CS4049
Bioinformatics

Spring 2025

Rushda Muneer

Brute Force Approach

- **MotifEnumeration**(*Dna*, *k*, *d*)
  - *Patterns* ← an empty set
  - **for** each *k*-mer *Pattern* in *Dna*
    - **for** each *k*-mer *Pattern'* differing from *Pattern* by at most *d* mismatches
      - **if** *Pattern'* appears in each string from *Dna* with at most *d* mismatches
        - add *Pattern'* to *Patterns*
  - remove duplicates from *Patterns*
  - **return** *Patterns*

Limitations of the Implanted Motif Problem

- Real biological datasets (e.g., DNA arrays) are **noisy**.
- Not all identified genes contain the expected motif.
- A single missing sequence **invalidates** a (k, d)-motif.
- **A Better Approach:**
  - Score motifs based on **similarity** to an "ideal" motif.
  - The **ideal motif** is unknown, so we approximate it.
- **Motif Matrix Representation:**
  - A **t × k matrix** of selected k-mers.
  - **Most frequent nucleotide** in each column forms the **consensus string**.
  - Highly conserved positions indicate **strong motif presence**.
- Instead of exact motif matching, a **scoring-based approach** using scoring matrices helps handle noisy datasets more effectively.

Scoring Motifs Using Profile Matrices

- **Motif Representation:**
  - Given **t DNA strings** (each of length **n**), select a **k-mer** from each to form a **t × k motif matrix**.
- **Consensus String:**
  - The **most frequent nucleotide** in each column represents the consensus sequence.

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

Conservation Analysis:

- Some positions in the matrix are **highly conserved** (e.g., **G at positions 2 and 3**).
- Others are **variable**, indicating **weaker motif conservation** (e.g., position 12).

Scoring Motifs

- **Score Definition:**
  - **Score(Motifs)** = Number of **lower-case letters** in the motif matrix.
  - Goal: Minimize the **score** by selecting the most conserved k-mers.
- By selecting the best k-mers, we aim to find the most **conserved motif matrix** with the **fewest unpopular nucleotides**.

Scoring Motifs

Motifs	T C G G G G g T T T t t c C G G t G A c T T a C a C G G G G A T T T t C T t G G G G A c T T t t a a G G G G A c T T C C T t G G G G A c T T C C T C G G G G A T T c a t T C G G G G A T T c C t T a G G G G A a c T a C T C G G G t A T a a C C
Score(Motifs)	$3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30$

Exercise: The minimium possible value of *Score*(*Motifs*) is 0 (if all the *k*-mers in *Motifs* are the same). What is the maximum possible value of *Score*(*Motifs*) for 10 motifs of length 12?

Answer:84

Consensus Motif

- *Consensus*(*Motifs*) provides an ideal candidate regulatory motif for these regions.
- For example, the consensus string for the NF-κB binding sites in the figure is "TCGGGGATTTCC".

Motifs

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

Score(Motifs) $3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 = 30$

The Motif Finding Problem

- Given a collection of strings, find a set of k-mers (one from each string) that minimizes the motif score.
- **Input & Output:**
  - **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).
- **Brute Force Approach :**
  - Compute score of every possible choice of k-mers in the DNA to form a motif matrix
  - Take a collection of all k-mers with the lowest score
- **Time Complexity:**
  - This is computationally expensive for large sequences.

Time Complexity of Brute Force algorithm via scoring matrix calculation

- Assume there are t strings of length n
  - Selecting 1 k-mer per string gives one matrix
  - Total k-mers per string would be found in n-k+1
  - Matrix = t x k
  - Total locations per matrix $(n-k+1)^{t \cdot k}$
- Biologically k<<n and k << t
- There are $(n-k+1)^t$ possiblities of how to form a motif matrix
- Scoring the matrix requires k . t steps (frequency of each item in matrix)
- Overall Runtime: $O(n^t . k . t)$ -> very slow

Changing Perspective

- **Old Approach:**
  - Create motif matrices first with the best score
  - It will give us the consensus string
- **New Approach:**
  - Find the consensus string first
  - Look for a motif matrix that scores best against this consensus
- **How to do this?**
  - Instead of computing Score(Motifs) column-by-column, we can compute it row-by-row.

Changing Perspective

- **Column-wise Calculation:**
  - Previously calculated as the sum of lowercase letters in each column.
  - Example: Score of NF-κB motif matrix = **30** (sum of non-consensus elements column-wise).
- **Row-wise Calculation:**
  - Score can also be computed by summing mismatches row-by-row.
  - Each row's score corresponds to the **Hamming distance** between the consensus string and the motif in that row.

Hamming distance

$$d(Pattern, Motifs) = \sum_{i=1}^{t} HammingDistance(Pattern, Motif_i).$$

Motifs	T C G G G G g T T T t t	3
	c C G G t G A c T T a C	4
	a C G G G G A T T T t C	2
	T t G G G G A c T T t t	4
	a a G G G G A c T T C C	3
	T t G G G G A c T T C C	2
	T C G G G G A T T c a t	3
	T C G G G G A T T c C t	2
	T a G G G G A a c T a C	4
	T C G G G t A T a a C C	+ 3
SCORE(*Motifs*)	3 + 4 + 0 + 0 + 1 + 1 + 1 + 5 + 2 + 3 + 6 + 4 =	30
CONSENSUS(*Motifs*)	T C G G G G A T T T C C	

- Example: **d(" TCGGGGATTTCC ", " TCGGGGgTTTtt ") = 3**.
- Both methods yield the same score
- Score(Motif) = d(Consensus(Motif), Motifs)

Equivalent Motif Finding Problem:

- Given a collection of strings, find a **pattern** and **a collection of k-mers** (one from each string) that **minimizes the distance** between **all possible patterns** and **all possible collections** of k-mers.
- **Input:** A collection of strings (**Dna**) and an integer **k**.
- **Output:** A k-mer **Pattern** and a collection of k-mers (one from each string in **Dna**) that minimizes **d(Pattern, Motifs)**.

Distance between k-mer and a longer string

- Calculate the distance of not only same sized string (motifs) but also with a longer string (sequence)
- For different lengths, start from a smaller string (k-mer) and compare it with the first k-mer in the longer string.
- Continue until we find the lowest string.
- For example,
    - *d*("**GATTCTCA**", "GCAAA**GA**C**G**CT**G**A**C**CAA") = **3**.
- **GATTCTCA**
- GCAAA**GA**C**G**CT**G**A**C**CAA
- = 3

Distance between a *k*-mer and a (longer) String

Distance: *d*(GATTCTCA, GCAAAGACGCTGACCAA) = ?

Distance: 7

d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between a *k*-mer and a (longer) String

d(GATTCTCA, GCAAAGACGCTGACCAA) = ?

Distance: 7 **6**

d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between a *k*-mer and a (longer) String

d(GATTCTCA, GCAAAGACGCTGACCAA) = ?

Distance: 7 6 **7**

d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between a *k*-mer and a (longer) String

d(GATTCTCA, GCAAAGACGCTGACCAA) = ?

Distance: 7 6 7 **5**

d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between a *k*-mer and a (longer) String

d(GATTCTCA, GCAAAGACGCTGACCAA) = ?

Distance: 7 6 7 5 **8**

d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between a *k*-mer and a (longer) String

d(GATTCTCA, GCAAAGACGCTGACCAA) = ?

Distance: 7 6 7 5 8 **3**

d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between a *k*-mer and a (longer) String

d(GATTCTCA, GCAAAGACGCTGACCAA) = ?

Distance: 7 6 7 5 8 3 **8**

d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between a *k*-mer and a (longer) String

d(GATTCTCA, GCAAAGACGCTGACCAA) = ?

Distance: 7 6 7 5 8 3 8 **7**

d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between a *k*-mer and a (longer) String

d(GATTCTCA, GCAAAGACGCTGACCAA) = ?

Distance: 7 6 7 5 8 3 8 7 **4**

d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between a *k*-mer and a (longer) String

d(GATTCTCA, GCAAAGACGCTGACCAA) = ?

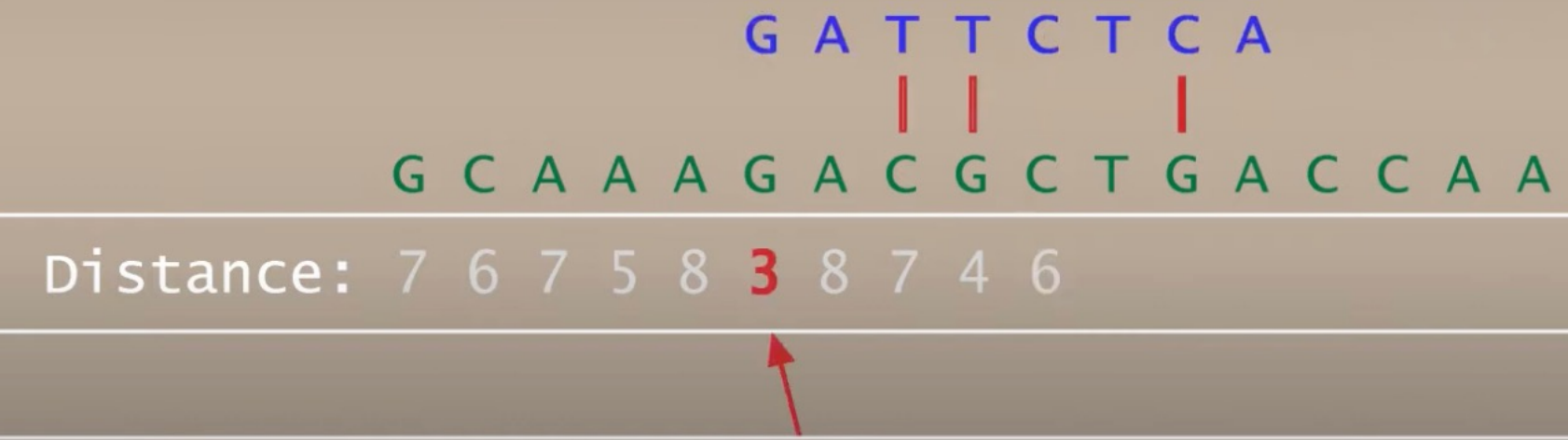
Distance: 7 6 7 5 8 3 8 7 4 **6**

d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between a *k*-mer and a (longer) String

d(GATTCTCA, GCAAAGACGCTGACCAA) = **3**



d(*Pattern*, *String*):

minimum distance between *Pattern* and all *k*-mers in *String*

Distance between k-mer and set of strings (motifs)

- For example, for the strings *Dna* shown below, the five colored 3-mers represent *Motifs*("AAA", *Dna*).

$$d(Pattern, Dna) = \sum_{i=1}^{t} d(Pattern, Dna_i).$$

- Given a *k*-mer *Pattern* and a set of strings *Dna* = {Dna_1, … , Dna_t}, we define *d*(*Pattern*, *Dna*) as the sum of distances between *Pattern* and all strings in *Dna*
- For example, *d*("AAA", *Dna*) = 1 + 1 + 2 + 0 + 1 = 5.

	ttaccttAAC	1
	gATAtctgtc	1
Dna	ACGgcgttcg	2
	ccctAAAgag	0
	cgtcAGAggt	1

Median String Problem

- A k-mer minimizing the distance with DNA and all possible k-mers motifs
- **Input:** A collection of strings Dna and an integer k.
- **Output:** A k-mer Pattern that minimizes dist(Pattern, Dna) among all possible k-mers.
- **Pseudocode:**
- **MedianString**(*Dna, k*)
  - *distance* ← ∞
  - **for** each *k*-mer *Pattern* from AA…AA to TT…TT
    - **if** *distance* > *d*(*Pattern*, *Dna*)
      - *distance* ← *d*(*Pattern*, *Dna*)
      - *Median* ← *Pattern*
  - **return** *Median*
- Finding a median string requires solving a **double minimization problem**.
- We must **find a *k*-mer *Pattern* that minimizes *d*(*Pattern*, *Dna*),** where this **function is itself computed by taking a minimum over all choices of *k*-mers** from each string in *Dna*.

Time Complexity of Median String

- 4 possible bases in k locations generate a k-mer (consensus candidate)
- Total possible candidates 4^k
- We compare the total hamming distance (k comparisons) across each string of length n
- Comparisons per string are n . K
- Total strings are t
- Overall runtime $O(4^k \cdot n \cdot k \cdot t)$

Comparison of Runtimes:

- **Median String Runtime:** (row wise)
  - $O(4^k \cdot n \cdot k \cdot t)$
- **Motif Search with Scoring Matrix Runtime:** (column wise)
  - $O(n^t \cdot k \cdot t)$
  - Slower than Median String when t (number of strings) is large.
- The length of a motif (k) typically does not exceed 20 nucleotides, whereas t is measured in the thousands.
- In the long run, Median string will be a slow yet better algorithm