CS2220: Introduction to Computational Biology
Greedy Motif Search

Somayyeh Koohi

Fall 2024

Materials for this presentation have been adapted from the following sources:

Suprakash Datta, http://www.cse.yorku.ca/course/5290

Outline

- Randomized Algorithms
- Greedy Profile Motif Search
- Gibbs Sampler

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

- **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.

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.

Profiles Revisited

- Let **s**=(s_1,...,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* x *l* ***alignment matrix*** and
  - 4 x *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

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

$$Prob(\mathbf{a}|\mathbf{P}) = \prod_{i=1}^{n} p_{a_i,i}$$

Scoring Strings with a Profile
(cont'd)

Given a profile: **P** =

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

The probability of the consensus string:
Prob(aaacct|P) = ???

Scoring Strings with a Profile
(cont'd)

Given a profile: **P** =

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

The probability of the consensus string:

Prob(aaacct|P) = 1/2 x 7/8 x 3/8 x 5/8 x 3/8 x 7/8 = .033646

Scoring Strings with a Profile
(cont'd)

Given a profile: **P** =

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**

The probability of the consensus string:

Prob(aaacct|P) = 1/2 x 7/8 x 3/8 x 5/8 x 3/8 x 7/8 = .033646

Probability of a different string:

Prob(atacag|P) = 1/2 x 1/8 x 3/8 x 5/8 x 1/8 x 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 =

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

Given a sequence = ctataaaccttacatc, find the P-most probable *l*-mer

P-Most Probable *l*-mer (cont'd)

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

Find the *Prob*(a|P) of every possible 6-mer:

First try: c t a t a a a c c t t a c a t c

Second try: c t a t a a a c c t t a c a t c

Third try: c t a t a a a c c t t a c a t c

-Continue this process to evaluate every possible 6-mer

P-Most Probable *l*-mer (cont'd)

Compute *prob*(a|P) for every possible 6-mer:

String, Highlighted in Red	Calculations	*prob*(**a**|**P**)
ctataaaccttacat	1/8 x 1/8 x 3/8 x 0 x 1/8 x 0	0
ctataaaccttacat	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 aaacct:

String, Highlighted in Red	Calculations	*Prob*(a|P)
ctataaaccttacat	1/8 x 1/8 x 3/8 x 0 x 1/8 x 0	0
ctataaaccttacat	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)

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

ctataaaccttacatc

because *Prob*(aaacct|P) = .0336 is greater than the *Prob*(a|P) of any other 6-mer in the sequence.

P-Most Probable *l*-mers in Many Sequences

- Find the **P**-most probable *l*-mer in each of the sequences.

P=

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

```
ctataaacgttacatc
atagcgattcgactg
cagcccagaaccct
cggtataccttacatc
tgcattcaatagctta
tatcctttccactcac
ctccaaatcctttaca
ggtcatcctttatcct
```

P-Most Probable *l*-mers in Many Sequences (cont'd)

1	a	a	a	c	g	t
2	a	t	a	g	c	g
3	a	a	c	c	c	t
4	g	a	a	c	c	t
5	a	t	a	g	c	t
6	g	a	c	c	t	g
7	a	t	c	c	t	t
8	t	a	c	c	t	t
A	5/8	5/8	4/8	0	0	0
C	0	0	4/8	6/8	4/8	0
T	1/8	3/8	0	0	3/8	6/8
G	2/8	0	0	2/8	1/8	2/8

ctataaacgttacatc

atagcgattcgactg

cagcccagaaccct

cggtgaaccttacatc

tgcattcaatagctta

tgtcctgtccactcac

ctccaaatcctttaca

ggtctacctttatcct

P-Most Probable *l*-mers form a new profile

Comparing New and Old Profiles

1	a	a	a	c	g	t
2	a	t	a	g	c	g
3	a	a	c	c	c	t
4	g	a	a	c	c	t
5	a	t	a	g	c	t
6	g	a	c	c	t	g
7	a	t	c	c	t	t
8	t	a	c	c	t	t
A	5/8	5/8	4/8	0	0	0
C	0	0	4/8	6/8	4/8	0
T	1/8	3/8	0	0	3/8	6/8
G	2/8	0	0	2/8	1/8	2/8

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

Red – frequency increased, Blue – frequency descreased

Greedy Profile Motif Search

Use P-Most probable *l*-mers to adjust start positions until we reach a "best" profile; this is the motif.

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

Gibbs Sampling: an Example

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

$s_1=7$	GTAAACAATATTTATAGC
$s_2=11$	AAAATTTACCTTAGAAGG
$s_3=9$	CCGTACTGTCAAGCGTGG
$s_4=4$	TGAGTAAACGACGTCCCA
$s_5=1$	TACTTAACACCCTGTCAA

Gibbs Sampling: an Example

2) Choose one of the sequences at random:

Sequence 2: AAAATTTACCTTAGAAGG

$s_1=7$	GTAAACAATATTTATAGC
$s_2=11$	AAAATTTACCTTAGAAGG
$s_3=9$	CCGTACTGTCAAGCGTGG
$s_4=4$	TGAGTAAACGACGTCCCA
$s_5=1$	TACTTAACACCCTGTCAA

Gibbs Sampling: an Example

2) Choose one of the sequences at random:

Sequence 2: AAAATTTACCTTAGAAGG

$s_1=7$ GTAAACAATATTTATAGC

$s_3=9$ CCGTACTGTCAAGCGTGG

$s_4=4$ TGAGTAAACGACGTCCCA

$s_5=1$ TACTTAACACCCTGTCAA

Gibbs Sampling: an Example

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

1	A	A	T	A	T	T	T	A
3	T	C	A	A	G	C	G	T
4	G	T	A	A	A	C	G	A
5	T	A	C	T	T	A	A	C
A	1/4	2/4	2/4	3/4	1/4	1/4	1/4	2/4
C	0	1/4	1/4	0	0	2/4	0	1/4
T	2/4	1/4	1/4	1/4	2/4	1/4	1/4	1/4
G	1/4	0	0	0	1/4	0	3/4	0
Consensus String	T	A	A	A	T	C	G	A

Gibbs Sampling: an Example

4) Calculate the *prob*(***a***|***P***) for every possible 8-mer in the removed sequence:

Strings Highlighted in Red	*prob*(**a**|**P**)
AAAATTTACCTTAGAAGG	.000732
AAAATTTACCTTAGAAGG	.000122
AAAATTTACCTTAGAAGG	0
AAAATTTACCTTAGAAGG	0
AAAATTTACCTTAGAAGG	0
AAAATTTACCTTAGAAGG	0
AAAATTTACCTTAGAAGG	0
AAAATTTACCTTAGAAGG	.000183
AAAATTTACCTTAGAAGG	0
AAAATTTACCTTAGAAGG	0
AAAATTTACCTTAGAAGG	0

Gibbs Sampling: an Example

5) Create a distribution of probabilities of *l*-mers *prob*(*a*|*P*), and randomly select a new starting position based on this distribution.

a) To create this distribution, divide each probability *prob*(*a*|*P*) by the lowest probability:

Starting Position 1: *prob(* AAAATTTA | P) = .000732 / .000122 = 6

Starting Position 2: *prob(* AAATTTAC | P) = .000122 / .000122 = 1

Starting Position 8: *prob(* ACCTTAGA | P) = .000183 / .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): 6/(6+1+1.5)= 0.706

Probability (Selecting Starting Position 2): 1/(6+1+1.5)= 0.118

Probability (Selecting Starting Position 8): 1.5/(6+1+1.5)=0.176

Gibbs Sampling: an Example

c) Select the start position according to computed ratios:

P(selecting starting position 1): .706

P(selecting starting position 2): .118

P(selecting starting position 8): .176

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.

$s_1=7$	GTAAACAATATTTATAGC
$s_2=1$	AAAATTTACCTCGCAAGG
$s_3=9$	CCGTACTGTCAAGCGTGG
$s_4=5$	TGAGTAATCGACGTCCCA
$s_5=1$	TACTTCACACCCTGTCAA

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.