profile</td>
<td>C</td>
<td>3/6</td>
<td>5/6</td>
<td>1/6</td>
<td>1/6</td>
<td>2/6</td>
<td>5/6</td>
<td>1/6</td>
<td>1/6</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>1/6</td>
<td>2/6</td>
<td>5/6</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
<td>4/6</td>
<td>2/6</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
<td>6/6</td>
<td>2/6</td>
<td>1/6</td>
<td>2/6</td>
<td>5/6</td>
</tr>
</tbody>
</table>
Greedy Motif Search

Solve Motif Finding problem
- Choose the profile-most probable $k$-mer in each string as the motif
  - *Greedy* choice
- Compute the profile from previously chosen motifs
- In first string, try all $k$-mers
Greedy Motif Search

GreedyMotifSearch($DNA, k, t$)

1: $BestMotifs \leftarrow$ the first $k$-mer of each string in $DNA$
2: for each $k$-mer $Motif$ in the first string in $DNA$ do
3:    $Motif_1 \leftarrow Motif$
4:    for $i \leftarrow 2$ to $t$ do
5:        form Profile from $Motif_1, \ldots, Motif_{i-1}$
6:        $Motif_i \leftarrow$ Profile-most probable $k$-mer in the $i$-th string in $DNA$
7:    $Motifs \leftarrow Motif_1, \ldots, Motif_t$
8:    if Score($Motif$) < Score($BestMotif$) then
9:        $BestMotifs \leftarrow Motifs$
10: return $BestMotifs$
Performance of GreedyMotifSearch

- Running time $O(n \cdot t \cdot k \cdot (n + t))$
  - polynomial not exponential
- May not find the best motifs
  - Early choices may lead to a wrong direction
Outline

Biological motivation

Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

Greedy Motif Search

Randomized Algorithms
Randomized Algorithms

- Make random choices during computation
- Use random number generator to “toss coins” or to “roll dice”

Why Randomness Helps?
- If a greedy algorithm fails for some input, it will always fail for that input
- If a randomized algorithm fails, it is unlikely to fail again in the same way
- We can run it many times and choose the best output
Monte Carlo and Las Vegas Algorithms

Monte Carlo algorithm
▶ May return an incorrect or inoptimal result
▶ Returns a correct answer or a good approximation with high probability (if repeated sufficiently many times)

Las Vegas algorithm
▶ Always returns a correct/optimal result
▶ Very long runtime is possible but very unlikely
Turning Monte Carlo into Las Vegas

1. Run the Monte Carlo algorithm
2. If the result is good, stop. Otherwise return to Step 1.
Turning Monte Carlo into Las Vegas

1. Run the Monte Carlo algorithm
2. If the result is good, stop. Otherwise return to Step 1.

▶ Requires that a correct or optimal result can be easily recognized
▶ This is not the case with the Motif Finding problem
    ▶ The following algorithms are Monte Carlo algorithms
Randomized Motif Search

Improving a set of motifs

- Starting with a set of motifs (one from each sequence)
  1. Compute a profile from the motifs
  2. Find the profile-most probable motifs in each sequence
- The result is a potentially better set of motifs
- Repeat this as long as the set of motifs keeps improving

Randomization

- Start with a random set of motifs
Randomized Motif Search

\texttt{RandomizedMotifSearch}(DNA, k, t)

1: randomly select \( k \)-mers \( \text{Motifs} = (\text{Motif}_1, \ldots, \text{Motif}_t) \), one from each string in \( DNA \)
2: \( \text{BestMotifs} \leftarrow \text{Motifs} \)
3: \textbf{while} forever \textbf{do}
4: \hspace{1em} \texttt{Profile} \leftarrow \texttt{Profile}(\text{Motifs})
5: \hspace{1em} \textbf{for} \hspace{1em} i \leftarrow 1 \textbf{ to } t \hspace{1em} \textbf{do}
6: \hspace{2em} \texttt{Motif}_i \leftarrow \texttt{Profile}-most probable \( k \)-mer in the \( i \)-th string in \( DNA \)
7: \hspace{1em} \text{Motifs} \leftarrow \text{Motif}_1, \ldots, \text{Motif}_t
8: \hspace{1em} \textbf{if} \hspace{1em} \text{Score}(\text{Motifs}) < \text{Score}(\text{BestMotifs}) \hspace{1em} \textbf{then}
9: \hspace{2em} \text{BestMotifs} \leftarrow \text{Motifs}
10: \hspace{1em} \textbf{else}
11: \hspace{2em} \textbf{return} \text{BestMotifs}
Why Randomized Motif Search Works?

- If \textit{Motifs} is a random set, the expectation is that \text{Profile}(\textit{Motifs}) has about the same probability 0.25 for each symbol in each column.
- If \textit{Motifs} contains some of the true motifs, it is not random and \text{Profile}(\textit{Motifs}) reflects this.
- Then \text{Profile}(\textit{Motifs}) is more likely to match the other true motifs.

- Thus we might need just a few of the true motifs in the initial set.
- This will happen eventually if repeated many times (may require thousands of repeats).
Gibbs Sampler

- Gibbs Sampler is a more refined randomized algorithm
- Compared to Randomized Motif Search Gibbs Sampler is
  - More cautious
  - More randomized
Gibbs Sampler Is More Cautious

- Randomized Motif Search might get some true motifs right but throw them all away in the next round
- Gibbs Sampler changes just one motif in each round
Gibbs Sampler Is More Randomized

- Randomized Motif Search uses randomness only in the beginning
- Gibbs Sampler uses randomness in every round
  - Choose a random motif to discard
  - Replace it with a random motif (from the same sequence)
  - The second random choice is biased: a profile-randomly generated $k$-mer
Profile-Randomly Generated $k$-Mer

- Given a *Profile* and a *String*
  1. Compute probabilities of all $k$-mers in *String*
  2. Choose one of the $k$-mers randomly but biased by the probabilities

- The probabilities with respect to *Profile* do not usually sum up to 1 and have to be normalized: Replace $p_1, \ldots, p_n$ with $p_1/C, \ldots p_1/C$, where $C = \sum_{i=1}^{n} p_i$

- Example
  - $p_1 = 0.1$, $p_2 = 0.2$, $p_3 = 0.3$
  - $C = 0.1 + 0.2 + 0.3 = 0.6$
  - $p_1/C = 1/6$, $p_2/C = 1/3$, $p_3/C = 1/2$
  - $p_1/C + p_2/C + p_3/C = 1/6 + 1/3 + 1/2 = 1$
Gibbs Sampler

GibbsSampler(DNA, k, t, N)
1: randomly select k-mers Motifs = (Motif_1, ..., Motif_t) in each string in DNA
2: BestMotifs ← Motifs
3: for j ← 1 to N do
4: i ← Random(t)
5: Profile ← profile matrix constructed from all strings in Motifs except for Motif_i
6: Motif_i ← profile-randomly generated k-mer in the i-th sequence in DNA
7: if Score(Motifs) < Score(BestMotifs) then
8: BestMotifs ← Motifs
9: return BestMotifs
Gibbs Sampler

- Because of randomness in every round, Gibbs Sampler can keep on running without getting stuck to single solution.
- However, it may end up exploring the same small set of solutions repeatedly: It gets stuck in a local optimum.
- This can be corrected by restarting from a new random set of motifs every now and then.