COS 551: Introduction to Computational Molecular Biology
Lecture: Oct 17, 2000
Lecturer: Mona Singh
Scribe: Jacob Brenner[1]

Database Searching

- In database search, we typically have a large sequence database (e.g., hundreds of thousands of sequences)
- Generally want to find similarity between a new query sequence and some known sequence in database.
- We look for similarity in *local* alignments (not global), because similarities often span only segments of the sequences involved. The reason the sequence similarity spans such short stretches is because of the modular fashion of protein evolution (mixing and matching of domains)

- Issues to consider:
    1) <u>Efficiency</u> – We need to do many comparisons, so cannot use the optimal but relatively slow (quadratic) algorithms discussed so far (e.g., Smith-Waterman)
    2) <u>Significance</u> – How do we tell when we find a significant match?

- Methods:
    <u>FASTA</u> :   http://www2.ebi.ac.uk/fasta3
                    Pearson & Lipman (1988). *PNAS* **85**, 2444-2448.
    <u>BLASTA</u> :   http://www.ncbi.nlm.nih.gov/BLAST
                    Altschul *et al*. (1990). *JMB* **215**, 403-440.
    Both methods are 10-100x faster than running Smith-Waterman. They are heuristics, so can also overlook (occasionally) similarities Smith-Waterman will find (i.e., they sacrifice sensitivity for speed).

We will now review the basic ideas behind the FASTA and BLAST heuristics, and then give a quick introduction to the statistics of local alignments.

FASTA
        Query sequence   s
        Target or text seq   t
- Search 2 sequences for common window of fixed size k
        For proteins $k = 2$
        For DNA $k = 4$-6
- Compare by sliding windows
- All k-tuples are stored in a table

Here is the idea of FASTA method:

Example:

```
        1   2   3   4   5   6   7
s =     A   G   A   G   A   G
t =     A   A   G   A   G   A   G
```

- Create a table of possible k-tuples (say, 4-tuples) for both s and t

e.g., possible 4-tuples of s are (formed by sliding a window of size 4):
```
s =     A   G   A   G   A   G
```


- Each entry is the location within the sequence at which the k-tuple occurs

```
k-tuple     Positions at which k-tuple occurs
AGAG        s: 1,3      t: 2,4
GAGA        s: 2        t: 3
AAGA                    t: 1
```

NOTE: You can pre-compute such a table for a database and use it repeatedly on separate occasions, or you can perform a linear scan of the text database with each use.

- What are the good "offsets" for s and t?
  Keep track of an OFFSET vector, which ranges from
  $$1 - |t| \quad \text{to} \quad |s| - 1$$
  Initially, the vector $= 0$
- Now, find "hot spots": For each table entry, look at one from s (position i) and one from t (at position j), and increment i-j in our offset table.

```
AGAG:       1 - 2 = -1
            1 - 4 = -3
            3 - 2 =  1
            3 - 4 = -1
GAGA:       2 - 3 = -1
```

So now our offset table looks like:

-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	(increment size)
			0		3		1					# of occurrences

- Can compute the offset vector by looking at just k-tuples occuring in the query sequences; do not have to look at all k-tuples
- Back to our example, the largest offset is –1, which indicates the best (gapless) shift of t with respect to s is –1.

Our example again:

```
s:  - A G A G A G
t:  A A G A G A G
```

Note the relationship between the offsets and diagonals:

-1th diagonal
0th diagonal
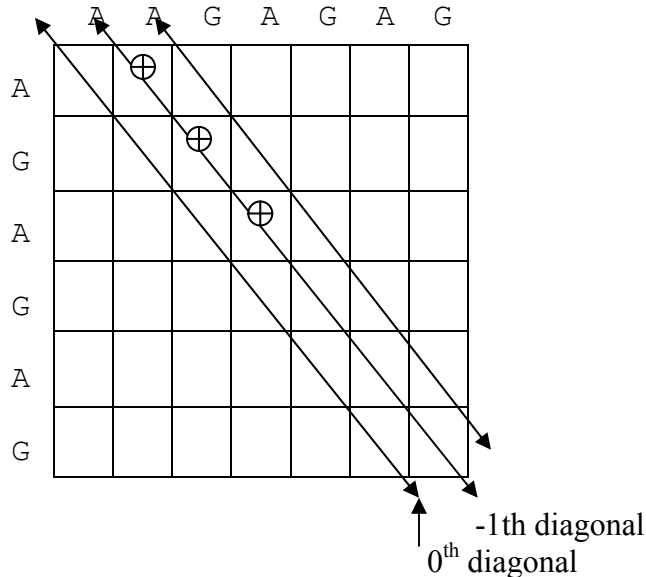
0th diagonal: s & t line up from the start
```
s AGAGAG-
t AAGAGAG
```

-1 diagonal:
```
s -AGAGAG
t AAGAGAG
```

-2 diagonal:
```
s --AGAGAG
t AAGAGAG-
```
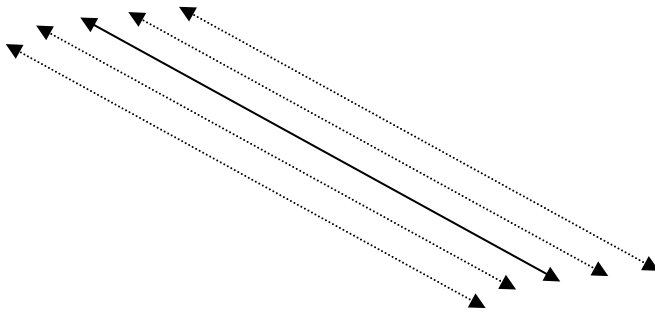
1st diagonal:
```
s AGAGAG--
t -AAGAGAG
```

So finding a good offset requires finding a good diagonal.
A <u>dense</u> diagonal has many *k*-tuple hits

<u>Actual method used</u>:
1) FASTA method picks out best diagonal runs (score is based on number of "hotspots" and number of spaces between hotspots) and combines them into regions.
2) Pick best-scoring (say) 10 regions and use a substitution matrix (PAM) to find subalignments of maximal score
3) Allow gaps using "banding"(explained below)

- So far we have not included gaps in the analysis. To allow gaps, use banding.
- Only fill out (in our DP matrix) diagonals that are *l* away from the best alignment.

<u>Example</u>: below, the solid line is the best alignment and $l = 2$

- We will expand the dynamic programming matrix only within these diagonals. Thus we are not filling out the entire matrix, only O(ln) of it.
- Tradeoffs:
    The width of the band *l* is a trade off between optimality and speed
    k-tuple: increase $k \Rightarrow$ lower sensitivity (need exact match)
                    decrease $k \Rightarrow$ increase sensitivity; more hotspots;more time

---

BLAST

<u>The basic idea</u>:

1) Condiser *k*-tuples in the query sequence. For each k-tuple, make a list of all other k-tuples (i.e., "seed words") that have score greater than or equal to T when compared to it using a particular substitution matrix.

    <u>Example</u>:

```
      query = ARNDD            k=4, T=15, matrix=PAM-120

      There are 2 4 tuples: ARND, RNDD

      Seed words for ARND are:
                  PAM-120 score
         ARND     18
         SRND     16
         ARDD     16
           ..        ..
```

2) Search for seed words in the database – these are called"hits." There are a
   few ways to do this quickly - one is to build a deterministic finite automata
   that accepts for the appropriate words, and we search through the entire
   sequence database. Alternatively, we may do a table look up (which is pre-
   computed for database)
   - Say k = 4
   - Map each word into a number ($<20^4$)indexed by 4-tuple protein sequence
   - Then each word's entry in the table points to "hits" in the database
   - Note: only a subset of 20^4 words actually occurs. Therefore use hashing to
     build a smaller table.

3) BLAST then extends hits in both directions (building up amino acid by amino
   acid)
   - The algorithm stops if the current score is significantly less than the
     maximum score seen so far.
   - Actually, the new BLAST requires two hits to be on same diagonal, non-
     overlapping, and within some distance A of each other before hits are
     extended.

4) Now, in certain cases (high enough scoring segments), BLAST allows gapped
   extensions
   - Main heuristic: Consider only cells for which local alignment score does
     not fall too far below the best score seen so far.

Newer BLAST improvements described in: Altschul, *et al*. (1997) Nucleic Acid
Residue. **25**, 3389-3402

Local Alignment Statistics
A rigorous treatement of local alignment statistics is well beyond the scope of this class.
For details, please see Karlin & Altschul 1990. Given a particular scoring system (i.e., a
particular substitution matrix), we want to know how to evaluate whether a particular
score is significant. That is, how many hits of this score are expected purely by chance?

The theoretical work makes several assumptions, including the following:
- The amino acid residues occur independently of each other in each position.
- A particular amino acid i occurs randomly at all positions with a certain probability p_i
- The expected score for aligning two amino acids is negative; i.e.,

$$\sum_{i,j} p_i p_j s_{ij} < 0$$

Here, s_{ij} is the score of aligning amino acids i and j. Note that this is a reasonable assumption for local alignments because otherwise it would tend to be advantageous always to lengthen your alignments (because adding on additional residues would tend to increase the score.)

- At least one $s_{ij} > 0$. Note that this is reasonable because otherwise the best local alignment is empty

If m is the size of the query and n is the size of the database, then you can show that by chance the expected number of hits with score at least s is given by:

$$Kmne^{-\lambda s}$$

K and λ are (computed) parameters that are depend on the substitution matrices and the probability of each amino acid occuring.   This gives us the e-score. As a sanity check, we observe that if we double the size of the database, we double the number of expected hits; similarily, if the score s of the alignment increases, the number of expected hits drops. This derivation is based on two observations:

1. Given 2 random strings, the score of the alignment follows an extreme value distribution.   More formally, if s is the score of a gapless local alignment of 2 random strings of length m & n, then when m and n are sufficiently large, s is well approximated by an extreme value distribution. Most importantly, a normal distribution is not appropriate, and would *overestimate* significance. Roughly speaking, just as the sum of many independent random variables results in a normal distribution, the maximum of many independent random variables yields an extreme value distribution. The local alignment is roughly the maximum of many alignment scores.

2. We are comparing the query with **many** strings, and must correct for that.

   Example: Suppose that a score has a probability of 10^{-5} of arising from a single pairwise sequence comparison, but the size of the database is $500,000 = 5 \times 10^5$. Thus we would expect 5 alignments of this score from chance alone. So things that look significant for pairwise comparisons must be corrected for multiple searches.

.

Note that the random model for protein sequence is not applicable to regions of restricted or unusual composition:

e.g., `qqqqqqqqqnnnnnns…`.

These subsequences are called "low complexity regions" or "repeats." They occur frequently and sequence databases, and do not give any indication of shared homology. To get around this problem, it is necessary *mask* low complexity regions and thereby get rid of them before the search. This can be done in BLAST by using the SEG or masking option (it is set by default). If you did not mask these sequences out, then "real" hits would be overwhelmed by spurious hits.

Gapped Alignments
- The theory has not been worked out as well for a gapped local alignment, but it is thought to be analogous to the ungapped case (and experimental work verifies this)
- Nevertheless, the modern BLAST programs perform gapped alignments, and the statistics are computed again by using an extreme value distribution, which is fit using comparisons of either simulated or real unrelated sequences

BLAST in practice
- BLAST reports E-values of scores, which is the expected number of hits in the database that would achieve this score by chance.
- Typically an e-score of less than 0.05 is considered significance. However, it is very common to see e-scores much, much smaller than this!
- BLAST can filter out low complexity regions (and this is the default setting).
- Whenever possible, it is better to compare protein sequences rather than DNA.
- Since BLAST statistics depend on the size of the database being searched, if you know which genome you would like to search in, it is better to restrict your searches to this genome.

References

1. Altschul, Boguski, Gish and Wooton (1994). *Nature Genetics* **6**, 119-129.
2. Altschul, Gish, Miller, Myers and Lipman (1990). *JMB* **215**, 403-440.
3. **Altschul, Madden, Schaffer, Zhang, Zhang, Miller and Lipman (1997). *NAR* 25(27)**, 3389-3402.
4. Karlin, Altschul (1990*). PNAS* **87**. 2264—2268.
5. Pearson & Lipman (1988). *PNAS* **85**, 2444-2448.