Randomized algorithms

Torgeir R. Hvidsten

Outline

- Randomized algorithms
- Greedy profile motif search
- Gibbs sampler

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 correct polynomial algorithm is known

Two types of randomized algorithms

- **Las Vegas Algorithms** – always produce the correct solution

- **Monte Carlo Algorithms** – do not always return the correct solution

- Las Vegas Algorithms are always preferred, but they are often hard to come by

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

Profiles

- 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** and
  - - 4 x *l* **profile P**

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

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

where $p_{a_i,\, i}$ is the frequency of nucleotide a_i in position *i* in the profile

Scoring strings with a profile

Given a profile: **P** =

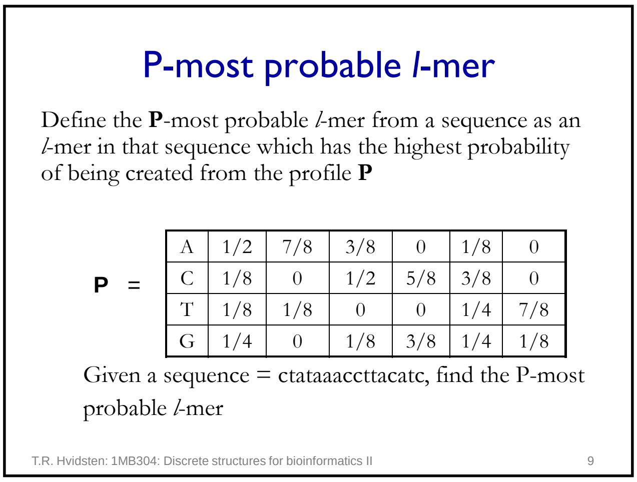
A	1/2	7/8	3/8	0	1/8	0
C	1/8	0	1/2	5/8	3/8	0
T	1/8	1/8	0	0	1/4	7/8
G	1/4	0	1/8	3/8	1/4	1/8

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

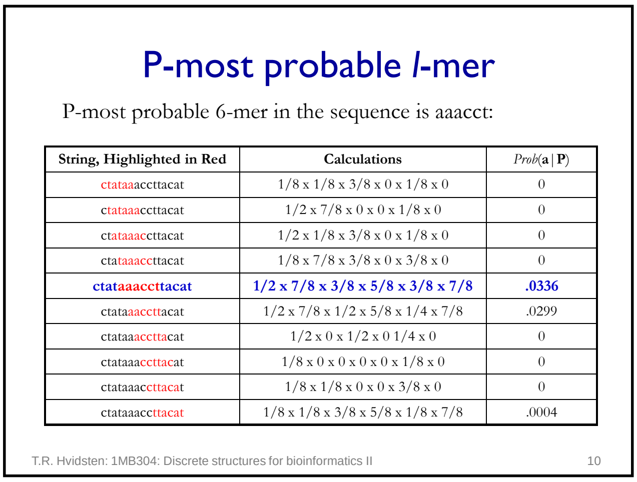


P-most probable *l*-mer

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**

$$\mathbf{P} = \begin{array}{|c|c|c|c|c|c|c|}\hline A & 1/2 & 7/8 & 3/8 & 0 & 1/8 & 0 \\ \hline C & 1/8 & 0 & 1/2 & 5/8 & 3/8 & 0 \\ \hline T & 1/8 & 1/8 & 0 & 0 & 1/4 & 7/8 \\ \hline G & 1/4 & 0 & 1/8 & 3/8 & 1/4 & 1/8 \\ \hline \end{array}$$

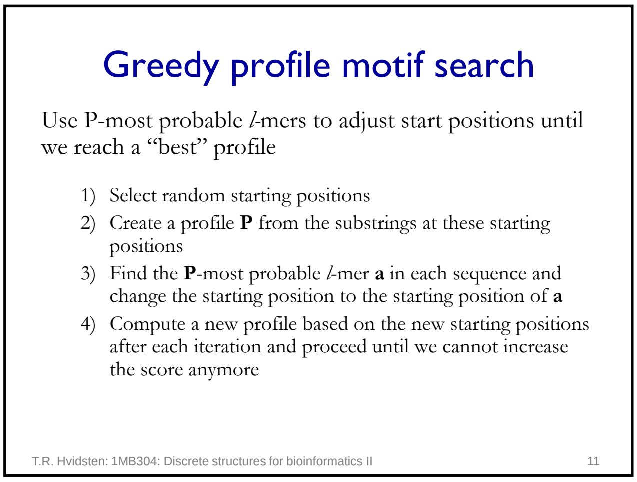
Given a sequence = ctataaaccttacatc, find the P-most probable *l*-mer



P-most probable *l*-mer

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

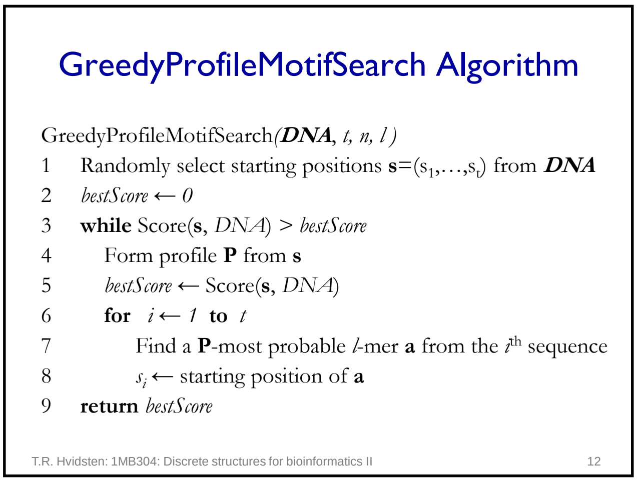
String, Highlighted in Red	**Calculations**	*Prob*(**a** | **P**)
ctataaaccttacat	1/8 x 1/8 x 3/8 x 0 x 1/8 x 0	0
ctataaaccttacat	1/2 x 7/8 x 0 x 0 x 1/8 x 0	0
ctataaaccttacat	1/2 x 1/8 x 3/8 x 0 x 1/8 x 0	0
ctataaaccttacat	1/8 x 7/8 x 3/8 x 0 x 3/8 x 0	0
ctataaaccttacat	**1/2 x 7/8 x 3/8 x 5/8 x 3/8 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



Greedy profile motif search

Use P-most probable *l*-mers to adjust start positions until we reach a "best" profile

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

```
GreedyProfileMotifSearch(DNA, t, n, l)
Randomly select starting positions s=($s_1$,…,$s_t$) from DNA
bestScore ← 0
while Score(s, DNA) > bestScore
    Form profile P from s
    bestScore ← Score(s, DNA)
    for  i ← 1  to  t
        Find a P-most probable l-mer a from the $i^{th}$ sequence
        $s_i$ ← starting position of a
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, ***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

3) Create profile ***P*** from *l*-mers in the 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

To create a proper distribution, divide each probability prob(***a*** | ***P***) by the sum of probabilities over all position:

Probability (Selecting Starting Position 1)	= 0.706
Probability (Selecting Starting Position 2)	= 0.118
…	
Probability (Selecting Starting Position 8)	= 0.176

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

Relative entropy

Repeats often make motif finding difficult

Solution: Incorporate background frequencies
to find biologically significant motifs:

$$\sum_{j=1}^{l} \sum_{r \in \{A,T,C,G\}} p_{rj} \log_2 \frac{p_{rj}}{br}$$

where p_{rj} is the frequency of nucleotide r in position j
and b_r is the background frequency of r