String matching problems from Bioinformatics which still need better solutions Extended abstract

Gaston H. Gonnet
ETH Informatik
Zurich, Switzerland

Abstract

Bioinformatics, the discipline which studies the computational problems arising from molecular biology, poses many interesting problems to the string searching community. We will describe two problems arising from Bioinformatics, their preliminary solutions, and the more general problem that they pose. We hope that this will encourage researchers to find better, general, algorithms to solve these problems.

1 Introduction

String searching problems in bioinformatics are normally posed in terms of returning a score as opposed to a binary answer. This is quite fundamental, most of the time these searching problems return the likelihood of a certain event. Consequently we will assume that all of our searches return a score, and we will be searching for the highest scoring strings.

Molecular information, for the purpose of this paper, consists of sequences of DNA/RNA (4 symbols) or sequences of amino acids (20 symbols) normally called proteins.

2 Identifying α-helices

The first problem is related to the identification of α-helices in protein sequences. A similar, but simpler, problem arises with the identification of β-sheets (or β-strands). We will concentrate on the first problem.

An α-helix is a secondary structure of proteins, found in almost every protein. About 10% of the total amino acids in any genome are part of α-helices. Recognition of α-helices from the primary structure, that is the sequence of amino acids, is usually called "secondary structure prediction", and is a very important and active area of research. An α-helix is a helical structure which rotates exactly $100°$ with every amino acid. Its existence and stability is normally due to its surroundings, a helix cannot stand by itself. In particular, most helices find themselves in the interface between the inner and outer part of the protein. This information is very useful to find (or confirm) helices, as one side of the helix is hydrophobic (the inner side) and the other is hydrophilic (the outer side, typically exposed to water).

Each amino acid has an index of hydrophobicity. Hydrophobicity indices range from -1 to 1. A natural score for a sequence of L amino acids starting at position j is the the internal product:

$$R = \sum_{i=j}^{j+L-1} H_i \sin(\frac{2\pi i \times 100}{360} + \theta)$$

where θ is the phase angle of the helix with respect to the inner/outer limit and H_i is the hydrophobicity index of amino acid i. Expanding and collecting we obtain:

$$R = S_j \cos\theta + C_j \sin\theta$$

with

$$S_j = \sum_{i=j}^{j+L-1} H_i \sin 5/9\pi i \qquad C_j = \sum_{i=j}^{j+L-1} H_i \cos 5/9\pi i$$

The phase angle θ which gives the maximum score for each j is easily obtained by solving the derivative of R equated to 0. The maximum score is given by:

$$R = \sqrt{C_j^2 + S_j^2}$$

A linear time algorithm is easy to derive (for a fixed L). The values of C_j and S_j can be updated in constant time by adding the $(j+L)^{th}$ term and subtracting the j^{th} term.

It is clear that such an algorithm will work for any score which is a function of sums (or other functions with an inverse) over individual values.

This optimal score has a surprisingly simple geometrical interpretation. If every H_i is viewed as a vector from $(0,0)$ to $(H_i \cos 5/9\pi i, H_i \sin 5/9\pi i)$, then we are looking for the longest path of L consecutive vectors.

Problem 1: Is there a more efficient algorithm to search an amino acid sequence for various values of L, e.g. L=5..30 ? (Even if it is more efficient in the constant term, not necessarily asymptotically.)

Problem 2: Are there any mathematical properties of the score R which make this search more efficient?

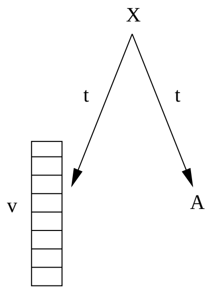

Figure 1: matching a probabilistic position against an amino acid

3 Matching profiles or probabilistic sequences

A probabilistic pattern is a sequence which is defined by probability vectors in each position. It is interpreted as each position having several choices (all the values of the probability vector which are non zero). This is a natural generalization of matching a regular expression, where we additionally want to find the most likely match (highest probability) of all the possible ones.

There are three main uses of probabilistic sequences: Probabilistic ancestral sequences, Patterns and Motifs. The scores that we compute in this section can be used for direct string matching or can be used in the dynamic programming alignment of sequences with indels.

Scoring an amino acid against a probabilistic sequence is not completely trivial if it is to be done according to the standard model of evolution. The score is given by the quotient between the probability of being homologous, i.e. coming from a common ancestor divided the probability of being matched by chance (null hypothesis).

Let v be the probability vector to be scored against amino acid A and t the distance between x and A and between x and v as shown in figure 1. The quotient of probabilities is:

$$\frac{Pr\{\text{common ancestor}\}}{Pr\{\text{random occurrence}\}} = \frac{\sum_x f_x \left(\sum_i Pr\{x \rightarrow i\} v_i\right) Pr\{x \rightarrow A\}}{\left(\sum_i v_i f_i\right) f_A}$$

That is, the probability is simply summed over all possible values of the ancestor (x) and all possible values of the probabilistic position (i). Using Dayhoff's mutation matrices, doing some simplifications and taking logarithms to have an additive score, we simplify the score to:

$$R = 10 \log_{10} \frac{u_A}{fv}$$

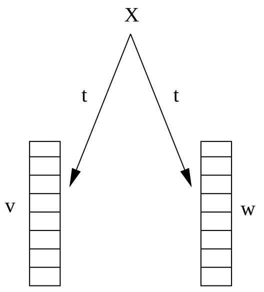

Figure 2: matching two probabilistic positions

where $f = (f_1, f_2, ...)$ is the vector of natural amino acid frequencies, $u = v^T M^{2t}$ and M is a 1-PAM mutation matrix. By precomputing the vectors u and fv for the pattern, we can compute the score for each position in time $O(1)$.

Scoring a probabilistic sequence against another is done following the same lines. Figure 2 describes the situation of scoring one particular position against another. This time we have a triple sum, over the ancestor and over each of the two probabilistic descendants, i.e.:

$$\frac{Pr\{\text{common ancestor}\}}{Pr\{\text{random occurrence}\}} = \frac{\sum_x \sum_i \sum_j f_x Pr\{x \rightarrow i\} v_i Pr\{x \rightarrow j\} w_j}{fv\ fw}$$

We use the same technique as before to simplify the computation and obtain:

$$R = 10 \log_{10}(u^* w^*)$$

where $v^* = \frac{v}{fv}$ can be precomputed for the first sequence, $w^* = \frac{w^T M^{2t} F}{fw}$ can be precomputed for the second sequence and F is a diagonal matrix containing the natural frequencies, i.e. $F_{ii} = f_i$. The computation of the score, besides precomputation, requires an internal product of two vectors of size 20 (number of symbols). This is quite efficient since the precomputation is dwarfed by the matching phase.

Problem 3: Is is possible to compute the scores more efficiently for this case?

Problem 4: All these string matching algorithms assume a brute force underlying string searching strategy. Is it possible to improve the efficiency by using the ideas of better string searching algorithms to these scores?