In Previous Lecture

BLAST & Database Search

Manolis

Piotr Indyk

BLAST and Database Search

Setup

The BLAST algorithm
BLAST extensions
Substitutions matrices
Why K-mers work
Applications

How to answer the query

• We could just scan the whole database
• But:
  – Query must be very fast
  – Most sequences will be completely unrelated to query
  – Individual alignment needs not be perfect. Can fine-tune
• Exploit nature of the problem
  – If you’re going to reject any match with idperc < 90%, then why bother even looking at sequences which don’t have a fairly long stretch of matching a.a. in a row.
  – Pre-screen sequences for common long stretches, and reject vast majority of them

What does “similar” mean?

• Simplest idea: just count the number of common amino-acids
  – E.g., RGRKW matches RGIKW with idperc = 80%
• Not all matches are created equal - scoring matrix
• In general, insertions and deletions can also happen

Setup

• Sequences of symbols:
  – Bases: A,G,T,C
  – Amino-acids (a.a.):
    Y,V,B,Y,X
• Database search:
  – Database.
  – Query:
  – Output: sequences similar to query

AIKWQPRSTW….
IKMQRHIKW….
HDLFWHLWH….
……………………
RGIKW
W-mer indexing

- **W-mer**: a string of length W
  - **Preprocessing**: For every W-mer (e.g., W=3), list every location it occurs in the database.
  - **Query**: Generate W-mers and look them up in the database.
  - **Process the results**

- **Benefit**:
  - For W=3, roughly one W-mer in $2^3 = 8$ will match, i.e., one in a thousand.

6.046 Digression

- This “lookup” technique is quite fundamental.
- Will see more in 6.046, lecture 7, on hashing.

BLAST and Database Search

Motivation

The BLAST algorithm

BLAST extensions

Substitutions matrices

Why K-mers work

Applications

BLAST

- Specific (and very efficient) implementation of the W-mer indexing idea
  - How to generate W-mers from the query
  - How to process the matches

The BLAST Search Algorithm

query word ($W = 3$)

neighborhood words

neighborhood score threshold

(query: GUESTGOSIALMACHRICHJHIEGHIKSVDFEDALOGOIL)

(PQQ 10

PQQ 15

PQQ 14

PQQ 13

PQQ 12

Prec 11

Prec 10

Prec 5

Prec 4

Prec 3

Prec 2)

(score: 125, 425, 716, 944)

(query): 226 MAEGECYTVSGMKLRLYANSQYLAC

(score: 285, 1151, 1920, 2980)

High-scoring Segment Pair (HSP)
Extending the seeds

- Extend until the cumulative score drops

Statistical Significance

- Karlin-Altschul statistics
  - P-value: Probability that the HSP was generated as a chance alignment.
  - Score: $-\log$ of the probability
  - E: expected number of such alignments given database

BLAST and Database Search

Motivation
The BLAST algorithm
BLAST extensions
Substitutions matrices
Why K-mers work
Applications

Extensions: Filtering

- Low complexity regions can cause spurious hits
  - Filter out low complexity in your query
  - Filter most over-represented items in your database

Extensions: Two-hit blast

- Improves sensitivity for any speed
  - Two smaller W-mers are more likely than one longer one
  - Therefore it’s a more sensitive searching method to look for two hits instead of one, with the same speed.
- Improves speed for any sensitivity
  - No need to extend a lot of the W-mers, when isolated

Extensions: beyond W-mers

- W-mers (without neighborhoods):
  - RGKW $\rightarrow$ RGI, GIK, IKW
- No reason to use only consecutive symbols
- Instead, we could use combs, e.g.,
  - RGKW $\rightarrow$ R*I^K*, RG**W, ...
- Indexing same as for W-mers:
  - For each comb, store the list of positions in the database where it occurs
  - Perform lookups to answer the query
- Randomized projection:\footnote{Buhler’01, based on Indyk-Motwani’98}
  - Choose the positions of * at random
  - Example of a randomized algorithm
**Amino Acids**

<table>
<thead>
<tr>
<th>Codon</th>
<th>Amino Acid</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTC</td>
<td>Leu</td>
</tr>
<tr>
<td>GCC</td>
<td>Pro</td>
</tr>
<tr>
<td>GAC</td>
<td>His</td>
</tr>
<tr>
<td>GGC</td>
<td>Arg</td>
</tr>
<tr>
<td>GTA</td>
<td>Ala</td>
</tr>
<tr>
<td>GCA</td>
<td>Arg</td>
</tr>
<tr>
<td>GAA</td>
<td>Lys</td>
</tr>
<tr>
<td>GGA</td>
<td>Arg</td>
</tr>
<tr>
<td>CTA</td>
<td>Leu</td>
</tr>
<tr>
<td>CCA</td>
<td>Pro</td>
</tr>
<tr>
<td>CAA</td>
<td>Lys</td>
</tr>
<tr>
<td>CGA</td>
<td>Arg</td>
</tr>
<tr>
<td>TTT</td>
<td>Phe</td>
</tr>
<tr>
<td>TTG</td>
<td>Leu</td>
</tr>
<tr>
<td>TCG</td>
<td>Ser</td>
</tr>
<tr>
<td>TAG</td>
<td>Stop</td>
</tr>
<tr>
<td>TGG</td>
<td>Trp</td>
</tr>
<tr>
<td>CTT</td>
<td>Leu</td>
</tr>
<tr>
<td>ATA</td>
<td>Ile</td>
</tr>
<tr>
<td>ACA</td>
<td>Thr</td>
</tr>
<tr>
<td>AAA</td>
<td>Lys</td>
</tr>
<tr>
<td>GTT</td>
<td>Val</td>
</tr>
</tbody>
</table>

**Substitution Matrices**

- Not all amino acids are created equal
  - Some are more easily substituted than others
  - Some mutations occur more often
  - Some substitutions are kept more often
- Mutations tend to favor some substitutions
  - Some amino acids have similar codons
  - They are more likely to be changed from DNA mutation
- Selection tends to favor some substitutions
  - Some amino acids have similar properties / structure
  - They are more likely to be kept when randomly changed
- The two forces together yield substitution matrices

**BLOSUM matrices**

- BioSum = BLOck SUBstitution matrices

- PAM matrices

- PAM = Point Accepted mutation

**Computing Substitution Matrices**

- Take a list of 1000 aligned proteins
  - Every time you see a substitution between two amino acids, increment the similarity score between them.
  - Must normalize it by how often amino acids occur in general.
  - Rare amino acids will give rare substitutions.

- BLOSUM matrices vs. PAM

  - BLOSUM were built only from the most conserved domains of the blocks database of conserved proteins.
  - BLOSUM: more tolerant of hydrophobic changes and of cysteine and tryptophan mismatches
  - PAM: more tolerant of substitutions to or from hydrophilic amino acids.

---

**BLAST and Database Search**

- Motivation
- The BLAST algorithm
- BLAST extensions
- Substitutions matrices
- Why K-mers work
- Applications
BLAST and Database Search

Motivation
The BLAST algorithm
BLAST extensions
Substitutions matrices
Why does this work
Applications

Overview: Why this works

- In worst case:
  - W-mer: W=3
  - Combs/random projection
- In average case
- Simulations
- Biological case: counting W-mers in real alignments
  - Long conserved W-mers do happen in actual alignments
  - There’s something biological about long W-mers

Pigeonhole principle

- Pigeonhole principle
  - If you have 2 pigeons and 3 holes, there must be at least one hole with no pigeon

Pigeonhole and W-mers

- Pigeonholing mis-matches
  - Two sequences, each 9 amino-acids, with 7 identities
  - There is a stretch of 3 amino-acids perfectly conserved

In general:
Sequence length: n
Identities: t
Can use W-mers for W= \([n/(n-t+1)]\)

Combs and Random Projections

- Assume we select k positions, which do not contain *, at random with replacement
- What is the probability we miss a sequence match?
  - At most: 1-idperc
  - In our case: 1-(7/9)^4 ≈ 0.63...
- What if we repeat the process l times, independently?
  - Miss prob. = 0.63^l
  - For l=5, it is less than 10%

True alignments: Looking for K-mers

- 850Kb region of human, and mouse 450Kb ortholog.
- Blasted every piece of mouse against human (6,50)
- Identify slope of best fit line

Personal experiment run in 2000.
Conclusions

- Table lookup – very powerful technique
- Deterministic, randomized
- More (on hashing) in 6.046