RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

IN-61
43062

Evolutionary Tree Reconstruction

8P

Peter Cheeseman and Bob Kanefsky

March 1990

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 90.27

NASA Cooperative Agreement Number NCC2-387

Evolutionary Tree Reconstruction

Peter Cheeseman and Bob Kanefsky

Research Institute for Advanced Computer Science
NASA Ames Research Center - MS: 230-5
Moffett Field, CA 94035

RIACS Technical Report 90.27

March 1990

Work reported herein was supported in part by Cooperative Agreements NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA).

Evolutionary Tree Reconstruction

Peter Cheeseman (RIACS) and Bob Kanefsky (Sterling Software)

Address: NASA Ames Research Center, Mail Stop 244-17, Moffett Field, CA 94035
Email: Cheeseman@Pluto.ARC.NASA.gov and Kanef@Charon.ARC.NASA.gov

Abstract

This paper describes how Minimum Description Length (MDL) can be applied to the problem of DNA and protein evolutionary tree reconstruction. If there is a set of mutations that transform a common ancestor into a set of the known sequences, and this description is shorter than the information to encode the known sequences directly, then strong evidence for an evolutionary relationship has been found. We describe a heuristic algorithm that searches for the simplest tree (smallest MDL) that finds close to optimal trees on our test data. Various ways of extending the MDL theory to more complex evolutionary relationships are discussed.

Introduction

A major challenge for the 90's will be making sense of the flood of data generated by the Human Genome project. If we know the evolutionary history of genes, we can often make strong predictions about their function, by comparison with other more distantly related genes whose function we do know. Such sequence comparisons are routinely performed for newly sequenced genes using existing gene databases (e.g. GenBank), and many interesting, unexpected relationships have been discovered. In addition to using evolutionary relationships to predict the function of genes, evolutionary tree reconstruction is of interest in its own right, as it often shows relationships between species that cannot be reconstructed from the fossil record alone. Also, evolutionary information provides useful information about protein 3-D structure, because proteins whose structure is known can give structural information about related proteins. We present a MDL approach to evolutionary tree reconstruction below.

Basic Theory

The problem addressed in this paper is: "Given a set of known DNA sequences, find the most probable evolutionary tree (or trees) that relates these sequences". A simple example of an evolutionary tree is shown in Fig. 1, where the known sequences are shown on the bottom row, and all other sequences are hypothetical reconstructions of ancestors. The length of a branch from parent to child sequence is proportional to time since divergence. The mutations that change a parent into the corresponding child sequence are shown on the connecting branches. A tree, such as Fig. 1, is a possible evolutionary model

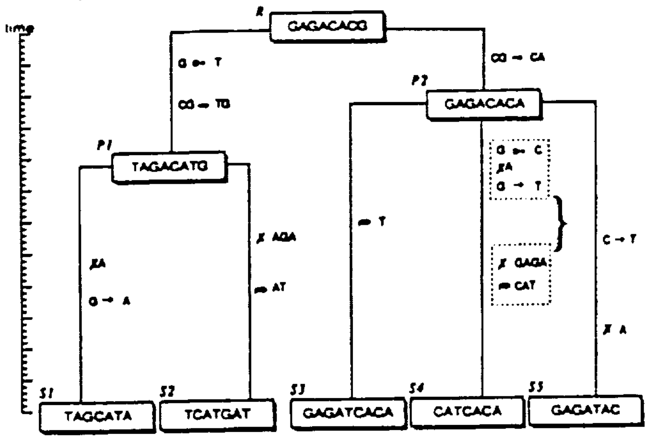

Figure 1: A Simple Evolutionary Tree

that "explains" how the known sequences (on the bottom row) were created. However, there are many possible trees, all having the same terminal sequences; so which tree (or set of trees) should be preferred over another? Intuitively, the "simplest" tree seems the most plausible, and this intuition has lead to the "parsimony" approach to evolutionary tree reconstruction [2]. The usual definition of parsiminoy is unable to account for more complex evolutionary models, such as unequal rates of mutation on parallel branches, and so in some cases asymptotically approaches the wrong answer as more data is provided [2]. The MDL approach described below captures the basic intuition behind the parsimony approach, but defines "parsimony" in terms of probabilities.

Although Fig. 1 relates all the known sequences through a single tree, we consider the general problem to include reconstructing multiple trees that jointly "explain" the sequence data. That is, we are not assuming that all sequences had a common ancestor at some time in the past, although our example will be developed under this assumption. For a set of related proteins, a graph (instead of a tree) is sometimes the correct representation of the evolutionary events[1].

[1]Sometimes a protein evolves from a combination of parts of

MDL Formulation of Problem

By Bayes Theorem, the relative posterior probability ratio of two different trees T_i and T_j given a set of known sequences S is:

$$\frac{p(T_i|S)}{p(T_j|S)} = \frac{p(T_i)p(S|T_i)}{p(T_j)p(S|T_j)}. \qquad (1)$$

By taking logarithms of this equation and negating we get:

$$\begin{aligned} -\log p(T_i|S) - (-\log p(T_j|S) = \\ -\log p(T_i) - (-\log p(T_j)) \\ -\log p(S|T_i) - (-\log p(S|T_j) \qquad (2) \end{aligned}$$

From information theory, $-\log p_i$ is the minimum possible message length to encode the ith outcome if this outcome had probability p_i. If the base of the logarithm is 2, then the message length is in bits. It is clear from Eqn. (2) that the particular tree T_i with the maximum posterior probability relative to any other tree T_j is also the tree with the shortest relative encoding (MDL). For each tree T_i, the MDL consists of two parts: a part to describe the model selected (T_i) and a part to describe the data given the model. To apply Eqn. (2) we need the prior probabilities of trees, $p(T_k)$, and the likelihood function, $p(S|T_k)$. Note in the following, a tree T refers to the hypothetical construct only and does not include the known sequences S Also, we only calculate the length of the theoretical minimum message; we do not actually do the encoding.

Tree Prior Probabilities

If the user has prior information about the target evolutionary tree, such as from fossil evidence, then this can be used directly in Eqn. (2), but this information is rarely available. A common evolutionary assumption is that the probability of a mutational event or branching event occurring per unit time is independent of the absolute time. These independence assumptions imply that the probability of a sub-tree is independent of the events in the super tree and only depends on the immediate parent and the time since divergence from the parent. Symbolically, these independences can be expressed as:

$$\begin{aligned} p(\text{Tree}) = p(\text{Root}) \\ p(\#\text{branches})p(\text{Subtree}_1|\text{Root})\cdots p(\text{Subtree}_N|\text{Root}) \end{aligned}$$

where the probabilities $p(\text{Subtree}_i|\text{Root})$ are recursively defined by a similar decomposition in which a subtree only depends on its immediate parent. This recursion stops when a subtree has only known terminal sequences for children. Taking Logarithms of this equation, including the recursive expansion of the subtrees, leads to a simple additive form for $-\log p(T_i)$. This additive form corresponds to a recursive coding scheme, where all immediate children are described as the result of a particular set of mutations of the parent. Since the root does not have a parent, it must be described separately. The leaf nodes are not regarded as part of the tree T_i. However, the definition of the conditional probability of a child given its parent is the same everywhere in the tree, including the leaf nodes. Because of the information theoretic interpretation of Bayes theorem, we believe that choosing a coding scheme is equivalent to accepting particular prior probabilities (and *vice versa*).

existing proteins (domains), and so can have multiple parents.

Sequence Probabilities

Except for the root, which is coded directly, the information required to describe all other nodes in the tree is reduced by describing each child given its parent. This reduced encoding uses the conditional probability of a child sequence given (a) its parent, (b) a set of mutations that transform the parent into the child and (c) a time difference between parent and child, i.e.

$$p(S_{\text{child}}|S_{\text{parent}}, \text{ mutations }, \text{ time difference}). \qquad (3)$$

This requires finding the most probable set of mutational events that could have transformed the parent into the child, or at least a very probable set. For example, in Fig. 1, two alternative sets of mutation between $P2$ and $S4$ are given—many more are possible. We note that the tree building procedure described by in [2] finds the maximum likelihood tree topology and branch lengths. It does not infer particular ancestral sequences, but averages over all possible ancestral sequences. This approach is answering a different question than the one addressed here, and it has difficulty taking into account insertions and deletions. Methods we use for finding the most probable mutation events and time differences are described below, but here we assume they are known. There are three types of time dependent mutations that transform the parent sequence into the corresponding child sequence; point mutations, insertions and deletions. Our coding scheme for a sequence transformation is as follows.

1. Deletions. At each letter[2] in the parent string, we state whether it is the beginning of a deleted

[2]We will use the term "letter" to refer to either nucleic acids or or amino acids, as appropriate.

string or not. Thus, if there are 300 letters in the parent, we encode up to 300 *Yes* or *No* messages. Since deletions are rare, each *No* message is typically a small fraction of a bit; its length is given by $-\log[1 - p_{del}(t)]$, while a longer *Yes* message has length $-\log p_{del}(t)$ bits. The *Yes* and *No* messages resume from the next letter that was not deleted. Whenever a deletion event occurs, the length of the deletion (beginning at the current letter) is described, using $-\log p(n)$ bits, where n is the length of the deletion. The t parameter is the number of time units between parent and child. Note that we are assuming that deletion length probabilities are independent of time.

2. Point mutations. For each letter we give a message describing the fate of that letter of length $-\log p(\text{new} \mid \text{old})$, where we use the fact that the probability that a particular base or residue will turn into another depends on what it was before. For example, a purine base is about twice as likely to turn into another purine (i.e. A $\rightarrow$ G) than it is to a pyramidine (e.g. A $\rightarrow$ (T or C)). An example change probability matrix is shown in Fig. 2. This coding scheme makes the simplifying assumption that the probability of a point mutation does not depend on its neighboring bases or residues or its location in the sequence. Both these assumptions are biologically incorrect. For example, a C followed by a G is much more likely to be transformed into a T than a C followed by A,C or T (CpG decay). Similarly, in coding DNA, silent third position bases have a much higher point mutation rate than non-redundant positions.

3. Insertions. These are encoded much the same way as we encode deletions. If the parent is 300 letters long, we encode exactly 301 *Yes* and *No* messages. This time, in addition to its position and length, we must describe the letters in each inserted string. Each inserted letter is described in $-\log p_i$ bits.

The encoding method described above is similar to describing a set of "edit commands" that transform the parent into the child sequence. However, instead of commands, such as: "skip the next 20 letters", we prefer to use the probabilities of different mutations at each position in the sequence. The use of biologically meaningful mutation probabilities allows learning to take place during tree reconstruction. The above coding procedure allows the message length $(-\log p(T_i))$ of a tree to be calculated provided all the probabilities mentioned above are known. The known sequences at the leaf nodes are not encoded in the tree description, since they are implied by it.

Time-Dependent Probabilities

For a particular letter, the probability that it will undergo a point mutation is a function of the time between the parent and the child. For a given number of "time units", these point mutation probabilities can be deduced from the unit-time transition matrix. An example of a unit time matrix for DNA is:

	A	C	G	T
A	.99	.0025	.005	.0025
C	.0025	.99	.0025	.005
G	.005	.0025	.99	.0025
T	.0025	.005	.0025	.99

Figure 2: Point Mutation Probability Matrix

To obtain the point mutation probability for n time units, this matrix is raised to the power n. The unit time matrix is chosen so that the probability of a point mutation in one time unit is .01. Even after 25 time units, the probability of a base remaining unaltered is .778, so that multiple mutation events at the same location are unlikely even for this large time difference. Note that we have quantised time, so that the information required to describe the time between a child and its parent is small $(-\log p(t))$. The message length clearly depends on the quantisation level selected, so for a given tree there is an optimal quantisation level. For simplicity, we assume that the time in which 1 PAM (1% point mutation rate) occurs is adequate. An improved version would dynamically optimise this parameter during tree building.

Probability of Known Sequences

The probability of all the known sequences, $p(S \mid T_i)$ in equation (1), is given by:

$$p(S \mid T_i) = \prod_j p(S_j \mid \text{Immediate parent of } S_j); \qquad (4)$$

where S_j is a particular known sequence. The individual probabilities in Eqn. (4) are coded the same way as any other child given its parent, as described above.

Dynamic Probability Learning

If all the component probabilities required above are known from past experience, then these can be used directly, with no need for learning. However, past experience usually only provides rough prior probabilities that can be used as initial values. As the tree is built, further information becomes available from the frequencies of types of mutation events in the current tree. To exploit this additional information, we compute our tree MDL serially, and update the probabilities for the rest of the tree, using information from the tree encoded so far. We use the standard Bayesian probability update formula, illustrated here for the probability of a deletion:

$$P_{del} = \frac{n_{del} + r_{del}}{N + R} \qquad (5)$$

where n_{del} is the observed number of deletions encountered so far, N is the total number of letters (deletion or not), r_{del} is a prior weight for deletions and R is the total prior weight. The situation for updating point mutation probabilities is not as simple, because the mutations typically occur after a number of time steps, while the transition probability matrix is defined for a single time step. We solve this problem by arbitrarily assigning a given point mutation to a single time step and all the other time steps count as no mutation. This approximation depends on the probability of alternative multiple mutations with the same end result being very unlikely.

We have now described how to compute the component probabilities in Eqn. (2), so that relative MDL, can be found. We use this relative MDL to search for the lowest MDL tree, as described below.

Sequence Alignment

The problem of finding the most probable mutation list, given a parent and a child sequence, is the standard pairwise sequence alignment problem. An alignment algorithm typically uses a "penalty function" that assigns a penalty to any proposed mutational events—the goal is to find the alignment with the minimum penalty. In the MDL approach, these penalties turn out to be mutation probabilities in disguise. There are many alignment algorithms in use. Generally, their performance depends on how well the user adjusts the penalties. We have implemented an MDL based alignment algorithm in LISP that uses given probabilities. This allows it to use knowledge of the estimated time difference between a parent and child, and to adjust itself to the dynamically changing posterior probabilities for different types of mutations. It is very similar in theory and practice to the independently conceived procedure described by [1].

The Tree Building Procedure

The previous sections describe a MDL measure for deciding which of two alternative trees is more probable. This suggests an iterative improvement procedure for finding an optimal tree—start with a tree that is approximately right, then look for local improvements of the MDL measure. The obvious way of constructing a good initial tree is to build an initial $(n \times n)$ "distance matrix" based on the MDL alignment values, then build the tree bottom-up. Unfortunately, the cost of constructing the distance matrix can be prohibitive—especially as the alignment algorithm will spend a lot of time producing alignments of very distantly related sequences that never get used by the tree building procedure. To reduce this cost, we have implemented a heuristic initial tree building procedure. Instead of using an alignment algorithm to produce distances between sequences, we use a combination of approximate measures that correlate with evolutionary distance. The best measures between sequences we found, in order of increasing accuracy, are:

- Length Ratio—sequences that are close evolutionarily tend to be about the same length, because insertions and deletions are rare.
- Longest common subsequence—since close sequences have fewer mutations, this estimator correlates negatively with distance.
- Squared difference of uncommon hexamer densities—where a hexamer is a string of six letters. This estimator works because in closely related sequences the probability that short uncommon substrings are unaltered is high.

All these heuristic estimators are cheap to compute compared to a full alignment. The initial tree building procedure builds a binary tree from the bottom up by adding each new node into the tree at the place that minimises the total squared error of the combined estimators for that node, i.e. it finds the height h and the place in the current tree to insert a new parent of the current sequence so that the following measure is minimized:

$$M(h) = \sum_{k=1}^{3} \sum_{i} \frac{(m_i^k - h)^2}{(\sigma_k)^2} \qquad (6)$$

where k is the measure index, i is an index ranging over all sequences in the current tree, m_i^k is the kth heuristic measure between the current sequence and the ith sequence, and σ_k is the standard deviation of the kth measure. Because the positioning of the early members of the heuristic tree did not have the benefit of the constraining influence of the later nodes, existing nodes are re-added, until the tree does not change. This heuristic approach builds remarkably accurate initial trees on our test data. Part of the reason for this accuracy is that the tree is the result of forcing many independent pieces of evidence into a consistent tree, so statistical averaging compensates for the crudeness of the heuristics. The initial tree is used to guide which sequences to align, and where to put new ancestors in a bottom-up MDL tree construction.

Unfortunately, this bottom-up tree building procedure does not construct optimal trees by the MDL criterion. The reason is that in bottom-up tree construction there is a degree of arbitrariness in how to assign insertions and deletions to a new parent. However, once a tree is constructed, we use a set of optimisation operations to improve it. For example, whenever common deletions or insertions are detected on neighboring branches, we can change the parent to eliminate them. In all these local optimisation steps, the criterion is always "does the proposed change lower the MDL?". This opportunistic local optimisation procedure runs until it is unable to find further improvements. The resulting tree is not guaranteed to be the globally optimal MDL, but on artificial data where we know the "correct" answer, we found that this procedure gets close. Typically, our current implementation gets to within 20% of the MDL of the true tree, and does a fairly good job of placing related sequences together. Preliminary results on a real DNA dataset, containing 126 human alu sequences, yields a slightly smaller description-length than a multiple-alignment based on a "consensus sequence". Our description is still somewhat worse than the four-level hierarchy proposed by Smith and Jurka [3], becuse our optimisation operations are still too simplistic.

Extended Theory

We made a number of simplifying assumptions in the tree building procedure described above in order to produce a working realistic program. The theory can be extended to remove many of these assumptions, although the effect of such changes on the search procedure may not be simple. The simplest extension is to allow the probabilities of point mutations to depend on such additional factors as neighboring letters, position in sequence, whether the sequence is a (RNA) coding sequence or not, different point mutation probability matrices (e.g. Fig. 2) on different branches, etc. In MDL approach, the question to be answered is whether the additional information required to describe these extra probabilities is paid for in improved predictive power. This question is answered by seeing if the additional probabilities do indeed produce lower MDL than without the additional probabilities.

Extending the tree building to allow for multiple parents (a graph) is more challenging because the search combinatorics are greater. Constructing a MDL that reflects the fact that some proteins evolved by combining domains from very different parent proteins is simple in principle. The code must state which parts came from which parents and how they are ordered, in addition to the usual mutation events. A successful graph building program must rely on efficient methods for identifying potential building blocks (domains).

Proteins provide a more interesting possibility for MDL—predicting secondary or higher structure from sequence information and information from proteins whose structure has been determined. The basic idea is that for proteins, instead if just hypothesising particular ancestor sequences, these sequences are segmented into typed regions, such as α-helix, β-turn, etc. From a MDL perspective, the question is "is the information required to describe these typed regions paid for by the information required to describe the sequence data given the regions?". For example, the statistics of particular amino acids at particular locations in a type of β-turn could provide a reduced encoding. If some proteins in a particular tree have known structure, then this greatly enhances the certainty of inferred ancestral structure, since secondary structure is strongly conserved.

Summary

This paper has described the basic theory of MDL applied to the problem of evolutionary tree reconstruction from sequence data. Also, a particular search method for finding MDL trees was outlined. This heuristic search procedure finds a tree as close as possible to the "true" tree, but is unlikely to find the globally optimal tree. Experiments with the algorithm on test data showed that it captures all the broad families in the true tree, and has a message length close to the optimum. These experiments are just the first step in applying MDL to the problem of evolutionary reconstruction, and various extensions to the theory are readily adapted to this MDL framework.

References

[1]Allison, Wallace and Yee, "Inductive Inference over Macrommolecules", in these proceedings.

[2]Felsenstein, J., "Phylogenies from molecular sequences: Inference and reliability". *Annu. Rev. Genet. 1988*, Vol. 22, pp. 512–565.

[3]Jerzy J. and Smith T., "A fundamental division in the *Alu* family of repeated sequences." *Proc. Natl. Acad. Sci. USA*, Vol. 85, pp. 4775–4778, July 1988.