Exhaustive Search: Motif Finding

Motif Discovery and Median String Detection

Informally. A *regulatory motif* is a nucleotide sequence widespread in DNA and conjectured to have some biological significance. *Motif* discovery *in-silico* usually involves spotting nucleotide sequences of certain lengths that tend to repeat in DNA significantly more often than the probability theory would dictate, if the occurrences were purely random. While finding repeated substrings in a string can be done using suffix trees (see previous handouts), *motifs* often vary from one another by a small number of differences. Figure 1 shows an example of a motif `AATGCTGA` as expressed in six different occurrences, each of which differs from the motif and other occurrences in at most two positions.

Informally, the motif finding problem is the problem of discovery of related patterns of a specified length in a collection of DNA fragment. We formalize this definition below.

Alignment Matrix. Let $D = (D_1, \ldots, D_t)$ be a collection of DNA strings in nucleotide alphabet $\{A, C, G, T\}$. Let $|D_i| = n$ and let $l < n$ be the length of the desired regulatory motif. Let $s = (s_1, \ldots, s_t)$, $1 \leq s_i \leq n - l + 1$ be a list of *starting positions* in strings from D. An *alignment matrix* $A(s)$ is a set of l-mers (strings of length l)from $D_1, \ldots D_t$ starting at positions $s_1, \ldots, s_t$ respectively.

Example. Figure 2 shows an alignment matrix of six DNA strings for a collection of positions $s = (4, 3, 5, 4, 3, 6)$.

Motif:	A	A	T	G	C	T	G	A
Sequence 1:	A	<u>T</u>	T	G	C	T	G	A
Sequence 2:	A	A	T	<u>C</u>	C	T	G	A
Sequence 3:	A	A	T	G	C	<u>A</u>	G	A
Sequence 4:	<u>T</u>	A	T	G	C	T	G	A
Sequence 5:	A	A	T	G	C	T	<u>C</u>	A
Sequence 6:	A	A	T	G	<u>G</u>	T	<u>C</u>	A

Figure 1: Motif `AATGCTGA` as expressed in six different occurrences.

		$A(s)$	
D_1:	ATC	ATTGCTGA	CCGTAT
D_2:	GC	AATCCTGA	TGTCCGT
D_3:	AGGCT	AATGCAGA	TATG
D_4:	TTAC	TATGCTGA	GTAAT
D_5:	TAT	AATGCTCA	TCATCC
D_6:	TGCAG	AATGGTCA	TTAT

Figure 2: Alignment matrix.

D_1:	ATC	A	T	T	G	C	T	G	A	CCGTAT
D_2:	GC	A	A	T	C	C	T	G	A	TGTCCGT
D_3:	AGGCT	A	A	T	G	C	A	G	A	TATG
D_4:	TTAC	T	A	T	G	C	T	G	A	GTAAT
D_5:	TAT	A	A	T	G	C	T	C	A	TCATCC
D_6:	TGCAG	A	A	T	G	G	T	C	A	TTAT
	A:	5	5	0	0	0	1	0	6	
P(s):	C:	0	0	0	1	5	0	2	0	
	G:	0	0	0	5	1	0	4	0	
	T:	1	1	6	0	0	5	0	0	
		A	A	T	G	C	T	G	A	

Figure 3: Profile matrix and the consensus string.

Profile matrix. Given a collection of DNA strings $D = (D_1, \ldots, D_t)$ and a list of positions $s = (s_1, \ldots, s_t)$, a *profile matrix* $P(s)$ is a function

$$P : \{1, \ldots, l\} \times \{A, C, T, G\} \to \{1, \ldots, l\},$$

defined as follows:

$P(s)(i, X) =$ number of occurrences of letter X at position i in $A(s)$.

Example. The profile matrix for the alignment matrix from Figure 2 is shown in Figure 3.

Consensus sequence. The consensus sequence for an alignment matrix $A(s)$ is a string formed out of the nucleotides, most frequently occurring in each position.

Example. The consensus sequence for the alignment matrix from Figure 2 is shown at the bottom of Figure 3.

Consensus score. Given an alignment matrix $A(s)$ its *consensus score* (or the consensus score of its starting position s) is defined as

$$Score(s) = \sum_{j=1}^{t} \max_{X \in \{A,C,G,T\}} P(s)(i, X)$$

Example. The *consensus score* of the alignment matrix $A(s)$ from Figure 2 is:

$$Score(s) = 5 + 5 + 6 + 5 + 5 + 5 + 4 + 6 = 41.$$

Note. When all strings in $A(s)$ are in full agreement, the consensus score will reach its maximum $Score(s) = t \cdot l$. The smallest possible consensus score is $Score(s) = \frac{t \cdot l}{4}$, which is achieved when $P(s)(i, A) = P(s)(i, C) = P(s)(i, G) = P(s)(i, T)$ for all $i = 1, \ldots l$.

As such, the maximum possible score for an alignment of six strings of length 8 is $Score(s) = 6 \cdot 8 = 48$. A score of 41 shows a high degree of agreement on the consensus string among the six strings from the collection D.

Motif Finding Problem. The formal definition of the *motif finding problem* is:

> *Given a collection of DNA strings $D = (D_1, \ldots, D_t)$, each of length n, and an integer number $0 < l \leq n$, find the list of starting positions (alignment) $s = (s_1, \ldots, s_t$ which maximizes the consensus score $Score(s)$ for all l-mer alignment matrices.*

Median String Problem

Median String problem is equivalent to the *motif finding* problem.

Hamming Distance. Let $S = s_1 \ldots s_l$ and $T = t_1 \ldots t_l$ be two strings of length l in the same alphabet Σ. **Hamming distance** between S and T, denoted $d_H(S, T)$ is total number of positions i in which $s_i \neq t_i$.

For example, for $S =$`ATTCGAT` and $T =$`ATGGCAT`, $d_H(S, T) = 3$.

Definition. Let $D = (D_1, \ldots, D_t)$ be a collection of DNA strings in the $\{A, C, G, T\}$ alphabet and l be an integer such that $0 < l \leq n = |D_1 = \ldots = |D_t|$. Let $s = (s_1, \ldots, s_t)$ be a list of starting positions in $D_1, \ldots D_t$ respectively, and let us denote the set of all lists of starting positions as $\mathcal{S}$. Let $A(s)$ be the alignment matrix for s. Let $v = v_1 \ldots v_l$ be a string in the alphabet $\{A, C, G, T\}$.

The **Hamming distance** between v and $A(s)$, denoted $d_H(v, s)$ is defined as

$$d_H(v, s) = \sum_{j=1}^{t} d_H(v, D_j[s_j : s_{j+l}]).$$

The total distance between a string v and the collection D is defined as

$$TotalDistance(v, D) = \min_{s \in \mathcal{S}} d_H(v, s).$$

A string $v = v_1 \ldots v_l$ is a **median** of the collection $D = (D_1, \ldots D_t)$ if $TotalDistance(v, D) \leq TotalDistance(v', D)$ for any other string v' of length l.

Median Finding Problem. Given a collection of DNA strings $D = (D_1, \ldots, D_t)$, each of length n, and an integer $0 < l \leq n$, find the median string of length l for D.

Note. The median finding problem and the motif finding problem are equivalent because the *consensus string* from the motif finding problem minimizes the Hamming distance between a string and a string collection.

Algorithm Ideas

Observations.

- For the motif finding problem, we need to iterate over all possibe starting sequences $s = (s_1, \ldots, s_t)$. There are $(n - l + 1)^t$ such seqences, and the optimal solution can, potentially be found at any of them.
- For the median string finding problem, we essentially need to find (or estimate) $TotalDistance(v, D)$ for **every** l-mer string in the $\{A, C, G, T\}$ alphabet. There are 4^t such strings.
- In either case we need to enumerate a space of possible solutions that are representable as sequence of known length.
- In either case, a naïve algorithm is to set up the enumeration, and then:
  - For each $s = (s_1, \ldots, s_t)$, compute $Score(s)$, and find the consensus string.
  - or, for each l-mer string $v = v_1 \ldots v_l$, compute $TotalDistance(v, D)$.
- Computation of $TotalDistance(v, D)$ is straightforward: for each string $D_i \in \{D_1, \ldots, D_t\}$, find the position s_i that minimizes $d_H(v, s_i)$. Take $s = (s_1, \ldots, s_t)$ computed this way, and add up $d_H(v, s_i)$ to obtain $TotalDistance(v, D)$.

So, solving either problem boils down to correctly and, if possible, efficiently enumerating the space of all possible solutions.

Search Trees

Definition. Consider a set $\mathcal{S}$ of all sequences $s = (s_1, \ldots, s_t)$, where $s_i \in \{1, \ldots K\}$. A search tree over $\mathcal{S}$ is a tree of depth $t + 1$ constructed as follows.

- Each non-leaf node in the tree has exactly K children.
- Each node in the tree has a label. The root has an empty label.
- The K children of the root have labels 1, 2, $\ldots K$.
- Given a node with a label s', its K children will have labels $s'1$, $s'2$, $\ldots$, $s'K$.

Proposition. The leaf nodes in a search tree over $\mathcal{S}$ enumerate every single sequence in $\mathcal{S}$.

Search trees for motif finding. For motif finding, $K = n - l + 1$, and the leaf labels will have the form $s = (s_1, \ldots, s_t)$, where t is the number of the DNA strings in D.

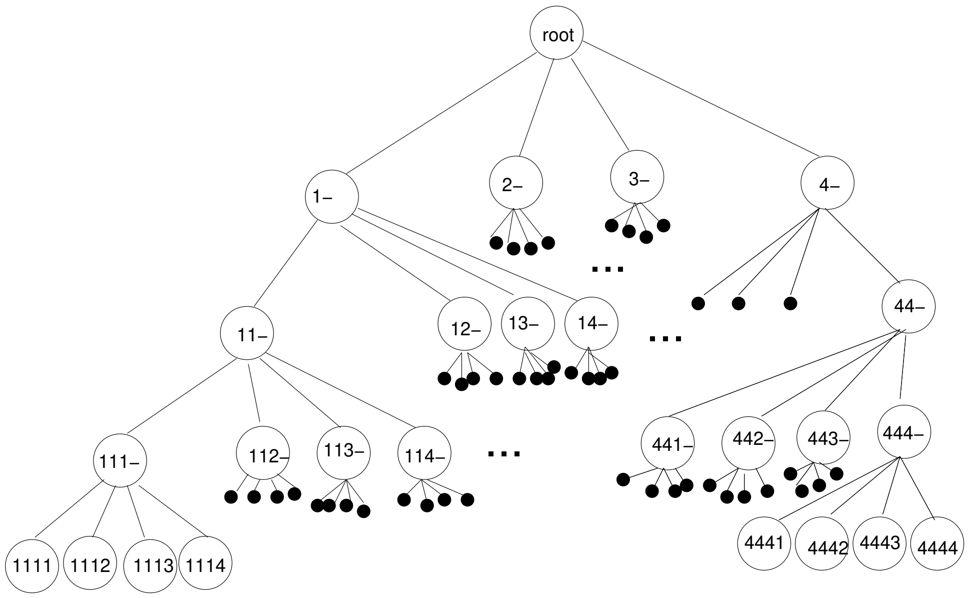

Figure 4: An example of a search tree.

Search trees for median string finding. For median string finding problem, we map $A = 1$, $C = 2$, $G = 3$, $T = 4$, set $K = 4$ and $t = l$. The labels will be l-mers in the alphabet $\{1, 2, 3, 4\}$, which uniquely map to l-mers in the nucleotide alphabet.

E.g., 1324 is `AGCT`, 11443 is `AATTG` and so on.

Example. Figure **??** shows how a search tree for a set of 4-mers in the $\{A, C, G, T\}$ alphabet would look.

Traversing the leaves of a search tree. Two simple algorithms NextLeaf() and AllLeaves() traverse the leaves of a search tree.

ALGORITHM NextLeaf(a[1..t],t,K)
begin `// a: current leaf; t: length of sequence; K: size of alphabet`
  for i= L **to** 1 **do**
    if $a[i] < K$ **then**
      $a[i] \leftarrow a[i] + 1$;
      return a;
    end if
    $a[i] \leftarrow 1$;
  end for
end

ALGORITHM AllLeaves(t,K)
begin
  $a = (1, \ldots, 1)$;
  repeat
   print a;
    $a \leftarrow$ NextLeaf(a, t, K);
  until $a = (1, \ldots, 1)$;
end

References

[1] S.S. Skiena, W.D. Smith, P. Lemke (1990) Reconstructing Sets From Interpoint Distances. In *Proc. Sixth Annual Symposium on Computational*

Geometry, pp. 332-339, Berkeley, CA, June 1990.