Lecture 7
Database Search

• Quadratic complexity not suitable for searching large databases
  – e.g. need to compare a query sequence to all sequences in a large database.
  – Alternative: Heuristics
    • BLAST
    • FAST

• Simple scoring scheme such as (+1, -1, -2) is not suitable for comparing protein sequences.
  – e.g. amino acids of similar size are more likely to get substituted for one another.
  – Alternative: Substitution matrix, $S(a,b) = \text{score for aligning } a \text{ with } b$
    • General approach for substitution matrices
    • PAM
    • BLOSUM
BLAST
(Basic Local Alignment Search Tool)

• BLAST returns a list of high scoring segment pairs between the query sequence and sequences in the database.

• A segment is a substring of a sequence.

• A segment pair is a pair of segments of the same length \( \rightarrow \) can from a gapless alignment.

• Basic BLAST is ungapped.

• Given a query sequence, BLAST returns all segment pairs between the query and a database sequence with score above a threshold \( S \).

• \( S \) can be set by the user.
HOW does BLAST work?

- It finds certain “seeds” which are very short segment pairs between the query and the database sequence.

- These seeds are then extended in both directions without gaps, until the maximum possible score for extensions is reached.

- Time reduction: the extension stops when the score falls below a carefully computed limit $X$. 
BLAST Algorithm

• For a given query sequence, compile a list of short high scoring strings (words in BLAST jargon)

• Search for hits – each hit gives a “seed”

• Extend “seeds”

• Return segments pairs with score > S.
**k-mers**

- How is the list of short high scoring strings obtained?

- **k-mers**: substrings of length \( k \).
  - DNA sequence: all \( k \)-mers.
  
  - Protein sequence: all \( k \)-mers in addition to neighboring \( k \)-mers. A neighboring \( k \)-mer is a \( k \) length string that scores high with some \( k \)-mer of the sequence.

- **Typical \( k \): 3 or 4**
Database

• The database is hashed and indexed by all words of size $k$.

• Each word will point to the locations where it exists in the database.

• We have only $4^k$ words in case of DNA sequences and $20^k$ words in case of proteins.

• This is much less than the number of sequences stored in the database.
Overview

$k = 3$

high scoring neighbors of PQG

DataBase

seed

GSVEDTTGSQSLAALLNKCKTPOQQRVLVIRQWKQPLOMDKNGIEERLNLVEAFVEDAELRQLQEDL

Saad Mneimneh
http://engr.smu.edu/~saad/
BLAST algorithm

- Split query into overlapping words of length $k$ (k-mers).

- For each word, find neighboring words that score at least $T$.

- Look into database where these words occur: seeds

- Extend each seed until score drops below $X$.

- If it scores $> S$, return segment pair.
Generating neighbors

- For every amino acid in the word, try all possibilities
- Score the words
- Keep those with within threshold
Looking in database

- Each neighboring word gives a list of locations where it’s found
- Follow pointers to obtain seeds
Extending seeds

- Extend seed until score drops below $X$.
- Return highest scoring segment pair.
Why $k$-mers make sense?

- If two sequences have some level of similarity (say $L\%$), they must contain a preserved $k$-mer for some $k$.

- Why?

- smurfhole principle!
Example smurfhole

• If we have 91 smurfs and 10 holes, there must be at least one hole with at least 10 smurfs.

• Proof: if non of the holes contain 10 smurfs, we have at most 9 x 10 = 90 smurfs!
Application to \( k \)-mers

- Two sequences of length 100 with > 90% similarity.

- There must be a preserved 10-mer.

Where will the 91st go?
Random model

• In the previous model, we cannot guarantee a $k$-mer for $k > 10$.

• What happens if we distribute the 91 similarities randomly?

• We get even better chance of having $k$-mers for other $k$s.
Running time

- $n$: length of query sequence
- $s$: number of seeds
- $L$: length of alignment

- Running time $= O(n + Ls)$

- For one sequence in the database,
  $s = O(n), L = O(n) \Rightarrow O(n^2)$

  But in practice faster than Smith-Waterman.
Variations

• 2-hit BLAST
  – Require two seeds that are within 40 amino acids of each other to start considering a database sequence.
  – Reduce the space of potential hits, speeding up the algorithm.

• Gapped BLAST
  – BLAST with gaps, find a seed, then find more seeds and extend them, then join segments with gaps in a band around the main seed.
FAST

- Record all occurrences of windows of certain size $k$ in the two sequences $x$ and $y$ (1-2 for DNA, 3-4 for proteins).

- If a window occurs at $x_i$ and at $y_j$, we say it occurs at an offset $i - j$.

- Offset range is 1 – $n$ to $m$ – 1.
Example

- Window of size 2

- \( x = \text{AGAGAG} \)
- \( y = \text{AAGAGAG} \)

- The window AG occurs at \( x_1 \) and \( y_4 \), so it occurs at offset \( 1 - 4 = -3 \). It also occurs at other offsets.

- What does it mean? Aligning \( x \) and \( y \) at offset -3 aligns the window AG.

  \[
  \begin{array}{c}
  \text{AGAGAG} \\
  \text{AAGAGAG}
  \end{array}
  \]

- What is the offset that maximizes the number of aligned windows?
FAST algorithm

• Need
  – lookup table: contains all possible windows of size $k$, e.g. $4^k$ and their occurrence in $x$ and $y$.
  – Offset vector: for each offsets, holds how many times that offset occurred.

• Fill the lookup table

• Compute the offset vector

• Choose the most frequent offset

• Align $x$ and $y$ at that offset
Example

- $x = \text{AGAGAG}$
- $y = \text{AAGAGAG}$

<table>
<thead>
<tr>
<th></th>
<th>$x: 1, 3, 5$</th>
<th>$y: 2, 4, 6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td></td>
<td>$y: 1$</td>
</tr>
<tr>
<td>AG</td>
<td>$x: 2, 4$</td>
<td>$y: 3, 5$</td>
</tr>
<tr>
<td>GA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AGAGAG
AAGAGAG
Variation

Run a bounded dynamic programming in a band centered at the offset diagonal.

AAGAGAGAG

basic FAST alignment obtained
Substitution matrices

Need mainly two things:

– For every pair \(a, b\): \(p_{ab}\), the probability of observing \(a\) aligned with \(b\). \(p_{ab} = p_{ba}\)

– For every \(a\): \(p_a\), the probability of observing an \(a\).
Aligned sequences
Related / Unrelated

• Let $M$ be the model in which $x$ and $y$ are related and obtained according to the joint probabilities $p_{ab}$.

• Let $R$ be the model in which $x$ and $y$ are unrelated and obtained independently at random according to the individual probabilities $p_a$. 
The score of a $x$ and $y$ is the relative likelihood that the sequences are related compared to being unrelated: *odds ratio*

\[
score(x, y) = \frac{p(x, y | M)}{p(x, y | R)} = \prod_i \frac{p_{x_i y_i}}{p_{x_i} p_{y_i}}
\]
Intuition

\[ x = \ldots \ldots a \ldots \ldots \]
\[ y = \ldots \ldots b \ldots \ldots \]

Let \( p[a \rightarrow b] = \) probability that \( a \) mutates into \( b \) (\( \neq p[b \rightarrow a] \))

Taking the point of view of \( a \), the probability that \( b \) is there is \( p[a \rightarrow b] \).

But there is a chance of \( p_b \) for a random occurrence of \( b \).

This ratio is: \[ \frac{p[a \rightarrow b]}{p_b} \]

But \( p_{ab} = p_a \cdot p[a \rightarrow b] \), therefore we get \[ \frac{p_{ab}}{p_a p_b} \]
Additive score

• The score is multiplicative
  \[ \prod_i \frac{p_{x_i y_i}}{p_x p_y} \]

• To make it additive, take the log
  \[ \log \prod_i \frac{p_{x_i y_i}}{p_x p_y} = \sum_i \log \frac{p_{x_i y_i}}{p_x p_y} \]

• Substitution matrix \( S \) where \( S(a,b) = \log \frac{p_{ab}}{p_a p_b} \)
PAM matrices

• Stands for *Point Accepted Mutations*.

• An accepted mutation is a mutation that was positively selected by the environment and did not cause the death of the organism.

• Given a PAM matrix $M$, $M_{ab} = p[a \rightarrow b]$ in a certain *evolutionary time period*.
Unit of Evolution

• It is difficult to capture from statistical data the relation of proteins that are evolutionary very far apart. If \( a \to b \), we don’t capture the intermediate mutations.

• Define 1 unit of evolution as the amount of evolution that will change 1 in 100 amino acids on average.

• Compute the 1-PAM matrix corresponding to 1 unit of evolution from short time interval statistical data.

• Obtain other \( k \)-PAM matrices from the first one.
1-PAM matrix

• Compute a matrix \( M \), \( M_{ab} = \rho[a \rightarrow b] \) for all \( a, b \).

• Scale \( M \) such that the expected number of mutations \( \Sigma_a \rho_a (1 - M_{aa}) \) is 0.01 (1%).
  [this is same as the probability of a mutation]

• Compute \( \rho_a \) for every \( a \).

• Then use
  \[
  S(a, b) = 10 \log_{10} \left( \frac{M_{ab}}{\rho_b} \right)
  \]
  to obtained an additive score.
1-PAM Computation

• Let $f_{ab}$ = number of times a is aligned with b (both direction).

• Let $f_a = \sum_b f_{ab}$ (number of a’s)

• Let $f = \sum_a f_a$ (all characters)

• Estimate $p_{ab}$ as $f_{ab}/f$

• Estimate $p_a$ as $f_a/f$

• Then $M_{ab} = p[a \rightarrow b] = p_{ab}/p_a$

• Note $\sum_b M_{ab} = 1$
Computation (cont.)

- $M_{ab} = \alpha M_{ab}$ (if $a \neq b$)
- $M_{aa} = \alpha M_{aa} + 1 - \alpha$

Note, we still have $\Sigma_b M_{ab} = 1$.

- $\Sigma_a \rho_a (1 - M'_{aa}) = \alpha \Sigma_a \rho_a (1 - M_{aa})$
2-PAM matrix

• $p_2[a \rightarrow b]$ in two units of evolution will be the probability of $a$ mutating to some character $c$ in one unit of evolution and $c$ mutating to $b$ in another unit of evolution.

• $p_2[a \rightarrow b] = \sum_c p[a \rightarrow c].p[c \rightarrow b] = \sum_c M_{ac}.M_{cb}$

• 2-PAM matrix $= M^2$
k-PAM matrix

- $k$-PAM = $M^k$

- $S_k(a,b) = 10 \log_{10} \left( \frac{M_{ab}^k}{p_b} \right)$
BLOSUM matrices (BLOCKS substitution matrices)

- BLOSUM matrices are derived from a database of BLOCKS (the BLOCKS database) where each block is a multiple ungapped alignment of related protein sequences.

- The goal is to obtain a scoring for protein sequences that are evolutionary far apart. How far?

- The sequences from each block are clustered, putting two sequences in the same cluster if they have more than $L\%$ similarity (percentage of aligned matching residues).

- Distant sequences $\Rightarrow$ occur in different clusters

Saad Mneimneh
http:// engr.smu.edu/~saad/
BLOSUM computation

- Count number of mutations between distant sequences only, i.e. less than $L\%$ similar.

- $a$ and $b$ aligned but end up in different clusters.

- Increment $f_{ab}$ by $1/n_1n_2$ every time this happens.
Computation (cont.)

• Estimate $p_a$ as $\frac{\sum_b f_{ab}}{\sum_{c,d} f_{cd}}$

• Estimate $p_{ab}$ as $\frac{f_{ab}}{\sum_{c,d} f_{cd}}$

• BLOSUM-L $(a,b) = \log \frac{p_{ab}}{p_a p_b}$
Example score


- Related model $M$: Assume 50% similarity
  \[ p_{aa} = \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8} \]
  \[ P_{ab} = \frac{1}{2} \cdot \frac{1}{(4^2 - 4)} = \frac{1}{24} \]

- $m = \log \frac{P_{aa}}{p_a p_a} = \log \frac{1/8}{1/4 \cdot 1/4} = 1$

- $s = \log \frac{P_{ab}}{p_a p_b} = \log \frac{1/24}{1/4 \cdot 1/4} = -0.585$