CS4049 Bioinformatics

Spring 2025

Rushda Muneer

Count and Profile Matrices

- **Count(Motifs)** counts the occurrences of each nucleotide in the corresponding column of the motif matrix.
- Divide each element in the count matrix by **t**, the number of rows in the motif matrix.
- This results in a **profile matrix** P = Profile(Motifs), where each element **P(i, j)** is the frequency of nucleotide **i** in column **j**.
- Each column of the profile matrix sums to **1** (representing nucleotide frequencies).

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

Count(Motifs)	A:	2	2	0	0	0	0	9	1	1	1	3	0
	C:	1	6	0	0	0	0	0	4	1	2	4	6
	G:	0	0	10	10	9	9	1	0	0	0	0	0
	T:	7	2	0	0	1	1	0	5	8	7	3	4

Profile(Motifs)	A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
	C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
	G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
	T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4

Using the Profile Matrix to Roll Dice

- Let *Motifs* be a collection of *k*-mers taken from *t* strings *Dna*
- View each column of *Profile*(*Motifs*) as **a four-sided biased die**
- A profile matrix with ***k* columns** can be viewed as a **collection of *k* dice**, which we will **roll to randomly generate a *k*-mer**

Example: The probability Pr("ACGGGGATTACC" | *Profile*) that *Profile* generates "ACGGGGATTACC" is computed by simply multiplying the entries of each nucleotide in respective column of the profile matrix.

$$\texttt{Profile}\quad \begin{array}{l|cccccccccccc}
\texttt{A:} & .2 & .2 & 0 & 0 & 0 & 0 & .9 & .1 & .1 & .1 & .3 & 0 \\
\texttt{C:} & .1 & .6 & .0 & 0 & 0 & 0 & 0 & .4 & .1 & .2 & .4 & .6 \\
\texttt{G:} & 0 & 0 & 1 & 1 & .9 & .9 & .1 & 0 & 0 & 0 & 0 & 0 \\
\texttt{T:} & .7 & .2 & 0 & 0 & .1 & .1 & 0 & .5 & .8 & .7 & .3 & .4
\end{array}$$

$$\begin{aligned}\texttt{Pr(ACGGGGATTACC, Profile)} &= .2 \cdot .6 \cdot 1 \cdot 1 \cdot .9 \cdot .9 \cdot .9 \cdot .5 \cdot .8 \cdot .1 \cdot .4 \cdot .6 \\ &= 0.000839808\end{aligned}$$

The probablity to generate "ACGGGGATTACC" based upon the given profile matrix is 0.000839808

Profile - Most Probable k-mer

- K-mer with the highest Pr(k-mer | Profile) among all k-mers in the sequence

Example: Finding profile-most probable 6-mer in **CTATAAACCTTACAT**

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

6-mer	Pr (6-mer | Profile)	
CTATAAACCTTACAT	1/8 x 1/8 x 3/8 x 0 x 1/8 x 0	0
C**TATAAA**CCTTACAT	1/8 x 7/8 x 0 x m0 x 1/8 x 0	0
CT**ATAAAC**CTTACAT	1/2 x 1/8 x 3/8 x 0 x 1/8 x 0	0
CTA**TAAACC**TTACAT	1/8 x 7/8 x 3/8 x 0 x 3/8 x 0	0
CTAT**AAACCT**TACAT	1/2 x 7/8 x 3/8 x 5/8 x 3/8 x 7/8	0.0336
CTATA**AACCTT**ACAT	1/2 x 7/8 x 1/2 x 5/8 x 1/4 x 7/8	0.0299
CTATAA**ACCTTA**CAT	1/2 x 0 x 1/2 x 0 x 1/4 x 0	0
CTATAAA**CCTTAC**AT	1/8 x 0 x 0 x 0 x 1/8 x 0	0
CTATAAAC**CTTACA**T	1/8 x 1/8 x 0 x 0 x 3/8 x 0	0
CTATAAACC**TTACAT**	1/8 x 1/8 x 3/8 x 5/8 x 1/8 x 7/8	0.0004

Greedy Motif Search

1. Select a **k-mer,** Motif$_1$ in DNA$_1$.
2. Construct a **profile matrix,** Profile for this lone k-mer.
3. Set Motif$_2$ as the **Profile-most probable k-mer** in DNA$_2$.
4. **Update** Profile using Motif$_1$ and Motif$_2$.
5. Set Motif$_3$ as the **Profile-most probable k-mer** in DNA$_3$.
6. Repeat for each **DNA$_i$**, selecting the Profile-most probable k-mer.
7. After obtaining a k-mer from each string, **form a collection Motifs**.
8. Compare Motifs with the **current best scoring motif collection**.
9. If the new Motifs collection has a **higher score**, update the **best collection**.
10. Move Motif$_1$ one symbol over in Dna1 and repeat the process.

Analyzing Greedy Motif Finding

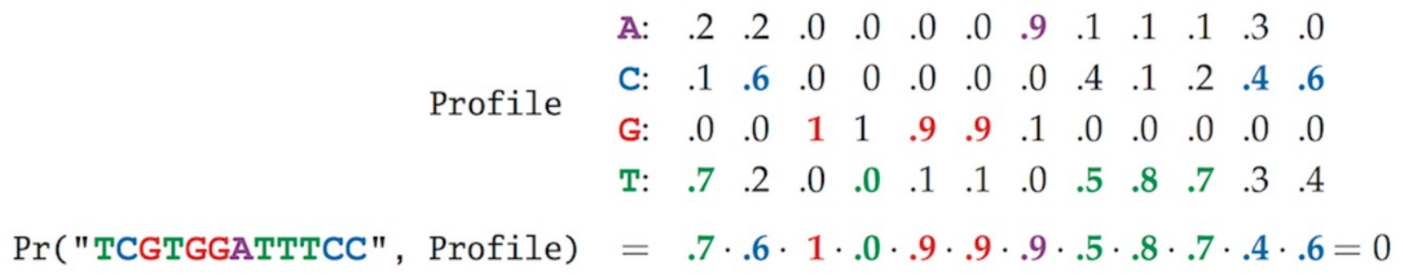
- GreedyMotifSearch() is significantly faster than MedianString().
- **Speed vs. Accuracy Trade-off:**
  - GreedyMotifSearch() sacrifices accuracy for speed.
  - Returns "gtAAAtAgaGatGtG" with a distance of 8.
  - The true implanted motif is "AAAAAAAAGGGGGGG", showing a clear deviation.
- While fast, GreedyMotifSearch() does not always find the most accurate motif.
- Useful for heuristic solutions but may require refinement or additional strategies for precise motif discovery.

Laplace’s Rule of Succession

Zero Probability Issue:

•The fourth symbol of " TCGTGGATTTCC " results in Pr(" TCGTGGATTTCC ", Profile) being zero.

•This causes the entire string to have a probability of zero, despite differing from the consensus string at only one position.



Profile

A:	.2	.2	.0	.0	.0	.0	.9	.1	.1	.1	.3	.0
C:	.1	.6	.0	0	.0	.0	.0	.4	.1	.2	.4	.6
G:	.0	.0	1	1	.9	.9	.1	.0	.0	.0	.0	.0
T:	.7	.2	.0	.0	.1	.1	.0	.5	.8	.7	.3	.4

$$\text{Pr("TCGTGGATTTCC", Profile)} = .7 \cdot .6 \cdot 1 \cdot .0 \cdot .9 \cdot .9 \cdot .9 \cdot .5 \cdot .8 \cdot .7 \cdot .4 \cdot .6 = 0$$

•A simple method for introducing **pseudocounts** (a small number used instead of zero).

•Inspired by Laplace’s principle used to estimate rare event probabilities.

•In motif finding, typically involves adding 1 (or another small number) to each element of Count(Motifs).

Laplace’s Rule of Succession

Example: we have the following motif, count, and profile matrices

$$\texttt{Motifs} \quad \begin{array}{cccc} T & A & A & C \\ G & T & C & T \\ A & C & T & A \\ A & G & G & T \end{array}$$

$$\texttt{Count(Motifs)} \quad \begin{array}{lcccc} A: & 2 & 1 & 1 & 1 \\ C: & 0 & 1 & 1 & 1 \\ G: & 1 & 1 & 1 & 0 \\ T: & 1 & 1 & 1 & 2 \end{array} \qquad \texttt{Profile(Motifs)} \quad \begin{array}{cccc} 2/4 & 1/4 & 1/4 & 1/4 \\ 0 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 0 \\ 1/4 & 1/4 & 1/4 & 2/4 \end{array}$$

Laplace’s Rule of Succession adds 1 to each element of *Count*(*Motifs*), updates the two matrices

$$\texttt{Count(Motifs)} \quad \begin{array}{lcccc} A: & 2+1 & 1+1 & 1+1 & 1+1 \\ C: & 0+1 & 1+1 & 1+1 & 1+1 \\ G: & 1+1 & 1+1 & 1+1 & 0+1 \\ T: & 1+1 & 1+1 & 1+1 & 2+1 \end{array} \qquad \texttt{Profile(Motifs)} \quad \begin{array}{cccc} 3/8 & 2/8 & 2/8 & 2/8 \\ 1/8 & 2/8 & 2/8 & 2/8 \\ 2/8 & 2/8 & 2/8 & 1/8 \\ 2/8 & 2/8 & 2/8 & 3/8 \end{array}$$

Randomized Motif Search

- Begin from a collection of randomly chosen k-mers *Motifs* in *Dna*
- Construct *Profile*(*Motifs*), and use this profile to generate a new collection of *k*-mers
  - *Motifs*(*Profile*(*Motifs*), *Dna*)
- Hope is that *Motifs*(*Profile*(*Motifs*), *Dna*) has a better score than the original collection of k-mers *Motifs*
- Form the profile matrix of these *k*-mers
  - *Profile*(*Motifs*(*Profile*(*Motifs*), *Dna*))
- Use it to form the most probable *k*-mers
  - *Motifs*(*Profile*(*Motifs*(*Profile*(*Motifs*), *Dna*)), *Dna*)
- Continue to iterate. . .
  - ...*Profile*(*Motifs*(*Profile*(*Motifs*(*Profile*(*Motifs*), *Dna*)), *Dna*))...
- As long as the score of the constructed motifs keeps improving

Entropy

- A measure of the uncertainty of a probability distribution (p_1, …, p_N), and is defined as follows:

$$H(p_1, \ldots, p_N) = -\sum_{i=1}^{N} p_i \cdot \log_2 p_i$$

Profile

A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4

Examples of Entropy Calculations:

2nd column: H(0.2, 0.6, 0.0, 0.2) → – (0.2 $\log_2$ 0.2 + 0.6 $\log_2$ 0.6 + 0.0 $\log_2$ 0.0 + 0.2 $\log_2$ 0.2)≈1.371

12th column: H(0.0, 0.6, 0.0, 0.4) → – (0.0 $\log_2$ 0.0 + 0.6 $\log_2$ 0.6 + 0.0 $\log_2$ 0.0 + 0.4 $\log_2$ 0.4)≈0.971

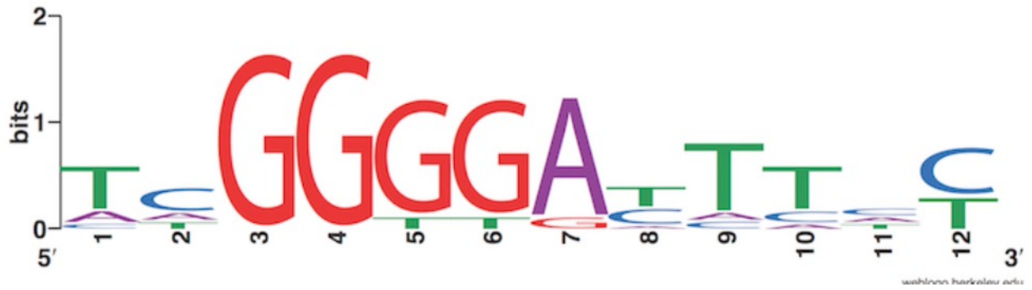
5th column: H(0.0, 0.0, 0.9, 0.1) → – (0.0 $\log_2$ 0.0 + 0.0 $\log_2$ 0.0 + 0.9 $\log_2$ 0.9 + 0.1 $\log_2$ 0.1)≈0.467

Entropy

- **STOP and Think:** What are the **maximum** and **minimum** possible values for the entropy of **a probability distribution** containing **four** values?
- **Max** = $-1 * (0.25 \log_2 0.25 + 0.25 \log_2 0.25 + 0.25 \log_2 0.25 + 0.25 \log_2 0.25) = 2.0$
- **Min** = $-1 * \log_2 1 = 0.0$
- **Minimum and Maximum Entropy:**
  - A completely conserved column has entropy = 0 (minimum entropy).
  - A column with equally-likely nucleotides (each probability = 1/4) has maximum entropy = 2.
  - The more conserved a column, the smaller its entropy.

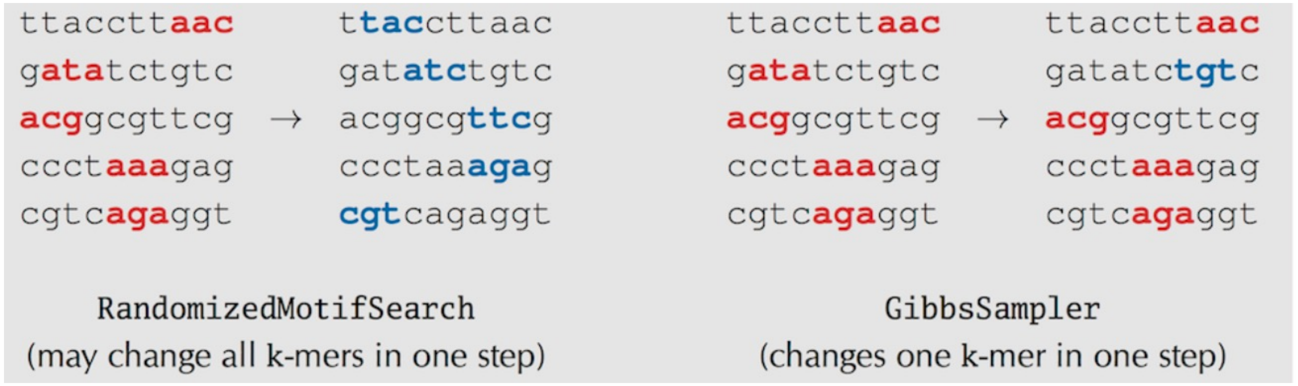
Motif Logo

- A diagram for visualizing motif conservation that consists of a stack of letters at each position
- The relative sizes of letters indicate their frequency in the column
- The total height of the letters in a column is based on the **information content** of the column
  - Defined as 2 − $H(p_1, \ldots, p_N)$
  - Lower entropy = higher information content.
  - Taller columns in the motif logo represent highly conserved positions.



Gibbs Sampling

- A more cautious iterative algorithm.
- Discards only a single k-mer at each iteration unlike Randomized Motif Search which may change all motifs in a single iteration.
- **GibbsSampler()** randomly selects an integer *i* between 1 and *t*, and then randomly changes only a single *k*-mer *Motif*$_i$.



Proposal Submission

- Sunday, February 9th, 2025 11:59 pm
- 2 pages Proposal
- Project title
- Group members (2-3), contact details
- Abstract (250 words)
- Introduction, Literature Review
- Proposed Methodology, Tentative/algorithms or tools
- Expected outcome or deliverables
- References

Quiz 1

- Wednesday, February 12th, 2025 11:59 pm
- Chapter 1 and 2 (all lecture slides 1.1 to 3.1)

Assignment 2

- Wednesday, February 12th, 2025 11:59 pm
- Programming based with some related questions