A Linear Programming Based Algorithm for Multiple Sequence Alignments

Fern Y. Hunt
Anthony J. Kearsley
Agnes O'Gallagher
National Institute of Standards and Technology
Mathematical and Computational Sciences Division
Gaithersburg, Maryland 20899
fern.hunt@nist.gov

Abstract

In this brief paper we discuss an approach to multiple sequence alignment based on treating the alignmenet process as a stochastic control problem. The use of a model based on a Markov decision process leads to a linear programming problem whose solution is linked to a suggested alignment. Our goal is to avoid the expense in time and computation encountered when dynamic programming based algorithms are used to align a large number of sequences. The dual linear programming problem can also be defined. We implemented the method on a set of cytochrome p450 sequences and compared our suggested alignments of 3 sequences with that obtained by CLUSTALW. Further details can be found in the reference [1] .

1. Introduction

A multiple sequence alignment of k strings can be thought of as a two- dimensional array $k \times L$ where L is the length of a single string. Algorithms for multiple alignment are based on the minimization of the total cost over the set of all alignments A,

$$\min_{\mathsf{A}} \sum_{j} c(\mathsf{a_j}), \tag{1}$$

where $\mathsf{a_j}$ is the jth column of the array, and $c(\mathsf{a})$, the cost of the aligned column a, is specified in advance. Dynamic programming is the most widely employed method for aligning small numbers of sequences. However there is an increasing need for efficient methods of aligning large numbers of sequences and/or very long sequences using algorithms that can take advantage of advances in large scale computation. Development and implementation of algorithms for solving large linear programming (LP) problems is a well established area of research with many industrial applications. Our goal is to use advances in this area to address the issue of computationally efficient large scale alignment. The LP discussed here is a standard formulation of a problem in Markov decision processes. Markov decision theory has been used to solve a variety of optimization problems in fields such as economics, biology and network control. However its application to alignment problems appears to be new.

2. Markov Decision Process

A markov decision process or controlled markov chain is a stochastic process $(X_t, a_t), t = 0, 1, \cdots$, where $X_t \in \mathbf{X}$, a finite sample space and $a_t \in A$, with A, called the set of actions. An element of the set $\mathbf{X}$ is called a state of the process. Define the history of the process as the sequence, $h_t = (x_1, a_1, \cdots x_{t-1}, a_{t-1}, x_t)$. A policy is a sequence $\pi = (\pi_1, \pi_2, \cdots)$. If history h_t is observed at time t then the controller chooses action a with probability $\pi_t(a|h_t)$. In full generality a policy is a sequence of probability measures indexed by time, but in many applications of interest these measures simplify. In particular a deterministic policy is defined by a function $f : X \to A$ with $a_t = f(x_t)$. Once an action at time t is chosen, the next state is chosen at random with probability $P_{ij}(a)$ where $X_t = i, X_{t+1} = j$ and $a_t = a$ where $i, j = 1, \cdots m$.

To fix ideas suppose our task is to find an optimal global alignment for 3 sequences. The MDP(Markov Decision Process) model for our application is based on an extended alphabet $A \cup \{-\}$. Here $\mathbf{X} = \{\mathbf{A} \cup -\}^{\mathbf{3}}$. Each state is a 3-tuple of elements from the extended alphabet associated with a column of a triple alignment of single sequences. The time index is the index of aligned columns moving from left to right. The set of actions is $A = \{0, 1\}^3$. At time t, the action a_t is a 3-bit binary that describes the pattern for the

0-7695-2000-6/03 $17.00 © 2003 IEEE

next column. The number 1 means a letter appears, while $-$ is denoted by 0. The protein alignment problem therefore, has a state space with 21^3 elements and 8 actions. Each pair (X_t, a_t) has a cost $C(X_t, a_t)$ associated with being in state X_t and making the decision to choose action a_t.

3. The Linear Programming Problem

There are a variety of possible expressions for the total cost of the alignment associated with the MDP described in the previous section. We will discuss the expected discounted cost. We seek a policy whose associated expected discounted cost is minimal. The minimal cost itself can be written for a fixed discount rate $0 < \alpha < 1$ as,

$$V_\alpha(i) = \min_\pi \mathbf{E}_\pi[\sum_{t=0}^{\infty} \alpha^t C(X_t, a_t)|X_0 = i] \qquad (2)$$

where minimization is performed on the set of all policies π. Thus $V_\alpha(i)$ is the least mean discounted cost of a state-action path starting at state i. Our goal is to find a policy π^* whose associated cost is this cost. Such an optimal policy exists for our problem and it is a deterministic policy. To find it we must first find the costs themselves. They are the solution of the following LP [1]

$$\max \textstyle\sum_{i=1}^{m} u(i) \qquad (3)$$

$$s.t. \forall a \quad u(i) \leq C(i,a) + \alpha \textstyle\sum_{j=1}^{m} P_{ij}(a) u(j) \qquad (4)$$

Here maximization is over all bounded nonegative functions, $u : X \rightarrow R^+$. Once such a function u is found, it can be shown that $u = V_\alpha$.

3.1. Alignment using the solutions of the LP

To generate a suggested alignment from the solution of equations (3)-(4), we recall (see reference [1] and cited references there) that $V_\alpha(i)$ satisfies the relation,

$$V_\alpha(i) = \min_a [C(i,a) + \sum_{j=1}^{n} P_{ij}(a) V_\alpha(j)] \qquad (5)$$

known as Bellman's equation. Let a_i^* be the value of a for which the minimum is attained. Then a_i^* is the assigned value for state i under the optimal deterministic policy. The suggested alignment comes from the insertion of gaps according to the zeros occurring in $\{a_i^*\}$.

3.2. Computational Results

The constraints in equation (4) require a knowledge of the elements $\{P_{ij}(a)\}$. These are estimated from a set of aligned protein sequences. Counts of the number of transitions from one state- an aligned column, to the next along with the action represented by the letter-gap pattern of the suceeding column are made and the frequency is taken to be the estimate of the transition probability. Thus if $n(i,j,a) =$ # of times $X_t = i, X_{t+1} = j$, action a is taken, and $n(i,a)$=# of times $X_t = i$, action a is taken

$$P_{ij}(a) \approx n(i,j,a)/n(i,a).$$

We implemented this method and computed a suggested alignment for 3 sequences coming from the cytochrome p450 family. The $\{P_{ij}(a)\}$ were computed from a set of 98 triples created from an original set of 100 aligned sequences. Each sequence was 775 symbols long. The sequences were aligned using CLUSTALW. The matrix building software used to identify states (4714) and compute transition frequencies took 155 seconds. The linear programming problem was solved with PCx and this took 255.79 seconds. The cost functions used in the problem came from Blosum62 along with a gap penalty and column costs were computed by adding all pairwise costs and dividing by 3. General agreement with the results of actual alignments was good. A discussion of the comparison between our suggested alignment and that given by CLUSTALW can be found in [1].

4. Other Work and Conclusion

Another linear programming problem the so-called dual problem can also be formulated. Here we seek quantities $\{x_{ja}\}_{j \in X, a \in A}$ that can be interpreted as the amount of discounted time the process spends in the state-action (j,a), that minimize the resulting cost,

$$\sum_{i \in X} \sum_{a \in A} x_{ia} C(i,a).$$

This problem and a related dual problem associated with the long term average cost will be implemented in the future. In the latter case we have been able to show asymptotic optimality in a pathwise sense as well as in the sense of optimal expected cost [1].

In conclusion, we have described the very first steps in creating a flexible software platform that on the one hand allows users to align sequences using cost functions and training data of their choice but at the same time allows for the inclusion of new linear programming and optimization solvers as they develop.

References

[1] F. Y. Hunt, A. J. Kearsley, and A. M. O'Gallagher. A linear programming based approach for Multiple Sequence Alignment. *in preparation*, 2003.