Master Course
Sequence Alignment

Lecture 7

Database searching (1)
Sequence searching - challenges

- Exponential growth of databases
Bioinformatics justification

- “Mind the Gap”
- There are far more sequence data than structural/functional data
- We need to fill this gap by analysis and prediction pipelines
Sequence searching - definition

• Task:
  • Query: short, new sequence (~1000b)
  • Database (searching space): very many sequences
  • Goal: find seqs related to query

• We want:
  • fast tool
  • primarily a filter: most sequences will be unrelated to the query
  • fine-tune the alignment later
Heuristic Alignment Motivation

• the dynamic programming algorithm has complexity $O(mn)$, which is too slow for large databases with high query traffic
• heuristic methods do fast approximation to dynamic programming
  – FASTA [Pearson & Lipman, 1988]
  – BLAST [Altschul et al., 1990]
Heuristic Alignment Motivation

• consider the task of searching SWISS-PROT against a query sequence:
  • say our query sequence is 362 amino-acids long
  • SWISS-PROT release 38 contains 29,085,265 amino acids
• finding local alignments via dynamic programming would entail $O(10^{10})$ matrix operations
• many servers handle thousands of such queries a day (NCBI > 50,000)
• Using the DP algorithm for this is clearly prohibitive
• Note: each database search can be sped up by “trivial parallelisation”
Heuristic Alignment

• Today: BLAST is discussed to show you a few of the tricks people have come up with to make alignment and database searching fast, while not losing too much quality.
What is BLAST

• **Basic Local Alignment Search Tool**
• **Bad news:** it is only a heuristic
  • Heuristics: A rule of thumb that often helps in solving a certain class of problems, but makes no guarantees.
    
    *Perkins, DN (1981) The Mind's Best Work*
  • Also see [http://en.wikipedia.org/wiki/Heuristic](http://en.wikipedia.org/wiki/Heuristic)
• **Basic idea:**
  • Discard putatively unrelated sequences fast
  • High scoring segments have well conserved (almost identical) part
  • As well conserved parts are identified, extend these to the *real* alignment

[8] 09-01-2007 Sequence Analysis
What means *well conserved* for BLAST?

- BLAST works with *k-words* (words of length k)
  - *k* is a parameter
  - different for DNA (>10) and proteins (2..4), default k values are 11 and 3, resp.
- word $w_1$ is *T-similar* to $w_2$ if the sum of pair scores is at least $T$ (e.g. $T=12$)

<table>
<thead>
<tr>
<th>Similar 3-words</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$: R K P</td>
</tr>
<tr>
<td>$w_2$: R R P</td>
</tr>
<tr>
<td>Score: 9 -1 7</td>
</tr>
</tbody>
</table>
BLAST algorithm
3 basic steps

1) Preprocess the query sequence: extract all the \textit{k-words}

2) Scan for \textit{T-similar} matches in database

3) Extend them to alignments
BLAST, Step 1: Preprocess the query

- Take the query (e.g. LVNRKPVVP)
- Chop it into overlapping k-words (k=3 in this case)

<table>
<thead>
<tr>
<th>Query:</th>
<th>LVNRKPVVP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word1:</td>
<td>LVN</td>
</tr>
<tr>
<td>Word2:</td>
<td>VNR</td>
</tr>
<tr>
<td>Word3:</td>
<td>NRK</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

- For each word find all similar words (scoring at least T)
- E.g. for RKP the following 3-words are similar:
  QKP  KKP  RQP  REP  RRP  RKP
Step 2: Scanning the Database with DFA (Deterministic Finite-state Automaton)

- search database for all occurrences of query words
- can be a massive task
- approach:
  - build a DFA (deterministic finite-state automaton) that recognizes all query words
  - run DB sequences through DFA
  - remember hits
DFA
Finite state machine

- abstract machine
- constant amount of memory (states)
- used in computation and languages
- recognizes regular expressions
  - cp dmt*.pdf /home/john
BLAST, Step 2: Find “exact” matches with scanning

- Use all the $T$-similar $k$-words to build the Finite State Machine
- Scan for exact matches

...VLQKPKKPRQPREPRRRPRKP...

...QKP KLKPVLRKRQPCCEVVRKPLVKVIRCLA...
Scanning the Database - DFA

1) Preprocess
2) Scan
3) Extend

Moore paradigm: the alphabet is (a, b), the states are q0, q1, and q2, the start state is q0 (denoted by the arrow coming from nowhere), the only accepting state is q2 (denoted by the double ring around the state), and the transitions are the arrows. The machine works as follows. Given an input string, we start at the start state, and read in each character one at a time, jumping from state to state as directed by the transitions. When we run out of input, we check to see if we are in an accept state. If we are, then we accept. If not, we reject.

Moore paradigm: accept/reject states

Mealy paradigm: accept/reject transitions

Example (next 2 slides):
- consider a DFA to recognize the query words: QL, QM, ZL
- All that a DFA does is read strings, and output "accept" or "reject."
- use Mealy paradigm (accept on transitions) to save space and time
a DFA to recognize the query words: QL, QM, ZL in a fast way

Mealy paradigm

Accept on red transitions

Go to start at each new query word

not (L or Z)

not (L or M or Q)

Q

L or M

not (L or M or Q)

Q

L or M

not (L or Z)

not (L or Z)

start

[16] 09-01-2007 Sequence Analysis
BLAST, Step 3: Extending “exact” matches

- Having the list of matches (hits) we extend alignment in both directions

Query: L V N R K P V
Subject: G V C R R P L K
Score: -3 4 -3 5 2 7 1 -2 -3

- ...till the sum of scores drops below some level X from the best known
Step 3: Extending Hits

- extend hits in both directions (without allowing gaps)
- terminate extension in one direction when score falls certain distance below best score for shorter extensions

\[ \text{current extension } c \geq \text{best extension } b - \epsilon \]

- return segment pairs scoring at least $S$
More Recent BLAST Extensions

- the two-hit method
- gapped BLAST
- hashing the database
- PSI-BLAST

all are aimed at increasing sensitivity while keeping run-times minimal

Altschul et al., *Nucleic Acids Research* 1997
The Two-Hit Method

- extension step typically accounts for 90% of BLAST’s execution time
- key idea: do extension only when there are two non-overlapping hits on the same diagonal within distance $A$ of each other
- to maintain sensitivity, lower $T$ parameter
  - more single hits found
  - but only small fraction have associated 2nd hit

[20] 09-01-2007 Sequence Analysis
The Two-Hit Method

Figure from: Altschul et al. Nucleic Acids Research 25, 1997
Gapped BLAST

- trigger gapped alignment if two-hit extension has a sufficiently high score
- find length-11 segment with highest score; use central pair in this segment as seed
- run DP process both forward & backward from seed
- prune cells when local alignment score falls a certain distance below best score yet
Gapped BLAST

Figure from: Altschul et al. Nucleic Acids Research 25, 1997
Combining the two-hit method and Gapped BLAST

- **Before:**
  - relatively high $T$ threshold for 3-letter word (hashed) lists
  - two-way hit extension (see earlier slides)
- **Current BLAST:**
  - Lower $T$: many more hits (more 3-letter words accepted as match)
  - Relatively few hits (diagonal elements) will be on same matrix diagonal within a given distance $A$
  - Perform 2-way local Dynamic Programming (gapped BLAST) only on ‘two-hits’ (preceding bullet)

*The new way is a bit faster on average and gives better (gapped) alignments and better alignment scores!*

[24] 09-01-2007 Sequence Analysis
Making things even faster-
indexing the complete database
(or genome sequence)

• SSAHA – Sequence Search and Alignment by Hashing Algorithms (Ning et al., 2001)
• BLAT – BLAST-like Alignment Tool (Kent, 2002)
• PatternHunter (Ma et al., 2002)
• BLASTZ – alignment of genomic sequences (Schwartz et al., 2003)
BLAT - BLAST-Like Alignment Tool

- Analyzing vertebrate genomes requires rapid mRNA/DNA and cross-species protein alignments. BLAT (the BLAST-like alignment tool) was developed by Jim Kent from UCSC. It is more accurate and 500 times faster than popular existing tools such as BLAST for mRNA/DNA alignments and 50 times faster for protein alignments at sensitivity settings typically used when comparing vertebrate sequences (e.g. BLAST).

- BLAT's speed stems from an index of all nonoverlapping k-mers in the genome. This index fits inside the RAM of inexpensive computers, and need only be computed once for each genome assembly. BLAT has several major stages. It uses the index to find regions in the genome likely to be homologous to the query sequence. It performs an alignment between homologous regions. It stitches together these aligned regions (often exons) into larger alignments (typically genes). Finally, BLAT revisits small internal exons possibly missed at the first stage and adjusts large gap boundaries that have canonical splice sites where feasible.

- From Wikipedia, the free encyclopedia
Hashing - associative arrays

• **Indexing** with the object, the

• Hash function:

  ![Hashing diagram](image)

  - set of possible objects - **large**
  - small (fits in memory)

• **Objects should be** “well spread”
Hashing - examples

- **T9 Predictive Text** in mobile phones
  - “hello”: 4, 4, 3, 3, 5, 5, 5, (pause) 5, 5, 5, 6, 6, 6
  - “hello” in T9: 4, 3, 5, 5, 6
  - **Collisions**: 4, 6:
    - “in”, “go”
Hashing - examples (cont..)

• Other easier hash function: let $a=1$, $b=2$, $c=3$, etc.
  • “hello” now gets hash address $8+5+12+12+15 = 52$
  • “olleh” will get same address (collision)
  • Each word encountered gets a hash address immediately and can be indexed.
• How good is this hash function?
Indexing the database: Find ”exact” matches with hashing

- Preprocess the database
  - Hash the database with k-words
  - For each k-word store in which sequences it appears

Hashed DB:
- QKP: HUgn0151194, Gene14, IG0, ...
- KKP: haemoglobin, Gene134, IG_30, ...
- RQP: HSPHOSR1, GeneA22...
- RKP: galactosyltransferase, IG_1...
- REP: haemoglobin, Gene134, IG_30, ...
- RRP: Z17368, Creatine kinase, ...
- ...
Indexing the database: Find “exact” matches with hashing

- The database is **preprocessed only once**! (independent from the query)
- In a constant time we can get the sequences with a certain *k-word*

**Hashed DB:**
- **QKP**: HUgn0151194, Gene14, IG0, ...
- **KRP**: haemoglobin, Gene134, IG_30, ...
- **RQP**: HSPHOSR1, GeneA22...
- **RKP**: galactosyltransferase, IG_1...
- **REP**: haemoglobin, Gene134, IG_30, ...
- **RRP**: Z17368, Creatine kinase, ...
- **...**
BLAST flavours

- **blastp**: protein query, protein db
- **blastn**: DNA query, DNA db
- **blastx**: DNA query, protein db
  - in all reading frames. Used to find potential translation products of an unknown nucleotide sequence.
- **tblastn**: protein query, DNA db
  - database dynamically translated in all reading frames.
- **tblastx**: DNA query, DNA db
  - all translations of query against all translations of db (compare at protein level)

[32] 09-01-2007 Sequence Analysis