Chapter 2: Sequence Alignment

2.3 Searching Sequence Databases; FASTA, BLAST

Prof. Yechiam Yemini (YY)
Computer Science Department
Columbia University

The Problem

- How to search a sequence database (DB) for local alignments of a query sequence?
  - E.g., Search a promoter sequence in a DB of $10^6$ sequences

Dynamic Programming is prohibitively complex

Need techniques that are:
- Fast: focus search on likely solutions (trade speed for completeness)
- Tunable: retrieve meaningful alignments (ones with sufficiently high score)

FASTA & BLAST


COMS4761-- 2007
Reconsider DP Geometry

- Diagonal matching segments provide the basis for alignments
- Alignment may be viewed as connecting matching diagonals
  - Using mismatched diagonals or horizontal/vertical gapped segments

- Scoring is additive contributions of matching diagonal and connectors
  - Mismatched diagonals or vertical/horizontal gapped segments may reduce the score
  - It is best to focus on high scoring diagonals and use connectors with positive score

Dot Matrix Heuristics

Rule 1: Find high-scoring diagonals
- Search small diagonal segments
- Extend to max diagonal matches
- Connect diagonals to max score

Rule 2: Focus on meaningful alignments
- Filter low-scoring diagonals
Tradeoff: Time vs. Optimality

Smith-Waterman: 10 min
FASTA: 2 min
Blast: 20 sec

FASTA

Key idea (Pearson & Lipman 88):
- Find short diagonals by indexing the DB
- Extend these to high scoring diagonals
- Use DP to connect them

A 4 steps process

(a) Find runs of identities
(b) Maximize using FAST matrix keep top scoring segments
(c) Apply ‘scaling threshold’ to estimate segments that are within the span of the alignment that includes highest scoring segment
(d) Use dynamic programming to optimize alignment to a reasonable level that demonstrates the top scoring segments.
Step (a): Find Diagonal Matches by Indexing

- Key idea: create an index of $k$-tuples of the DB
  - Scan database to index $k$-tuples \([k=1..5]\)
  - Scan query to index $k$-tuples
  - Find all diagonal matches of length $k$ by comparing the hash tables
  - Merge these short diagonals into maximal diagonal matches

Example:

Database \(d\): TATCGATCGA

Position: 1 2 3 4 5 6 7 8 9 10

Query \(q\): GATCG

Position: 1 2 3 4 5

1. Extract index

2. Find matches

3. Merge diagonal matches

FASTA Steps (b-d): Optimize Score

b) Filter low-score diagonals

c) Extend diagonals to max score; keep high-scoring segments

d) Use DP for a narrow band around the high scoring segments
BLAST: Basic Local Alignment Search

- Altschul & Karlin [1990]: a family of algorithms
- Idea: find matches with significant score statistics
  - Find maximal segment pairs (MSP): segments with significant score
  - Based on extensive statistical theory (summarized soon)

Base Algorithm:
- Step 1: index DB for words of size W (W-mers);
  index query sequence for W-mers with score >Threshold
  - W=3 for protein, 11 for nucleotides
- Step 2: search for matches with high score (HSP=high scoring pairs)
- Step 3: extend hits to maximal score segments
- Step 4: report matches with score above S
**BLAST Step 1-3: Finding Short High-Scoring Pairs (HSP)**

- Create an Index of W-mers for database & query
  - For proteins W=3 means a dictionary of \(20^3=8000\) words

- Match W-mers that score above a threshold T
  - FASTA searches for exact matches of \(k\)tuples
  - BLAST, in contrast, searches for high scoring pairs (HSP)
  - Key idea: exploit the fast part of the search to max the score rather than push the maximization for later, slower, phases

  From A. Baxevanis, "Nucleotide and Protein Sequence Analysis I" via Kellis & Indyk, MIT, "BLAST & Database Search, Lecture 2"

**Blast Steps 3-4: Extending Short HSPs**

- The short HSPs are extended to increase the score

- Report above threshold HSPs and their scores

---

**Query:** GSVDDTITGQLAAQLNKRCKIGQGLVQNRWQKQPLMDKRNKIEELNLVLVEAFAVFAELRTQLEDE

**Neighborhood words**

- PQG 18
- PEG 15
- PEG 14
- PNG 13
- PQL 13
- PMG 13
- PGN 13
- PQO 12
- PGN 12
- etc...

**Score**

Max score extension

**Subject:** 325 SLAALNKCKIGQGLVQNRWQKQPLMDKRNKIEELNLVLVEA 365

**Subject:** 299 TLSVLDCTVRPGE6RMKRWLPVMRDELRQLEQGITA 330
Statistics Background

- How do we distinguish “meaningful” alignment from a random one?
  - E.g., suppose an alignment of a query q with a sequence d scores s, is it “meaningful”?
  - If s is much higher than the average score of a random alignment, the answer is positive

- Key idea: use statistics of alignment scores to distinguish “meaningful”

- Basic probability:
  - Given a sample space (e.g., possible alignments)
  - A random variable is a real-valued function of samples (e.g., alignment score)
  - The statistics of a random variable is described by a distribution function:
    \[ F(x) = \Pr[S \leq x] \]
    with a density function \( f(x) = \frac{dF}{dx} \), \( f(x) \Delta x \sim \Pr[x-\Delta x \leq S \leq x + \Delta x] \)

- Standard deviation, \( \sigma = \sqrt{E[S^2] - E[S]^2} \)
- Mean, \( \mu = E[S] = \int_{-\infty}^{\infty} f(x)dx \)
Distinguishing Meaningful From Random

- "meaningful" ~ score is at the tail of the distribution
- Z-Score: \( Z(s) = \frac{(s-\mu)}{\sigma} \)  
  \( [z(s) \geq 7 \Rightarrow d \text{ is meaningful}] \)
- P-Score: \( p = P(s > s) = 1 - F(s) \)  
  \( [p \leq 0.02 \Rightarrow d \text{ is meaningful}] \)

How Significant is An Alignment?

- Key idea: Consider the highest matching score \( S \) as a random variable and use Z-score to determine whether an alignment is meaningful

Strategic:
- Define the highest matching score \( S \) as a random variable
  - Let \( q \) be a query and \( d \) a sequence from a database \( D \)
  - Define \( S(q,d) = \max \{ s(q,q') \} \) where \( s(q,q') \) is the score of an alignment of \( q \) and a subsequence \( q' \) of \( d \)
  - \( S \) is a random variable defined over the space of local alignments \( \{(q,d)\} \)
- Suppose \( S \) has mean \( \mu \) and standard deviation \( \sigma \)
- Use Z-score, \( Z(s) = \frac{(s-\mu)}{\sigma} \) to determine significance of a local alignment scoring \( s \)
- For protein sequences \( Z(s) > 7 \) is considered significant

Key challenge: how do we determine the distribution of \( S \) ?

What Is The Distribution of $\mathbf{S}$?

- Consider the scores $S'(q,q')$ for $N$ random sequences $q'$
  - The score $S'(q,q')=\sum s(q_i, q'_i)$ is the sum of independent random variables
  - For sufficiently long sequences $S'(q,q')$ is normally distributed

$$f(x; \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(x-\mu)^2}{2\sigma^2} \right).$$

- BLAST maximizes the score: $S(q,d)=\max \{ S'(q,q') : q' \text{ is sub of } d \}$
  - $S$ has extreme order distribution: $F(x)=P[S(x)]$ for all $q'$ sub of $d$
  - Extreme order theory: $F(x)$ is exponentially distributed

- Karlin-Altschul Statistics: $F(x)=P[S(q,d)<x] \sim 1-\exp[-(Kmn)\exp(-\lambda x)]$
  - Here $m=|q|, n=|d|$, $\lambda$ and $K$ may be computed from the scoring statistics
  - A good approximation for aligning sequences of length $m,n$ is: $\lambda = -\log(mn), \sigma=1$.

Application Example

- Consider local alignment of protein sequences $(q,d)$
  - Suppose the best local alignment is:
    $$X=FWLEVEGNSMTAPTG \quad Y=FWLDVQGDSMTAPAG$$

- Compute the score of this local alignment:
  - Using PAM250, the score is $s=73$; normalized to log2 scale this gives $s'=\log(s)=24.3$

- Compute the parameters of a random scoring normal distribution:
  - Suppose $|q|=|d|=256$
    - The normal distribution for random scoring: $\mu=\log(mn)=\log(256^2)=16$, $\sigma=1$;

- Compute
  - $Z(s')= (24.3-16)/1= 8.3$ bits $> 7 \Rightarrow$ alignment is significant
Conclusions

- Indexing the sequence DB can accelerate alignment
  - FASTA: accelerate search for diagonal matches, then optimize alignment
  - BLAST: accelerate both, search for matches and optimizing the alignment

- FASTA innovations:
  - Exploit the geometry of diagonals
  - Grow alignments from seeds

- BLAST innovation: recast alignment as a pure search
  - Use scoring statistics as a mere guideline for the search
  - Contrast with DP use of (inexact) scoring to solve an exact optimization problem