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

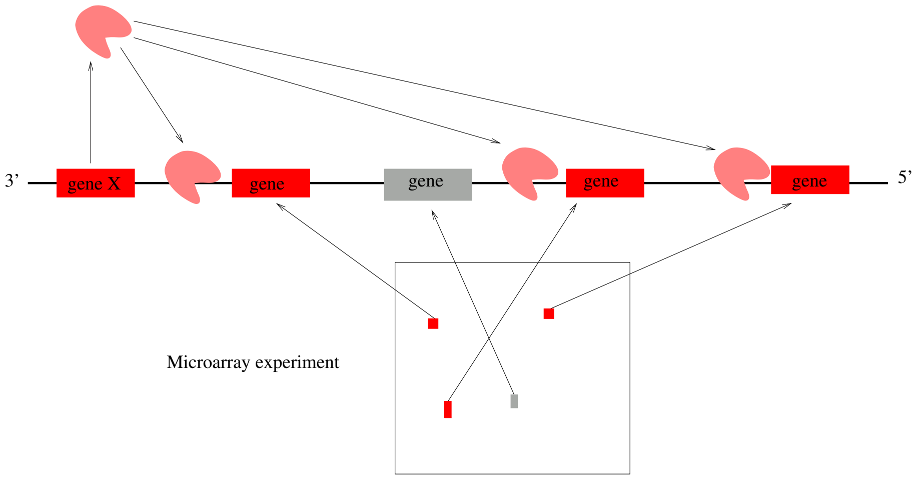
Implanted motifs - an introduction

Motif Finding Problem and Median String Problem

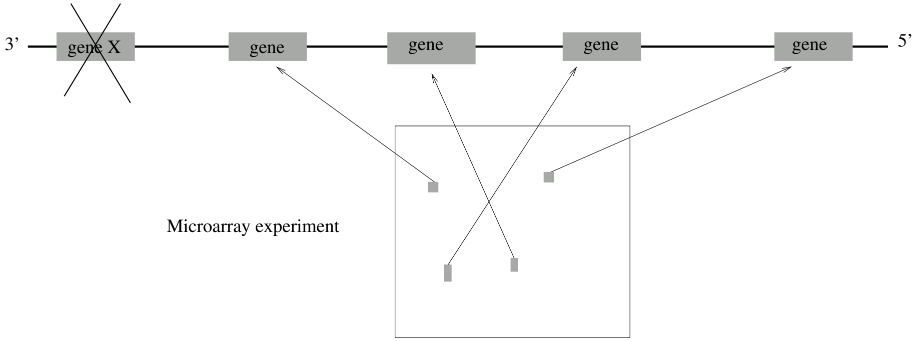
Greedy Motif Search

Randomized Algorithms

Biological Motivation



Biological Motivation (cont'd)



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?

Regulatory Proteins

- 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

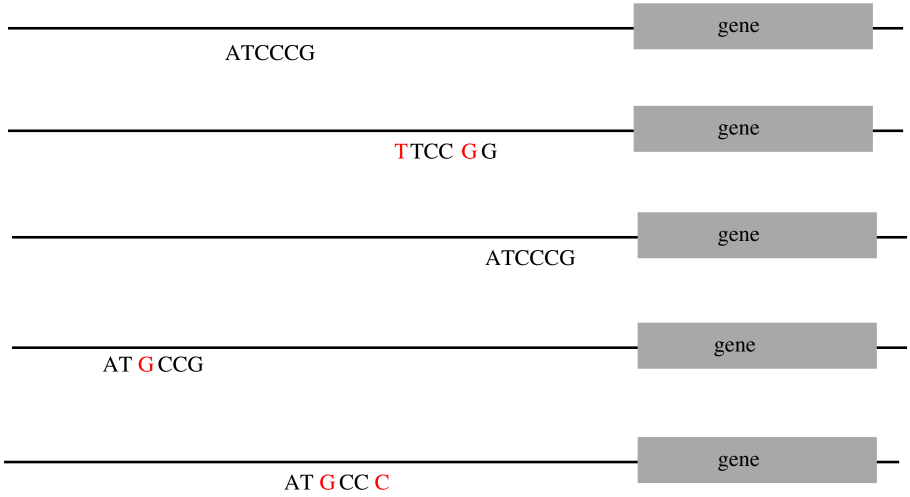
Regulatory Regions

- 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

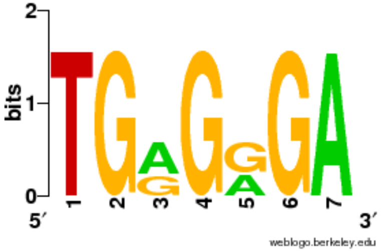
Motifs and Transcriptional Starting Sites



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
TGAGGGA
```



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

```
atgaccgggatactgataccgtatttggcctaggcgtacacattagataaacgtatgaagtacgttagactcggcgccgcc
acccctattttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaatactgggcataaggtac
tgagtatccctgggatgacttttgggaacactatagtgctctcccgatttttgaatatgtaggatcattcgccagggtccg
gctgagaattggatgaccttgtaagtgttttccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggag
tcccttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatggcccacttagtccacttata
gtcaatcatgttcttgtgaatggatttttaactgagggcatagaccgcttggcgcacccaaattcagtgtgggcgagcgca
cggttttggcccttgttagaggcccccgtactgatggaaactttcaattatgagagagctaatctatcgcgtgcgtgttca
aacttgagttggtttcgaaaatgctctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatttcaacgtatgccgaaccgaaagggaa
ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttctgggtactgatagc
```

Implanting Motif AAAAAAAAGGGGGGG

```
atgaccgggatactgatAAAAAAAAGGGGGGGggcgtacacattagataaacgtatgaagtacgttagactcggcgccgcc
acccctattttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaataAAAAAAAAGGGGGGG
tgagtatccctgggatgacttAAAAAAAAGGGGGGGtgctctcccgatttttgaatatgtaggatcattcgccagggtccg
gctgagaattggatgAAAAAAAAGGGGGGGtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggag
tcccttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAAAAAAAAGGGGGGGcttata
gtcaatcatgttcttgtgaatggatttAAAAAAAAGGGGGGGgaccgcttggcgcacccaaattcagtgtgggcgagcgca
cggttttggcccttgttagaggcccccgtAAAAAAAAGGGGGGGcaattatgagagagctaatctatcgcgtgcgtgttca
aacttgagttAAAAAAAAGGGGGGGctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatAAAAAAAAGGGGGGGaccgaaagggaa
ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttAAAAAAAAGGGGGGG
```

Where are the implanted motifs?

```
atgaccgggatactgataaaaaaaagggggggggcgtacacattagataaacgtatgaagtacgttagactcggcgccgcc
acccctattttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaataaaaaaaaagggggggg
tgagtatccctgggatgacttaaaaaaaagggggggtgctctcccgatttttgaatatgtaggatcattcgccagggtccg
gctgagaattggatgaaaaaaaagggggggtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggag
tcccttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaataaaaaaaagggggggcttata
gtcaatcatgttcttgtgaatggatttaaaaaaaaggggggggaccgcttggcgcacccaaattcagtgtgggcgagcgca
cggttttggcccttgttagaggcccccgtaaaaaaaagggggggcaattatgagagagctaatctatcgcgtgcgtgttca
aacttgagttaaaaaaaagggggggctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcataaaaaaaagggggggaccgaaagggaa
ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttaaaaaaaaggggggg
```

Implanting Motif AAAAAAAAGGGGGGG with four mutations

```
atgaccgggatactgatAgAAgAAAGGttGGGggcgtacacattagataaacgtatgaagtacgttagactcggcgccgcc
acccctatttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaatacAAtAAAAcGGcGGG
tgagtatccctgggatgacttAAAAtAAtGGaGtGGtgctctcccgatttttgaatatgtaggatcattcgccagggtccg
gctgagaattggatgcAAAAAAAAGGGattGtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggag
tccctttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatAtAAtAAAGGaaGGGcttata
gtcaatcatgttcttgtgaatggatttAAcAAtAAGGGctGGgaccgcttggcgcacccaaattcagtgtgggcgagcgca
cggttttggcccttgttagaggccccgtAtAAAcAAGGaGGGccaattatgagagagctaatctatcgcgtgcgtgttca
aacttgagttAAAAAAtAGGGaGccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatActAAAAAGGaGcGGaccgaaagggaa
ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttActAAAAAGGaGcGG
```

Where are the implanted motifs???

```
atgaccgggatactgatagaagaaaggttgggggcgtacacattagataaacgtatgaagtacgttagactcggcgccgcc
acccctattttttgagcagatttagtgacctggaaaaaaaatttgagtacaaaacttttccgaatacaataaaacggcggg
tgagtatccctgggatgacttaaaataatggagtggtgctctcccgatttttgaatatgtaggatcattcgccagggtccg
gctgagaattggatgcaaaaaaagggattgtccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggag
tcccttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaatataataaaggaagggcttata
gtcaatcatgttcttgtgaatggatttaacaataagggctgggaccgcttggcgcacccaaattcagtgtgggcgagcgca
cggttttggcccttgttagaggcccccgtataaacaaggagggccaattatgagagagctaatctatcgcgtgcgtgttca
aacttgagttaaaaaatagggagccctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt
ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcatactaaaaaggagcggaccgaaagggaa
ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagcttactaaaaaggagcgg
```

Why finding (15,4)-motifs is hard?

atgaccgggatactgat$\boxed{\texttt{AgAAgAAAGGttGGG}}$ggcgtacacattagataaacgtatgaagtacgttagactcggcgccgcc

acccctatttttgagcagatttagtgacctggaaaaaaatttgagtacaaaacttttccgaata$\boxed{\texttt{cAAtAAAAcGGcGGG}}$

tgagtatccctgggatgactt$\boxed{\texttt{AAAAtAAtGGaGtGG}}$tgctctcccgattttgaatatgtaggatcattcgccagggtccg

gctgagaattggatg$\boxed{\texttt{cAAAAAAAGGGattG}}$tccacgcaatcgcgaaccaacgcggacccaaaggcaagaccgataaaggag

tccctttttgcggtaatgtgccgggaggctggttacgtagggaagccctaacggacttaat$\boxed{\texttt{AtAAtAAAGGaaGGG}}$cttata

gtcaatcatgttcttgtgaatggattt$\boxed{\texttt{AAcAAtAAGGGctGG}}$gaccgcttggcgcacccaaattcagtgtgggcgagcgca

cggttttggcccttgttagaggccccgt$\boxed{\texttt{AtAAAcAAGGaGGGc}}$caattatgagagagctaatctatcgcgtgcgtgttca

aacttgagtt$\boxed{\texttt{AAAAAAtAGGGaGcc}}$ctggggcacatacaagaggagtcttccttatcagttaatgctgtatgacactatgt

ttggcccattggctaaaagcccaacttgacaaatggaagatagaatccttgcat$\boxed{\texttt{ActAAAAAGGaGcGG}}$accgaaaggaa

ctggtgagcaacgacagattcttacgtgcattagctcgcttccggggatctaatagcacgaagctt$\boxed{\texttt{ActAAAAAGGaGcGG}}$

Aligning two first occurrences of the motif

```
AgAAgAAAGGttGGG
..|..|||.|..|||
cAAtAAAAcGGcGGG
```

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:

```
cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtacgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtacgtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaacgtacgtc
```

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

The Motif Finding Problem

- Given a random sample of DNA sequences:

```
cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtacgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtacgtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaacgtacgtc
```

- 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:

cctgatagacgctatctggctatcc$\boxed{\texttt{acgtacgt}}$aggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgat$\boxed{\texttt{acgtacgt}}$acaccggcaacctgaaacaaacgctcagaaccagaagtgc

aa$\boxed{\texttt{acgtacgt}}$gcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctt$\boxed{\texttt{acgtacgt}}$ataca

ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgtta$\boxed{\texttt{acgtacgt}}$c

The Motif Finding Problem (cont'd)

The motifs revealed with 2 mutations:

```
cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaCcgtacgGc
```

The Motif Finding Problem (cont'd)

The motifs revealed with 2 mutations:

cctgatagacgctatctggctatcc$\boxed{\texttt{aGgtacTt}}$aggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgat$\boxed{\texttt{CcAtacgt}}$acaccggcaacctgaaacaaacgctcagaaccagaagtgc

aa$\boxed{\texttt{acgtTAgt}}$gcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctt$\boxed{\texttt{acgtCcAt}}$ataca

ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgtta$\boxed{\texttt{CcgtacgG}}$c

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

Motif Matrix

Motifs		a	G	g	t	a	c	T	t
		C	c	A	t	a	c	g	t
		a	c	g	t	T	A	g	t
		a	c	g	t	C	c	A	t
		C	c	g	t	a	c	g	G
Count(Motifs)	A	3	0	1	0	3	1	1	0
	C	2	4	0	0	1	4	0	0
	G	0	1	4	0	0	0	3	1
	T	0	0	0	5	1	0	1	4
Consensus(Motifs)		A	C	G	T	A	C	G	T

Score(Motifs) $2+1+1+0+2+1+2+1=10$

- 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 Score($Motifs$)

Parameters

$k = 8$

DNA

$t = 5$

```
cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatCcAtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaCcgtacgGc
```

$n = 69$

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

Hamming Distance

- The Hamming distance $d(v, w)$ the number of mismatches between two k-mers v and w
- For example:

$$d(\mathrm{AAAAAA}, \mathrm{A{\color{red}C}AAA{\color{red}C}}) = 2$$

Computing Score

Motifs	a	G	g	t	a	c	T	t	2
	C	c	A	t	a	c	g	t	2
	a	c	g	t	T	A	g	t	2
	a	c	g	t	C	c	A	t	2
	C	c	g	t	a	c	g	G	2

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

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

Then

- $\mathrm{Score}(\mathit{Motifs}) = d(\mathrm{Consensus}(\mathit{Motifs}), \mathit{Motifs})$

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
- $\mathrm{Score}(Motifs) = d(Pattern, Dna)$

Motif Finding Problem = Median String Problem

- *Motifs*: output of Motif Finding
- *Pattern*: output of Median String
- $\mathrm{Score}(\mathit{Motifs}) = d(\mathit{Pattern}, \mathit{Dna})$

Why?

- If $\mathrm{Score}(\mathit{Motifs}) < d(\mathit{Pattern}, \mathit{Dna})$, we could choose $\mathrm{Consensus}(\mathit{Motifs})$ as a better *Pattern*
- If $\mathrm{Score}(\mathit{Motifs}) > d(\mathit{Pattern}, \mathit{Dna})$, we could choose the best match occurrences of *Pattern* as better *Motifs*

Median String Algorithm

MedianString(DNA, k)

```
1: $BestPattern \leftarrow$ AAA...A
2: for each $k$-mer $Pattern$ from AAA...A to TTT...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 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

a	G	g	t	a	c	T	t
C	c	A	t	a	c	g	t
a	c	g	t	T	A	g	t
a	c	g	t	C	c	A	t
C	c	g	t	a	c	g	G

Count(Motifs)

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

Profile(Motifs)

$$\begin{array}{c|cccccccc} A & .6 & 0 & .2 & 0 & .6 & .2 & .2 & 0 \\ C & .4 & .8 & 0 & 0 & .2 & .8 & 0 & 0 \\ G & 0 & .2 & .8 & 0 & 0 & 0 & .6 & .2 \\ T & 0 & 0 & 0 & 1 & .2 & 0 & .2 & .8 \end{array}$$

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

$$\begin{array}{llcccccccc} \textit{Profile} & A & .6 & 0 & .2 & 0 & .6 & .2 & .2 & 0 \\ & C & .4 & .8 & 0 & 0 & .2 & .8 & 0 & 0 \\ & G & 0 & .2 & .8 & 0 & 0 & 0 & .6 & .2 \\ & T & 0 & 0 & 0 & 1 & .2 & 0 & .2 & .8 \end{array}$$

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

- $\Pr(\text{AGGTACTT} \mid \textit{Profile}) = .6 \cdot .2 \cdot .8 \cdot 1 \cdot .6 \cdot .8 \cdot .2 \cdot .8 = 0.0073728$
- 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
  - $\Pr(\text{AGGTACTT} \mid \textit{Profile}) = 0.0073728$

$$\textit{Profile} \quad \begin{array}{ccccccccc} A & .6 & 0 & .2 & 0 & .6 & .2 & .2 & 0 \\ C & .4 & .8 & 0 & 0 & .2 & .8 & 0 & 0 \\ G & 0 & .2 & .8 & 0 & 0 & 0 & .6 & .2 \\ T & 0 & 0 & 0 & 1 & .2 & 0 & .2 & .8 \end{array}$$

Problem: Zero Probabilities

$$
\begin{array}{lcccccccccc}
\textit{Profile} & & \text{A} & .6 & 0 & .2 & 0 & .6 & .2 & .2 & 0 \\
 & & \text{C} & .4 & .8 & 0 & 0 & .2 & .8 & 0 & 0 \\
 & & \text{G} & 0 & .2 & .8 & 0 & 0 & 0 & .6 & .2 \\
 & & \text{T} & 0 & 0 & 0 & 1 & .2 & 0 & .2 & .8 \\
 & & & & & & & & & & \\
\text{Consensus} & & & \text{A} & \text{C} & \text{G} & \text{T} & \text{A} & \text{C} & \text{G} & \text{T}
\end{array}
$$

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

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

Pseudocounts

- Add one to all counts
- Avoids zero counts

Count	A	3	0	1	0	3	1	1	0
	C	2	4	0	0	1	4	0	0
	G	0	1	4	0	0	0	3	1
	T	0	0	0	5	1	0	1	4
PseudoCount	A	4	1	2	1	4	2	2	1
	C	3	5	1	1	2	5	1	1
	G	1	2	5	1	1	1	4	2
	T	1	1	1	6	2	1	2	5

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

$$
\text{PseudoCount} \quad \begin{array}{ccccccccc}
A & 4 & 1 & 2 & 1 & 4 & 2 & 2 & 1 \\
C & 3 & 5 & 1 & 1 & 2 & 5 & 1 & 1 \\
G & 1 & 2 & 5 & 1 & 1 & 1 & 4 & 2 \\
T & 1 & 1 & 1 & 6 & 2 & 1 & 2 & 5
\end{array}
$$

$$
\text{Profile} \quad \begin{array}{ccccccccc}
A & 4/6 & 1/6 & 2/6 & 1/6 & 4/6 & 2/6 & 2/6 & 1/6 \\
C & 3/6 & 5/6 & 1/6 & 1/6 & 2/6 & 5/6 & 1/6 & 1/6 \\
G & 1/6 & 2/6 & 5/6 & 1/6 & 1/6 & 1/6 & 4/6 & 2/6 \\
T & 1/6 & 1/6 & 1/6 & 6/6 & 2/6 & 1/6 & 2/6 & 5/6
\end{array}
$$

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 $\mathrm{Score}(Motif) < \mathrm{Score}(BestMotif)$ 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 algoriths 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

```
RandomizedMotifSearch($DNA, k, t$)
 1: randomly select $k$-mers $Motifs = (Motif_1, \ldots, Motif_t)$, one from each
    string in $DNA$
 2: $BestMotifs \leftarrow Motifs$
 3: while forever do
 4:     $Profile \leftarrow \mathrm{Profile}(Motifs)$
 5:     for $i \leftarrow 1$ to $t$ do
 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 $\mathrm{Score}(Motifs) < \mathrm{Score}(BestMotifs)$ then
 9:         $BestMotifs \leftarrow Motifs$
10:     else
11:         return $BestMotifs$
```

Why Randomized Motif Search Works?

- If *Motifs* is a random set, the expectation is that $\mathrm{Profile}(\mathit{Motifs})$ has about the same probability 0.25 for each symbol in each column
- If *Motifs* contains some of the true motifs, it is not random and $\mathrm{Profile}(\mathit{Motifs})$ reflects this
- Then $\mathrm{Profile}(\mathit{Motifs})$ 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 $p_1, \dots, p_n$ with $p_1/C, \dots p_1/C$, 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 $Motifs = (Motif_1, \ldots, Motif_t)$ in each string
   in $Dna$
2: $BestMotifs \leftarrow Motifs$
3: for $j \leftarrow 1$ to $N$ do
4:     $i \leftarrow \mathrm{Random}(t)$
5:     $Profile \leftarrow$ profile matrix constructed from all strings in $Motifs$
       except for $Motif_i$
6:     $Motif_i \leftarrow$ profile-randomly generated $k$-mer in the $i$-th sequence in
       $DNA$
7:     if $\mathrm{Score}(Motifs) < \mathrm{Score}(BestMotifs)$ then
8:         $BestMotifs \leftarrow Motifs$
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