Introduction to Computational Biology
Lecture # 5: PSI-BLAST

Barak Segal
21/1/2008

1 Course Introduction

The course’s objective is to study computational methods used for solving biological-molecular problems. We will gain perspective on the different types of research questions in the field and the various approaches for solving these questions. If in last year’s seminar we used many algorithms as ‘black boxes’, in this course we’re going to ‘look under the hood’ and see how they work.

A biological problem can be partitioned to three layers:

1. **Modeling** - Translating the real-life problem into a mathematical model.
2. **Algorithms** - Solving the mathematical problem using relevant algorithms.
   It is critical to define the mathematical problem well, otherwise the algorithm may become cumbersome or provide weak results.
3. **Validation** - Once we obtain the results, we want to verify that they are valid.
   There are two approaches for performing validation:
   - Statistical Validation - e.g. checking the significance of the results - P-Value, etc.
   - Biological Validation - Comparing the results to ‘wet’ tests at the laboratory, check if the results provide biological insights and if they make sense biologically, etc.

The kind of problems we’ll encounter during the course:

- Sequences (comparing, finding short subsequences, evolution of sequences).
- Networks (genes interactions).
- Gene Expression.

2 Sequence Comparisons

2.1 Problem Definition

Given a pair of sequences:

\[
s = x_1, x_2, \ldots, x_n \\
t = y_1, y_2, \ldots, y_m
\]

We want to measure how similar these sequences are. Using this similarity we wish to measure the sequence \( s \) against a database in order to get the most similar sequences to \( s \).
2.2 The Distance Function

The distance is measured using a distance function, which depends on the alignment between the two sequences. This function gives a certain penalty for gaps in the alignment. In previous lessons we used dynamic programming techniques to create an alignment, and we also used heuristics to improve the algorithm’s runtime complexity.

One of the common approaches for building the distance function is basing the function on the amino acids’ divergence throughout the evolution. For example, if amino acid ‘a’ often switches to amino acid ‘b’ during the evolution, the distance function $\sigma(a, b)$ will receive a high score.

Let us define the distance function:

**Definition 2.1** The Distance function:

$$\sigma(a, b) = \log \frac{P_1(a, b)}{P_0(a) \cdot P_0(b)}$$

- Our zero hypothesis is that the two amino acids do not share the same evolutionary ancestor.
- $P_1(a, b)$ - The probability that two proteins from a common evolutionary source will have ‘a’ and ‘b’ at the same position (this does not imply a specific position, but any general position within the sequence).
- $P_0(a)$ - The probability to find amino acid ‘a’ randomly in any given position, independent of other amino acids.

### 2.2.1 Estimating $P_0(a)$

$P_0(a)$ can be estimated empirically according to the frequency of ‘a’ in a large-enough sample of proteins. This might be a little problematic - for example, if the amino acid distribution is taken from an existing database, the data may be biased since certain ‘interesting’ proteins could be overrepresented. Nevertheless, it seems that contemporary databases are big enough to provide a truthful representation of the protein space.

A possible solution for the problem is to use a non-redundant database such as Uniref90, which do not contain proteins that have more than 90% similarity.

### 2.2.2 Estimating $P_1(a, b)$

Calculating $P_1(a, b)$ is a little more difficult. We can get an approximation of it by taking an existing database, and performing sequence alignment between the proteins in it using a certain made-up distance. Once they are aligned, we can collect statistics both on conserved amino acids and alternating amino acids.

Example: we observe (after aligning the proteins in the database) that amino acid ‘a’ and amino acid ‘b’ usually appear in the same positions in different proteins, therefore the value of $P_1(a, b)$ is high.

This presents two problems:

1. The level of conservation may be dependent on the location of the amino acid within the protein (e.g. amino acid ‘a’ is conserved at the C-terminus but often change to amino acid ‘b’ at the N-terminus).
2. Two proteins could be 99% identical, thus their evolutionary distance is very short. Proteins could also be very dissimilar, implying long evolutionary distance (or no relation at all). Either way, such distance won’t enable us to learn anything about the probability of amino acid switching.

A good solution for the second problem is the BLOSUM matrix.
2.2.3 BLOSUM

BLOSUM stands for BLOcks of Amino Acid SUBstitution Matrix. BLOSUM$_n$ means that we sample proteins with a similarity of no more than n%. We observe the amino acids distribution in these proteins’ alignment within conserved blocks (blocks that do not have gaps in the alignment). From these conserved blocks an amino acid substitution matrix is created, where the score in a cell is high if the two amino acids corresponding to it are frequently substituted. BLOSUM tables with different n’s can be used for different goals.

Example: A frequently used n-value is n = 62. Using BLOSUM$_{62}$, the evolutionary distance between the proteins is good enough to indicate that if amino acids 'a' and 'b' usually appear in the same positions in the blocks, then it is a good indication of conservation between these two amino acids.

Note: the similarity percentage is not actually a marker for evolutionary distance, but it is measurable and easy to work with.

2.3 Evaluating our solution

We proposed testing the sequence s against the whole database using the $\sigma(a, b)$ function, to find similar proteins. But is this a good solution?

Let us explore some of the problems with this solution:

1. Each Amino Acid is regarded outside of its context, ignoring which amino acids surround it.
   (Apparently, this fact does not disrupt the results too much)

2. Biological insights are ignored.
   For example, suppose the amino acid at position 7 is hydrophobic. As a rule of thumb, hydrophobicity is an attribute that is conserved throughout the evolution. However, we do not know whether in this particular case position 7’s hydrophobicity is critical (an amino acid in the active part of the protein), or negligible (part of a loop, etc.).

Both problems originate from the fact that we aren’t familiar with the protein a-priori.

Possible solutions:

- Gain insights using a domains database.
- Search the entire protein space for the top proteins which are most similar to s. We can check which amino acids are important and which aren’t within these proteins, which are most likely very close in the evolutionary sense. Using this knowledge we create a new distance function.

The latter solution brings the discussion to PSI-BLAST.

3 PSI-BLAST (Position-Specific Iterative BLAST)

3.1 Synopsis

For a sequence $s = x_1, x_2, \ldots, x_n$ we perform BLAST against all the sequences in the database and find the sequences whose similarity to s surpasses a given threshold. Let us denote these similar sequences: $t_1, \ldots, t_k$. Using the alignment provided by BLAST, we calculate the amino acids distribution for each position (figure 1). The distribution enables us to define a new distance function for each position separately.

3.2 PSI-BLAST Distance Function

As stated before, the distance function is position-specific. Therefore, we can define it for the i’th position as:

$$\sigma_i(x_i, b)$$
We observe that since \( \sigma_i \) is built for \( x_i \), we can define the function without explicitly mentioning \( x_i \).

Using this observation and our previous definition of \( \sigma \) we obtain the following definition:

**Definition 3.1** The PSI-BLAST Distance function (for a specific sequence \( s \)):

\[
\forall i \in 1, \ldots, n \quad \sigma_i(b) = \log \frac{P_i(x_i, b)}{P_0(x_i) \cdot P_0(b)} = \log \frac{P_i(b|x_i)}{P_0(b)}
\]

Examples:

- Suppose that at position \( i \), the amino acid distribution is 97% K and 3% L. \( \sigma_i(K) \) will return a high score.
- Suppose that at position \( j \), the distribution is 10% K and the rest is composed of other amino acids. \( \sigma_j(K) \) will return a low score.

The representation that is used to portray the distance functions that compose \( \sigma \) is called a PSSM.

**Definition 3.2** PSSM - Position Specific Score Matrix. PSSM is matrix of score values where each cell in the matrix specifies the frequency of a specific amino acid in the \( i \)'th position in the proteins aligned against \( s \). (figure 2). Sometimes alternatively named:

- **PSMW** - Position Specific Weight Matrix (PSWM).
- **PWM** - Position Weight matrix.

Note: Apart from the alignment PSI-BLAST provides, the PSSM that is created by it provides significant biological insights on the conservation levels of the original protein sequence.

### 3.3 Implementation

The PSI-BLAST algorithm may be repeated iteratively, each time creating a distance function which will be used to create the next distance function. This allows PSI-BLAST to relate to sequences which are farther away from our original sequence in the evolution.

As in pairwise sequence alignment, we can implement PSI-BLAST using dynamic programming. Following from what we have seen in previous lectures, we get:

\[
V(i, j) = \max \left\{ V(i − 1, j − 1) + \sigma_i(y_j), V(i, j − 1) + \sigma_0(−), V(i − 1, j) + \sigma_i(−) \right\}
\]
Introduction to Computational Biology Lecture # 5

Figure 2: An example of a PSSM. The columns represent the alignment positions. Each row represents a certain amino acid. The matrix’s cells represent the frequency of the corresponding amino acid in the aligned sequences in the corresponding position.

Where \( \sigma_0(\cdot) \) is a constant describing the penalty of placing a gap in any position in the second sequence (not the original one) and \( \sigma_i(\cdot) \) is the score of placing a gap parallel to the original sequence in the \( i \)'th position.

PSI-BLAST’s designers wanted to use a variation of the BLAST method in their implementation. Reminder: pairwise BLAST iterates through a sequence, indexing keywords in the sequence of a certain length (6 for example), and seeks keywords of the same length, which pass a certain similarity threshold to keywords found previously, in the other sequence. These similar keywords (HSP - High-scoring Segment Pair) appear as diagonals in the dynamic programming matrix, where on the longest diagonal (consisting of patches of HSPs), the Banded DP algorithm is run (for more details see lecture 2). The difference in PSI-BLAST, is that now testing of the similarity between the keywords changes since each position \( i \) in the original sequence \( s \) has a different score function \( \sigma_i \).

### 3.4 Issues

These are some of the major problems with PSI-Blast:

1. We may get sequences with varying evolutionary distances. Evolutionary close sequences will have ‘false’ conservation - most of the amino acids will appear conserved, regardless of their actual conservation level. Similarly, sequences which are far off do not indicate conservation or divergence of amino acids.

   **Solution:** We use the same method as BLOSUM - choose only sequences with a given evolutionary distance range.

2. We assume that the sequences are independent of each other, though this may not always be the case.
   
   For example: *the researchers who manage a certain database could be focusing on E.coli, and as a result the database will be saturated with proteins originating from E.coli strains and the results will be biased accordingly.*

   **Solution:** We can give less weight to very similar sequences when creating the distance function. There are heuristic methods to do it fast and efficiently.

3. The sample space could be too small. Thus, we may need to base our score function from a set which is too small.

   **Solution:** This problem is more complex than it seems. Even if the sample size is considerable, we could still get 0 or a small count for specific amino acids in specific positions. Since the distance function uses log, this result in a very low score. For example:

   \[
   \text{if } P_i(a|x_i) = 0 \Rightarrow \sigma_i(a) = -\infty.
   \]
In the next lecture we will see how this can be solved using by adding pseudo observations (pseudocounts), that affect all the counts equally (for example, adding 5 to each count of the amino acids in all positions), and prevent the occasion where we get extremely small number of amino acids.