doi:10.1006/jmbi.2001.5405 available online at http://www.idealibrary.com on IDEAL® *J. Mol. Biol.* (2002) **317**, 753–764

JMB

An Expectation Maximization Algorithm for Training Hidden Substitution Models

I. Holmes* and G. M. Rubin

Howard Hughes Medical Institute, University of California, Berkeley, CA 94720, USA

*Corresponding author

We derive an expectation maximization algorithm for maximum-likelihood training of substitution rate matrices from multiple sequence alignments. The algorithm can be used to train hidden substitution models, where the structural context of a residue is treated as a hidden variable that can evolve over time. We used the algorithm to train hidden substitution matrices on protein alignments in the Pfam database. Measuring the accuracy of multiple alignment algorithms with reference to BAliBASE (a database of structural reference alignments) our substitution matrices consistently outperform the PAM series, with the improvement steadily increasing as up to four hidden site classes are added. We discuss several applications of this algorithm in bioinformatics.

Keywords: molecular evolution; bioinformatics; amino acid substitution rates; Markov models

Introduction

Substitution models for DNA, RNA and protein sequences are used widely in sequence analysis. An early application of these models was in molecular evolution, to estimate divergence times between related genes and species and make phylogenetic trees.[1] Scoring matrices derived from such models, including the PAM series,[2] are used routinely in sequence homology searches. The most familiar substitution models are continuous-time finite-state Markov chains,[3] where the instantaneous rate of substitution from residue i to residue j is given by R_{ij}, independently of the substitution history.

Such PAM-style models have experimental and theoretical deficiencies. For example, the BLOSUM series of score matrices, collected empirically from alignments binned by percentage identity,[4] are not well-modeled by any single set of rates R_{ij}.[5] A clear deficiency is that the model treats evolution as homogeneous along the sequence, ignoring the (possibly changing) structural context of selection.

In contrast, profile-based models such as hidden Markov models (HMMs) are heterogeneous: different sites can experience different selective pressure. The selective pressure acting on each site is estimated from data, e.g. by counting residue frequencies in each column of an alignment. This effectively permits an infinite variety of different structural contexts. Less commonly, the model may force sites to choose from a finite, representative set of contexts.[6,7] Applications of such models in sequence analysis include well-defined probabilistic searches such as HMMs and stochastic context-free grammars (SCFGs)[5] as well as heuristic methods like PSI-BLAST.[8] Few of these models incorporate phylogenetic relationships properly; a notable exception is the RIND program,[9,10] which has a fully phylogenetic heterogeneous substitution model.

We consider here the problem of parameterising a substitution model with a finite number of hidden states that can be associated with each residue, representing the site's biophysical context.†[11,12] This is closely related to the problem of training models, such as RIND, that use a unique rate matrix for each site. Parameterisation is a critical issue, complicated by the fact that the record of the substitution history of a site is incomplete: only the final episodes of the process, the sequences at the leaves of the phylogenetic tree, are observed.

There is an effective algorithm for training models from incomplete datasets, known as expectation maximization (EM).[13] A special case of EM, called the Baum-Welch algorithm, is ubiquitously used to train profile HMMs with Dirichlet mixture priors.[14] Another EM variant is the Inside-Outside algorithm, a generalisation of Baum-Welch used to

† http://psb.stanford.edu/

Abbreviations used: HMM, hidden Markov model; SCFG, stochastic context-free grammar; EM, expectation maximisation; PAM, point accepted mutation.

E-mail address of the corresponding author: ihh@fruitfly.org

0022-2836/02/050753–12 $35.00/0

train SCFG profiles for RNA.[5] EM has been applied to train constrained substitution models where the rate R_{ij} is a function of j only.[9] In another setting, the modeling of membrane-spanning ion channels, a numerical approach was given for the EM estimation of transition rates between different conductivity states.[15] Very recently, the EM algorithm has been applied to phylogenetic reconstruction.[16]

In the next section, we derive a fast, analytic version of the EM algorithm for maximum likelihood training of substitution models. We show that sufficient statistics for solving the EM optimisation are (i) the expected composition of the root node, (ii) the expected number of $i \to j$ substitutions and (iii) the expected time spent in state i. We give analytic expressions for these statistics for pairwise alignments (i.e. single-branch trees). We extend the calculation to multiple alignments by treating the tree as a Bayesian network and applying Pearl's belief propagation algorithm.[17]

The rate matrix EM algorithm described here is flexible, allowing for variants using prior distributions and/or alternative parameterisations. We implemented several of these variants in freely available software and used this software to train hidden substitution models on domain alignments from the Pfam database.[18] We evaluated these models for multiple alignment using the BAliBASE database of structural alignments. These results are presented and discussed in Results.

Finally, in Discussion, we mention some other applications of our method, including estimation of substitution rates of covariant sites in RNA, and discuss possible future directions for this work.

Algorithm

Consider a Markov chain with m states and parameters $\vartheta = \{\mathbf{R},\pi\}$. The transition rate matrix is $\mathbf{R}$ and the initial state distribution is π, a row vector. If the chain is initially at equilibrium, then $\pi\mathbf{R} = \mathbf{0}$.

If $p_i(t)$ is the probability of being in state i at time t (with $\mathbf{p}$ also a row vector), then the time-evolution of the chain is described by the matrix differential equation:

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = \mathbf{pR}, \quad \mathrm{p}(0) = \pi$$

which has the solution $\mathbf{p}(t) = \pi\ \mathbf{M}(t)$, where $\mathbf{M}(t)$ is the matrix exponential $\mathbf{M}(t) = \mathrm{e}^{\mathbf{R}t}$.

In practise, $\mathbf{M}(t)$ is evaluated using the diagonal form of $\mathbf{R}$. If the model is time-reversible, then $\mathbf{R}$ obeys detailed balance ($\pi_i R_{ij} = \pi_j R_{ji}$) and is related to a symmetric matrix $\mathbf{S}$ (by $S_{ij} = R_{ij}\sqrt{\pi_i/\pi_j}$) leading to the following result for the entries of $\mathbf{M}(t)$

$$M_{ij}(t) = \left(\frac{\pi_j}{\pi_i}\right)^{1/2} \sum_k^m v_i^{(k)} e^{\mu^{(k)}t} v_j^{(k)} \qquad (1)$$

in terms of the eigenvalues $\mu^{(k)}$ and orthonormal eigenvectors $\mathbf{v}^{(k)}$ of $\mathbf{S}$ (so that $\mathbf{Sv}^{(k)} = \mu^{(k)} \cdot \mathbf{v}^{(k)}$ and $\mathbf{v}^{(k)}\mathbf{v}^{(l)} = \delta_{kl}$, where δ_{kl} is the Kronecker delta: 1 if $k = l$, 0 otherwise).

The eigenvectors describe the independent modes of decay of the system: the more negative an eigenvalue, the faster that information encoded by the corresponding eigenvector is lost. For example, in Kimura's two-parameter model for nucleotide substitution,[19] the most negative eigenvalues are associated with eigenvectors that distinguish between bases with the same number of rings (i.e. A *versus* G, or C *versus* T) because this is exactly the information that is destroyed by transitions, the fastest kind of substitution.

We are interested in using data to find a better parameterisation $\vartheta' = \{\mathbf{R}', \pi'\}$ for the model. We show how to do this using the EM algorithm, first considering the simplest case: a single column of a pairwise alignment.

Consider two related proteins, separated by evolutionary time T (here we treat T as a ''given''). Let a and b be residues observed at aligned sites in these two proteins. We suppose that a is the ancestor and b is the descendant.

Denote by $\mathbf{h}$ the precise substitution history during time T, i.e. the path from a to b (this history is, of course, unknown to us). Let h_t be the state of the Markov chain at time t. Note that $h_0 = a$ and $h_T = b$.

The EM algorithm[13] consists of maximising the following sum over possible histories with respect to ϑ':

$$\mathcal{Q}(\vartheta, \vartheta') = \sum_{\mathbf{h}} P(\mathbf{h}|a, b, T, \vartheta) \log P(\mathbf{h}|T, \vartheta') \qquad (2)$$

Intuitively, this corresponds to first fixing the posterior distribution of the missing data $P(\mathbf{h}|a, b, T, \vartheta)$ given the current set of parameters ϑ, then choosing a new set of parameters ϑ' that maximises the expected log likelihood of the data according to this distribution. Given that the relative entropy of any two distributions is non-negative, it can be shown that the new parameters ϑ' are always at least as good a model as ϑ, with equality holding when the likelihood is locally maximal.

We can rewrite the second term of equation (2) as an integral over infinitesimal timesteps $\mathrm{d}t$:

$$\log P(\mathbf{h}|T, \vartheta') = \log P(h_0|\vartheta) + \int_{t=0}^{T} \log P(h_{t+\mathrm{d}t}|h_t, \vartheta')$$

and thus, since the probability of going from state i to state j in time $\mathrm{d}t$ is $\delta_{ij} + R_{ij}\,\mathrm{d}t$, we have:

$$\begin{aligned}\mathcal{Q}(\vartheta, \vartheta') &= \sum_{\mathbf{h}} P(\mathbf{h}|a, b, T, \vartheta) \log P(h_0|T, \vartheta') \\ &\quad + \int_{t=0}^{T} \sum_{\mathbf{h}} P(\mathbf{h}|a, b, T, \vartheta) \log P(h_{t+\mathrm{d}t}|h_t, \vartheta') \\ &= \sum_i P(h_0 = i|a, b, T, \vartheta) \log \pi'_i \\ &\quad + \int_{t=0}^{T} \sum_i \sum_j P(h_t = i, h_{t+\mathrm{d}t} = j|a, b, T, \vartheta) \\ &\quad \log(\delta_{ij} + R'_{ij}\mathrm{d}t)\end{aligned}$$

Now:

$$\begin{aligned}&P(h_t = i, h_{t+\mathrm{d}t} = j|a, b, T, \vartheta) \\ &\qquad = \frac{M_{ai}(t) \times (\delta_{ij} + R_{ij}\mathrm{d}t) \times M_{jb}(T-t)}{M_{ab}(T)}\end{aligned}$$

and:

$$\log(\delta_{ij} + R'_{ij}\mathrm{d}t) =$$

$$\begin{cases} R'_{ii}\mathrm{d}t & \text{if } i = j \text{ (by Taylor expansion)} \\ \log R'_{ij} + \log\ \mathrm{d}t & \text{if } i \neq j \end{cases}$$

Note that $\log\ \mathrm{d}t = -\infty$. Thus the log-likelihood of any particular history, and consequently the expected log-likelihood over the posterior distribution of histories, is minus infinity, since the probability of an event happening in a time-interval Δt tends to zero as $\Delta t \to 0$. However, the log $\mathrm{d}t$ term is independent of ϑ', and so it will disappear when we differentiate with respect to ϑ' to find the maximum of $\mathcal{Q}$. For convenience, we drop the log $\mathrm{d}t$ now, obtaining:

$$\begin{aligned}\mathcal{Q}(\vartheta, \vartheta') &= \sum_i^m \hat{s}_i \log \pi'_i + \sum_i^m \hat{w}_i R'_{ii} \\ &\quad + \sum_i^m \sum_{j \neq i}^m \hat{u}_{ij} \log R'_{ij}\end{aligned} \tag{3}$$

with the intepretation that $\hat{s}_i$ is the expected number of paths that start in state i, $\hat{w}_i$ is the expected wait in state i (i.e. the amount of time spent in i) and $\hat{u}_{ij}$ is the expected usage of transition $i \to j$.

These expectations are defined to be:

$$\begin{aligned}\hat{s}(a, b, T) &= \delta_{ia} \\ \hat{w}_i(a, b, T) &= \frac{1}{M_{ab}(T)} \mathcal{I}_{ii}^{ab}(T) \\ \hat{u}_{ij}(a, b, T) &= \frac{R_{ij}}{M_{ab}(T)} \mathcal{I}_{ij}^{ab}(T)\end{aligned} \tag{4}$$

where:

$$\begin{aligned}\mathcal{I}_{ij}^{ab}(T) &= \int_{t=0}^{T} M_{ai}(t) M_{jb}(T-t) \mathrm{d}t \\ &= \left(\frac{\pi_i \pi_b}{\pi_a \pi_j}\right)^{1/2} \sum_k^m v_a^{(k)} v_i^{(k)} \sum_l^m v_j^{(l)} v_b^{(l)} \mathcal{J}_{kl}(T)\end{aligned}$$

with:

$$\mathcal{J}_{kl}(T) = \begin{cases} T \exp(\mu^{(k)} T) & \text{if } \mu^{(k)} = \mu^{(l)} \\ \dfrac{\exp(\mu^{(k)} T) - \exp(\mu^{(l)} T)}{\mu^{(k)} - \mu^{(l)}} & \text{if } \mu^{(k} \neq \mu^{(l)} \end{cases}$$

by equation (1). The $\mathcal{J}_{kl}$ give the interactions between pairs of decay modes over the time-interval.

We want to maximise $\mathcal{Q}$ while ensuring that $\mathbf{R}$ is a valid rate matrix (i.e. $\Sigma_j R_{ij} = 0$) and π is a normalised probability vector (i.e. $\Sigma_i \pi_i = 1$).

Introducing these constraints *via* Lagrange multipliers $\{\alpha, \beta\}$, the function to be maximised is:

$$\begin{aligned}\mathcal{L}(\vartheta, \vartheta', \alpha, \beta) &= \sum_i^m \hat{s}_i \log \pi'_i + \sum_i^m \hat{w}_i R'_{ii} \\ &\quad + \sum_i^m \sum_{j \neq i}^m \hat{u}_{ij} \log R'_{ij} \\ &\quad + \sum_i^m \alpha_i \sum_j^m R'_{ij} + \beta \left(\sum_i^m \pi'_i - 1 \right)\end{aligned} \tag{5}$$

Differentiating equation (5) with respect to ϑ', α_i and β, then equating the derivatives with zero, leads to a system of simultaneous equations:

$$\begin{aligned}\partial\mathcal{L}/\partial\pi'_i &= \hat{s}_i/\pi'_i + \beta &&= 0\ \forall i \\ \partial\mathcal{L}/\partial R'_{ii} &= \hat{w}_i + \alpha_i &&= 0\ \forall i \\ \partial\mathcal{L}/\partial R'_{ij} &= \hat{u}_{ij}/R'_{ij} + \alpha_i &&= 0\ \forall i, j \neq i \\ \partial\mathcal{L}/\partial\alpha_i &= \sum_j R'_{ij} &&= 0\ \forall i \\ \partial\mathcal{L}/\partial\beta &= \sum_i \pi'_i - 1 &&= 0\end{aligned}$$

with the solution:

$$\begin{aligned}R'_{ij} &= \frac{\hat{u}_{ij}}{\hat{w}_i} && \text{for all } i, j \neq i \\ R'_{ii} &= -\sum_{j \neq i} \frac{\hat{u}_{ij}}{\hat{w}_i} && \text{for all } i \\ \pi'_i &= \frac{\hat{s}_i}{\sum_j \hat{s}_j} && \text{for all } i\end{aligned} \tag{6}$$

i.e. the optimal transition rate from i to j is the expected number of $i \to j$ transitions divided by the expected time spent in i, whereas the optimal initial distribution π is just the normalised form of $\hat{s}$, the expected distribution of the root state.

Alternative parameterisations and priors

It is straightforward to derive variations of the EM algorithm for cases when the model depends on a reduced parameter set. In such situations we simply insert the relevant expressions for R_{ij} and π_j into equation (5).

For example, we can set $R_{ij} \propto \pi_j$, implying that the substitution rate from i to j is proportional to the equilibrium frequency of j (independent of i). This rate matrix is reversible, by definition. Maximising equation (5) then gives a solution similar to the EM algorithm devised by Bruno for the RIND program.[9] Another reduced parameterisation is given below (see Hidden classes).

We can easily incorporate prior distributions by adding a term of the form $\log P(\vartheta')$ to equation (5) (strictly speaking, it should have been in equation (2) to begin with). Appropriate use of Dirichlet and exponential priors amounts to adding pseudo-counts to $\hat{s}_i$ and $\hat{u}_{ij}$ and a ''pseudo-time'' to $\hat{w}_i$.

Extension to multiple sequences

So far, we have considered only the evolution of a single site in two related proteins. We can extend the training algorithm to include information from multiple proteins related by a tree. We do this by adding up the separate contributions of each branch to $\hat{\mathbf{s}}$, $\hat{\mathbf{w}}$ and $\hat{\mathbf{u}}$.

The phylogenetic tree is a directed graphical model, or Bayesian network. The marginal probability of any particular parent-child pair being in the configuration (a, b) can be calculated using Pearl's belief propagation algorithm as follows.[17]

Suppose $\mathcal{T}_N$ is a tree with N nodes, sorted children-before-parents (so that node N is the root). For any node n, let the set of its children be $\mathcal{C}_n$ and let (p, g) be the parent and grandparent of n. Let the branch length from node m to node n be t_{mn} (the evolutionary timespan separating m and n).

Imagine for a minute that we knew in what state the Markov chain had been at every node of the tree. Then we could write ϕ_n for the actual state at node n (which must be equal to the observed state, if n is a leaf node) and:

$$\Phi(\mathcal{T}) = \{\phi_{n'} : n' \in \mathcal{T}\}$$

for the collective set of states at all the nodes n' in some subtree $\mathcal{T}$ (a subtree is a subset of the nodes, together with all their connecting branches). We refer to $\Phi(\mathcal{T})$ as an allowed history of subtree $\mathcal{T}$. Of course, $\Phi(\mathcal{T})$ is really missing data, and we intend to sum it out of the likelihood.

Denote by $\mathcal{T}_n$ the subtree containing node n and all its descendants. Define:

$$\mathrm{U}_b^{(n)} = \sum_{\Phi(\mathcal{T}_n)} P(\Phi(\mathcal{T}_n)|\phi_n = b, \vartheta)$$

i.e. the likelihood of all the observed data in $\mathcal{T}_n$, conditioned on the chain being in state b at node n, and summed over all allowed histories. Note that, at leaf nodes, $\mathrm{U}_b^{(n)} = 0$ if b is incompatible with the observed state j at that node.

The $\mathbf{U}^{(n)}$ may be computed recursively:

$$\mathrm{U}_b^{(n)} = \begin{cases} \delta_{bj} & \text{if } n \text{ is a leaf node in state } j \\ \prod_{c \in \mathcal{C}_n} \sum_k M_{bk}(t_{nc}) \mathrm{U}_k^{(c)} & \text{otherwise} \end{cases} \tag{7}$$

which is Felsenstein's algorithm.[1] The $\mathbf{U}^{(n)}$ are computed in ascending order, i.e. starting with $\mathbf{U}^{(1)}$ and ending with $\mathbf{U}^{(N)}$. The full likelihood is:

$$P(\mathbf{x}|\vartheta) = \sum_b \pi_b \mathrm{U}_b^{(N)}$$

where $\mathbf{x}$ represents the observed data at all leaf nodes.

Now let $\overline{\mathcal{T}_n}$ be the complement of $\mathcal{T}_n$, i.e. all nodes except n and its descendants. For $n < N$, $\overline{\mathcal{T}_n}$ contains at least one member: p, the parent of n. Define:

$$D_a^{(n)} = \sum_{\Phi(\overline{\mathcal{T}_n})} P(\Phi(\overline{\mathcal{T}_n}), \phi_p = a|\vartheta)$$

i.e. the likelihood of all the observed data not in $\mathcal{T}_n$, including the state of the chain a at node p, summed over all allowed histories. (Note, however, that this time we do not condition on the state of the chain at node p; it is included in the probability.) Again, there is a recursive method:

$$D_a^{(n)} = \left(\prod_{\substack{c \in \mathcal{C}_p, \\ c \neq n}} \sum_j M_{aj}(t_{pc}) \mathrm{U}_j^{(c)} \right) \times \begin{cases} \pi_a & \text{if } p \text{ is root} \\ \sum_i D_i^{(p)} M_{ia}(t_{gp}) & \text{otherwise} \end{cases} \tag{8}$$

where the $\mathbf{D}^{(n)}$ are computed in descending order. (By analogy with the forward-backward algorithm we nickname this the up-down algorithm, although, as noted previously, it is really just an application of belief propagation.)

As with the forward-backward algorithm, we can use $\mathbf{U}$ and $\mathbf{D}$ to calculate the posterior probability $q_{ab}^{(n)}$ of a parent-child pair (p, n) being in states (a, b):

$$
\begin{aligned}
q_{ab}^{(n)} &= P(\phi_p = a, \phi_n = b|\mathbf{x}, \vartheta) \\
&= \frac{P(\phi_p = a, \phi_n = b, \mathbf{x}|\vartheta)}{P(\mathbf{x}|\vartheta)} \\
&= \frac{1}{P(\mathbf{x}|\vartheta)} \sum_{\Phi(\mathcal{T}_N)} P(\Phi(\mathcal{T}_N), \phi_p = a, \phi_n = b|\vartheta) \\
&= \frac{1}{P(\mathbf{x}|\vartheta)} \left(\sum_{\Phi(\overline{\mathcal{T}}_n)} P(\Phi(\overline{\mathcal{T}}_n), \phi_p = a|\vartheta) \right) \\
&\quad \times P(\phi_n = b|\phi_p = a, \vartheta) \\
&\quad \times \left(\sum_{\Phi(\mathcal{T}_n)} P(\Phi(\mathcal{T}_n)|\phi_n = b, \vartheta) \right) \\
&= \frac{D_a^{(n)} M_{ab}(t_{pn}) U_b^{(n)}}{P(\mathbf{x}|\vartheta)}
\end{aligned}
$$

and the posterior probability $q_b^{(N)}$ of the root node N being in state b:

$$
q_b^{(N)} = P(\phi_N = b|\mathbf{x}, \vartheta) = \frac{\pi_b U_b^{(N)}}{P(\mathbf{x}|\vartheta)}
$$

We can use these probabilities to obtain multiple-branch estimates for $\hat{\mathbf{s}}$, $\hat{\mathbf{w}}$ and $\hat{\mathbf{u}}$:

$$
\begin{aligned}
\hat{s}_i^{(\text{multi})} &= q_i^{(N)} \\
\hat{w}_i^{(\text{multi})} &= \sum_{n=1}^{N-1} \sum_a \sum_b q_{ab}^{(n)} \hat{w}_i(a, b, t_{pn}) \\
\hat{u}_{ij}^{(\text{multi})} &= \sum_{n=1}^{N-1} \sum_a \sum_b q_{ab}^{(n)} \hat{u}_{ij}(a, b, t_{pn})
\end{aligned}
$$

with the single-branch estimates $\hat{w}_i(a, b, t_{pn})$ and $\hat{u}_i(a, b, t_{pn})$ defined as in equation (4). Substituting in these single-branch definitions, then rearranging, leads us to:

$$
\begin{aligned}
\hat{s}_i^{(\text{multi})} &= q_i^{(N)} \\
\hat{w}_i^{(\text{multi})} &= \sum_k v_i^{(k)} \sum_l v_i^{(l)} \mathcal{C}_{kl} \\
\hat{u}_{ij}^{(\text{multi})} &= S_{ij} \sum_k v_i^{(k)} \sum_l v_j^{(l)} \mathcal{C}_{kl}
\end{aligned}
\tag{9}
$$

where $\mathcal{C}$ is a matrix of eigenbasis counts:

$$
\mathcal{C}_{kl} = \frac{1}{P(\mathbf{x}|\vartheta)} \sum_{n=1}^{N-1} \mathcal{D}_k^{(n)} \mathcal{J}_{kl}(t_{pn}) \mathcal{U}_l^{(n)} \tag{10}
$$

where $\mathcal{D}^{(n)}$ and $\mathcal{U}^{(n)}$ denote left and right eigenbasis projections of $\mathbf{D}^{(n)}$ and $\mathbf{U}^{(n)}$:

$$
\begin{aligned}
\mathcal{D}_k^{(n)} &= \sum_a^m D_a^{(n)} v_a^{(k)} \pi_a^{-\frac{1}{2}} \\
\mathcal{U}_l^{(n)} &= \sum_b^m U_b^{(n)} v_b^{(l)} \pi_b^{\frac{1}{2}}
\end{aligned}
\tag{11}
$$

Thus, a rate matrix can be trained on a database of multiple alignments by summing $\mathcal{C}$ over every column in every alignment, inserting the multiple-branch estimates (equations (9)) into equation (6) and iterating.

Intuitively, the **D** and **U** vectors give the probability distribution of residues at internal nodes, the $\mathcal{D}$ and $\mathcal{U}$ vectors give the same probability distributions resolved onto the rate eigenvectors and the $\mathcal{C}$ matrix counts the ''eigensubstitutions'', i.e. the substitutions resolved onto the rate eigenvectors, or (conceptually) the interactions between the various decay modes. After these counts have been collected, $\mathcal{C}$ is transformed back from the eigenbasis to the residue basis to yield the number of times each residue substitution occurred. It is more efficient to do things this way, deferring the back-transformation step until the end and counting eigensubstitutions rather than residue substitutions during the main loop, even though intuition is most comfortable with residue substitutions.

Imposing reversibility

We can impose reversibility by using the following symmetrised form of equation (10):

$$
\mathcal{C}_{kl} = \frac{1}{P(\mathbf{x}|\vartheta)} \sum_{n=1}^{N-1} \mathcal{J}_{kl}(t_{pn}) (\mathcal{D}_k^{(n)} \mathcal{U}_l^{(n)} + \mathcal{U}_k^{(n)} \mathcal{D}_l^{(n)})/2
$$

We can also impose the initial-equilibrium condition, while making use of the root-node estimates $\hat{\mathbf{s}}$, by considering the most recent substitution $a \to b$ occuring at some time T before the root (see Figure 1). Given b, the posterior probability of a is $-R_{ba}/R_{bb}$ (assuming $\mathbf{R}$ is reversible) and the posterior expectation of T is $-1/R_{bb}$. (Note that R_{bb} is negative, hence the minus signs.) Thus, we can make revised, ''equilibrated'' estimates $\hat{\mathbf{w}}$ and $\hat{\mathbf{u}}$ that incorporate $\hat{\mathbf{s}}$ as follows:

$$
\begin{aligned}
\hat{w}_b^{(\text{multi})} &\leftarrow \hat{w}_b^{(\text{multi})} - \frac{\hat{s}^{(\text{multi})}}{R_{bb}} \\
\hat{u}_{ab}^{(\text{multi})} &\leftarrow \hat{u}_{ab}^{(\text{multi})} - \frac{R_{ba} \hat{s}_b^{(\text{multi})}}{R_{bb}}
\end{aligned}
$$

Hidden classes

Up until now, we have assumed that each residue corresponds to a single state. This model can be generalised to have C available states for each observed residue. The actual state for a residue

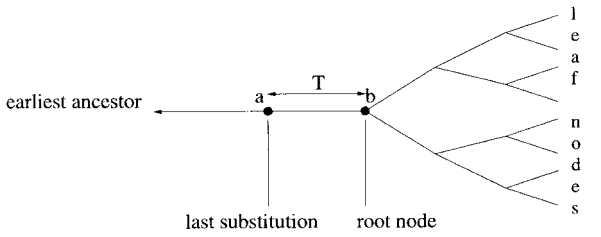

Figure 1. Counting the initial state at the root node as a pseudo-substitution (see Algorithm, section Imposing reversibility). Let the state of the root node be b, and suppose there is a hypothetical branch extending back in time from the root node. Consider the most recent substitution on this branch; let the time from this substitution to the root node be T, and the state of the chain prior to this substitution be a (with $a \neq b$). Then we can count the initial state as a pseudo-substitution ($a \to b$) with a wait time T.

observed at a particular site is a hidden variable representing the class of that site.

For example, consider a globular cytoplasmic protein. Buried residues in such proteins display significantly different patterns of substitution than those of solvent-exposed residues.[20] In the notation of this section, sites in this protein could be modeled using two classes, so that $C = 2$. The two-class amino acid alphabet contains 40 symbols, including two types of alanine (A1 and A2; one buried and one exposed), two types of arginine (R1, R2), and so on. Any observed alanine residue in a sequence can potentially be either A1 or A2; the 1 or 2 is the class of the residue, which is hidden from view but may be inferred from the substitution patterns of the site.

Such a model has been described as a hidden substitution model.[11,12] Technically, it is a hidden Markov model, but this term is avoided due to the special meaning it has in bioinformatics.[5]

In terms of the stochastic model, each symbol corresponds to a state, as before (so, in the above example, there would be 40 states). For a state a, let $\rho(a)$ represent the observed residue for that state (in the above example, $\rho(\mathrm{A1}) = \rho(\mathrm{A2}) = \mathrm{A}$) and let $\gamma(a)$ represent the hidden class (so $\gamma(\mathrm{A1}) = 1$ and $\gamma(\mathrm{A2}) = 2$).

The EM algorithm described remains valid; the only qualification is as follows. It was stated above that the up likelihoods $U_b^{(n)}$ are, by definition, zero if n is a leaf node whose observed residue j is incompatible with state b. For simple substitution models, this means that $U_b^{(n)} = \delta_{bj}$ at leaf nodes.

This principle holds for hidden substitution models, except that there is now more than one state b compatible with j. Thus we now have $U_b^{(n)} = \delta_{\rho(b)j}$ for leaf nodes.

To avoid overfitting, we use a reduced parameter space, setting $R_{ab} = 0$ unless $\rho(a) = \rho(b)$ or $\gamma(a) = \gamma(b)$. In other words, a site cannot change both its residue and its class in the same instant.

Results

We trained models using 200 domain alignments selected at random from the May 19, 2001 release of the Pfam database.[18] We used a further 200 alignments from Pfam as a test set (having no overlap with the training set). We conditioned the training on phylogenetic trees created by the weighbor program from distance matrices that we generated using the PAM (point accepted mutation) substitution model.[2]

We used three different training procedures.

Unguided. Starting from an initially random seed matrix, we ran the EM algorithm (see Algorithm, above).

Restricted. Starting from the same random seed as with the Unguided procedure, we ran the EM algorithm, with the constraint that the substitution rate from i to j, R_{ij}, depends only on the new residue j (see Alternative parameterisations and priors, above).

Guided. Using the matrix generated by the Restricted procedure as a seed matrix, we again ran the EM algorithm unconstrained.

Table 1. Test set log-likelihoods for EM-trained hidden substituion models with one to four site classes (C)

		Test set log-likelihood (bits)		
C	Procedure	Seed 1	Seed 2	Seed 3
1	Unguided	−1.9223e + 06	−1.9223e + 06	−1.9223e + 06
	Restricted	−2.033e + 06	−2.033e + 06	−2.033e + 06
	Guided	−1.9223e + 06	−1.9223e + 06	−1.9223e + 06
2	Unguided	−1.8972e + 06	−1.8987e + 06	−1.899e + 06
	Restricted	−1.9774e + 06	−1.9775e + 06	−1.9774e + 06
	Guided	−1.8932e + 06	−1.8933e + 06	−1.8933e + 06
3	Unguided	−1.8939e + 06	−1.8932e + 06	−1.8931e + 06
	Restricted	−1.959e + 06	−1.9591e + 06	−1.9724e + 06
	Guided	−1.8891e + 06	−1.8893e + 06	−1.8916e + 06
4	Unguided	−1.8948e + 06	−1.8926e + 06	−1.894e + 06
	Restricted	−1.9574e + 06	−1.9574e + 06	−1.9572e + 06
	Guided	−1.8887e + 06	−1.889e + 06	−1.8889e + 06

The procedures are described in Results.

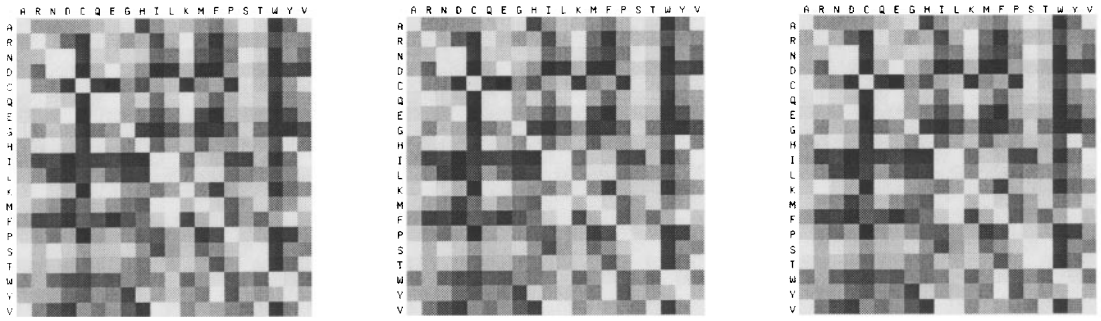

Figure 2. The results of three consecutive training runs of a single-class substitution model (effectively having no hidden states) using the Unguided training procedure. Light squares indicate relatively high substitution rates. The matrices are consistent and give similar likelihoods for the test set, which consisted of a random 200 alignments from Pfam (May 19, 2001 release). The training set comprises a different random 200 Pfam alignments. A different randomised seed matrix was used for each training run.

We repeated each procedure with the number of hidden classes C set to 1, 2, 3 and 4 (note that $C = 1$ corresponds to the simple model with just one class, which thus is effectively not hidden). We carried out this entire procedure three times, starting from three different random seed matrices.

The log-likelihoods of the test set following these training runs are shown in Table 1. For a single class ($C = 1$), the choice of initial random seed does not appear to make much difference, but for $C > 1$ the Unguided procedure becomes increasingly sensitive to the seed. The Guided procedure, however, is stable even for $C > 1$, and returns consistently higher likelihoods than the Unguided procedure. The Restricted likelihoods are lower than the Unguided likelihoods, but they are more reproducible, in keeping with the Restricted procedure using considerably fewer parameters. Using the Restricted-trained models as seeds for the Guided procedure appears both to add stability and to improve the test likelihood.

Some examples of models trained using different procedures are shown in Figures 2-6. The two classes learned by the $C = 2$ models appear consistent with substitution patterns for ''buried'' and ''exposed'' sites (Figures 3 and 4). The buried class favours hydrophobic side-chains (specifically A, I, L, M and V) and has slow intra-class substitution rates, while the exposed class favours charged and polar molecules and has faster rates. In addition to the buried and exposed categories identified above, the three-class (Figure 5) and four-class (Figure 6) models find a third ''tiny'' category favouring alanine, glycine and serine (all of which have very small side-chains).

To evaluate some potential benefits of hidden substitution models for sequence analysis, we used the substitution models trained using the Guided procedure as parameters for doing multiple

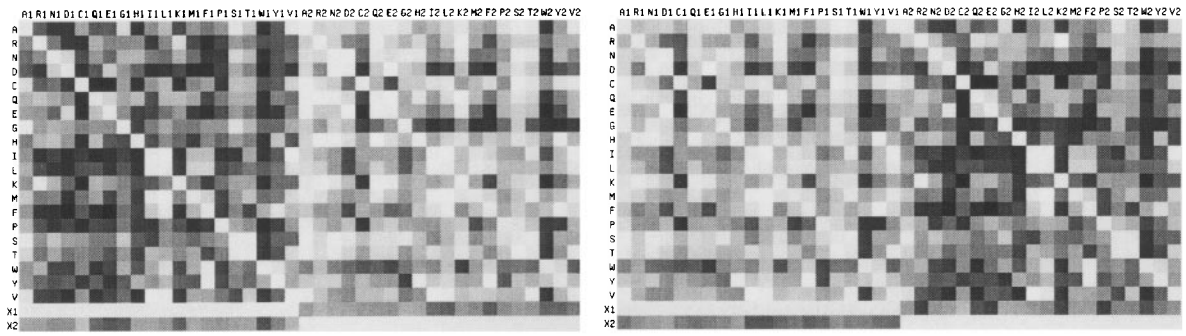

Figure 3. The results of two consecutive training runs of a two-class substitution model using the Unguided training procedure. The main part of the matrices represent intra-class substitutions, while the lower two rows (labelled X1, X2) represent inter-class substitutions. The classes reflect buried and exposed sites, although no prior knowledge of protein structure was supplied during training. The class assignments have flipped between the two models, due to undecidability (the random seed matrix is the only factor determining which class ends up being buried and which exposed). Apart from the flip, the results are similar but show subtle differences, apparent as variations in the likelihood score for the test set (Table 1): training is suboptimal and sensitive to the initial random seed.

Table 2. BAliBASE alignment categories

Subdirectory	Description
ref1/test1	Equidistant, similar lengths; high identity (>35%)
ref1/test2	Equidistant, similar lengths; medium identity (20%-40%)
ref1/test3	Equidistant, similar lengths; low identity (<25%)
ref2/test1	Highly-related family (>25%) plus ''orphan'' outliers (<20%)
ref3/test	Equidistant divergent subfamilies (<20% between subfamilies)
ref4/test	N/C terminal extensions
ref5/test	Insertions

sequence alignments of sequences in BAliBASE, a benchmark database of multiple alignments constructed using crystallographic structures.[21] BAliBASE comprises several categories of multiple alignment designed to test various aspects of multiple alignment categories (Table 2). The BAliBASE authors define several metrics of alignment accuracy; the score we use is the proportion of correctly aligned residue pairs in the test alignment.

For alignment software, we used Handel,[22] a package for doing reverse Bayesian inference (i.e. multiple alignment) on stochastic process models of sequence evolution where the indel model is the single-residue birth-death process due to Thorne *et al*.[23] Handel includes several alignment algorithms, including impatient-progressive alignment (a single pass up through the tree, estimating profiles of ancestral sequences by aligning siblings) and greedy-refined alignment (multiple iterated passes through the tree, stopping when there are no more improvements to be found). Greedy-refined alignment generally gives better results than impatient-progressive alignment, but is slower.

Accuracy results for both greedy-refined and impatient-progressive algorithms are given in Tables 3 and 4, respectively. Accuracy scores for the PAM substitution model are shown for comparison.

Overall, there is a steady improvement as C is increased, corresponding to about one added percentage point of accuracy per site class. The gain in performance is most notable for BAliBASE category ref3/test, which contains families of sequences related by trees with a long internal branch. This suggests that the use of hidden variables to model the site class addresses the inadequacy of simple substitution models over a wide range of branch lengths. By keeping track of the hidden class of a site, the model is better able to model selection effects on long branches.

All models outperform PAM for accuracy, with the four-class model correctly aligning nearly 5 % more residues than PAM. The relative improvements do not seem to depend on which alignment algorithm is used, although in absolute terms, greedy-refined alignment is about 2 % more accurate than impatient-progressive (as expected).

It is noteworthy that even the simple EM-trained model ($C = 1$) in impatient-progressive mode outperforms the PAM model in greedy-refined mode. This suggests that using EM-trained rate matrices (or adding a couple of hidden site classes) is a more efficient strategy than iterative refinement for obtaining accurate alignments.

Discussion

The EM algorithm we describe is fast, efficient and general. Without supervision, it classifies columns of multiple alignments into biologically meaningful classes and finds hidden substitution rates between residues and classes. These matrices

Table 3. Percentages of correctly aligned residue pairs for BAliBASE alignments using the Handel program in impatient-progressive alignment mode

BAliBASE subdirectory	Correctly aligned residue pairs (%)				
	PAM	$C = 1$	$C = 2$	$C = 3$	$C = 4$
ref1/test1	83.8	84.0	84.0	84.0	83.4
ref1/test2	69.3	70.8	70.8	71.3	71.3
ref1/test3	74.6	76.3	75.6	75.9	75.6
ref2/test1	85.1	86.2	86.6	87.7	86.8
ref3/test	50.7	53.9	56.2	57.8	58.8
ref4/test	30.1	31.7	29.1	28.8	34.4
ref5/test	61.6	61.8	66.8	67.1	66.6
All	63.5	65.4	66.9	67.9	68.3

The PAM substitution model is compared to models with one to four site classes trained by the Guided procedure.

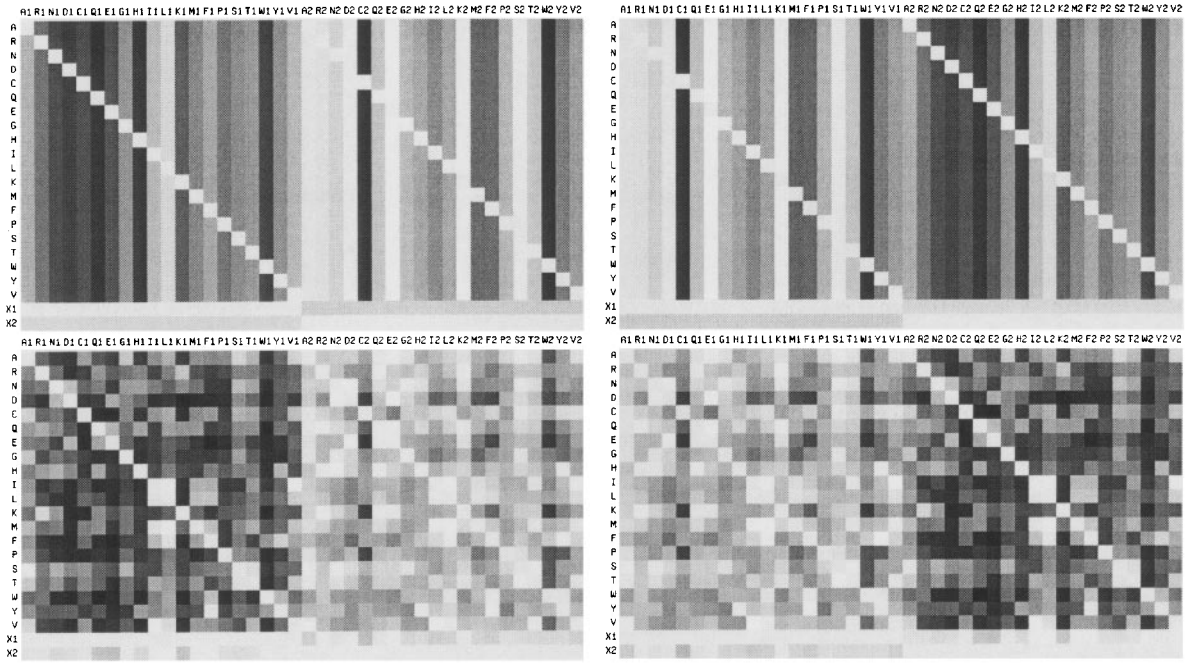

Figure 4. The results of two consecutive training runs for a two-class substitution model using the Guided training procedure. Again, the class assignments are flipped, following the same random seed matrices as Figure 3. Both models scored identically on the test set (Table 1), with a higher score than the Unguided training of Figure 3. (Top left and top right) The results of the Restricted training procedure, which is the first stage of the Guided procedure, yields matrices whose columns are constrained to have identical intra-class substitution rates (see Algorithm). (Bottom left and bottom right) The second stage of the Guided procedure is unconstrained training, yielding models similar to the Unguided models but more consistently and with better likelihoods.

lend increased accuracy to multiple alignment algorithms based on evolutionary models.

Guiding the EM algorithm, by initially constraining substitution rates to be independent of the source residue, helps it home-in on a better solution. Such an approach might be useful for on-the-fly rate estimation, where a rate matrix is estimated by the multiple alignment algorithm itself.

Our results suggest that, while a two-class model is an improvement over a simple memoryless model, a larger number of classes is needed to model protein evolution adequately. This agrees with the analogous observation for non-evolutionary sequence models, such as profile HMMs: many Dirichlet mixture components are required to provide adequate priors.[24] It is possible to combine such Dirichlet mixture priors with the EM rate-training approaches described here and by Bruno,[9] and multiple alignment methods based on stochastic indel models[22] to design algorithms for simultaneous profiling, phylogeny and alignment (I.H., unpublished results).

An interesting way to extend the substitution processes described here is to look for covarying residues. Annotated multiple alignments of RNA sequences typically specify which columns are

Table 4. Percentages of correctly aligned residue pairs for BAliBASE alignments using the Handel program in greedy-refined alignment mode

BAliBASE subdirectory	Correctly aligned residue pairs (%)				
	PAM	$C = 1$	$C = 2$	$C = 3$	$C = 4$
ref1/test1	84.7	85.1	85.9	85.5	85.1
ref1/test2	70.9	73.0	73.5	73.4	73.6
ref1/test3	75.2	77.3	77.1	77.6	77.3
ref2/test1	86.4	87.6	88.3	88.5	88.1
ref3/test	53.3	56.2	58.7	59.5	60.6
ref4/test	30.8	31.9	30.2	30.8	36.2
ref5/test	63.4	63.6	69.4	70.6	69.3
All	65.3	67.2	68.9	69.5	70.1

The PAM substitution model is compared to models with one to four site classes trained by the Guided procedure.

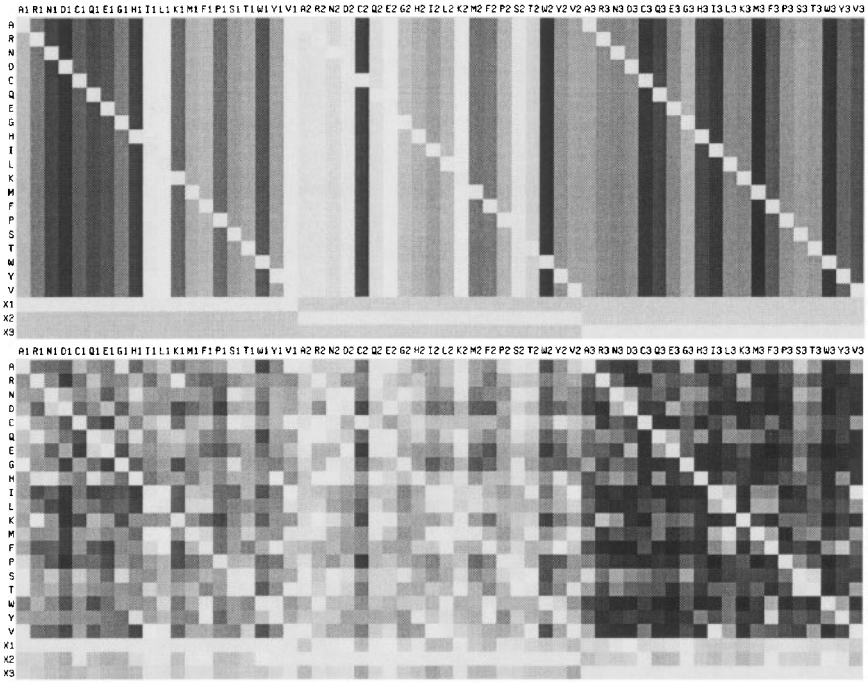

Figure 5. A three-class substitution model trained using the Guided procedure (bottom) *via* an intermediate Restricted-trained model (top).

structurally paired. The coevolving base-pairs in these columns may be modeled using a single Markov chain with 16 states (state 1 is A-A, state 2 is A-C etc. up to U-U).[25] Our EM algorithm is entirely applicable to this situation, as indeed it is to the inference of covariation in protein sequence alignments.

Many other extensions of the EM algorithm described here are possible. An EM algorithm for learning phylogenetic topologies has been

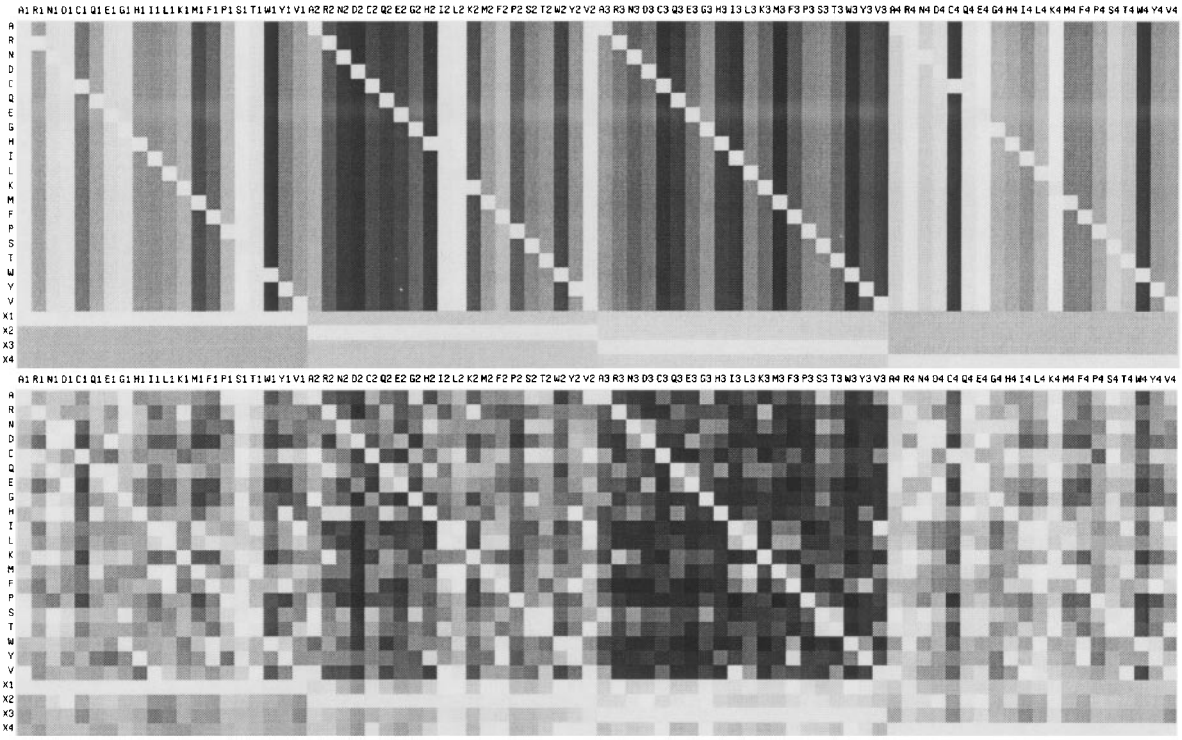

Figure 6. A four-class substitution model trained using the Guided procedure (bottom) *via* an intermediate Restricted-trained model (top).

developed by Friedman *et al*.[16] Their treatment is close to our method; in particular, the authors give versions of our equations (3), (7) and (8). The principal difference is that, while Friedman *et al*. mention the possibility of learning heterogeneous mutation rates for a sequence, their focus is on estimation of the optimal tree. A natural next step is to combine our rate EM algorithm with this tree EM algorithm, i.e. learn the mutation process and the phylogenetic history, simultaneously. We are developing an EM algorithm for estimating insertion and deletion rates in the Thorne *et al*. links model[23] (the indel model underpinning the Handel software, used here for benchmarking[22]). Recent work on ensemble learning with HMMs could be combined with our methods, allowing phylogenetic algorithms to use more information from the posterior distribution than just the single best rate matrix.† All of these algorithms can be combined to learn the optimal parameters for doing multiple alignment without any prior knowledge of the relationships between the specific sequences being aligned.

We hope that the methods and software we have developed here may be of use to people developing better evolutionary models. To this end we are distributing our code under the GNU Public License, at the URL www.biowiki.org/hsm

Acknowledgments

We gratefully acknowledge the support of the Informatics team at the Berkeley Drosophila Genome Project, particularly Erwin Frise and Amanda Dahl. The idea of evolving hidden states came from a conversation with Ewan Birney. One of us (I.H.H.) especially thanks Eliza McKenna for support and encouragement.

References

1. Felsenstein, J. (1981). Evolutionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.* **17**, 368-376.
2. Dayhoff, M. O., Schwartz, R. M. & Orcutt, B. C. (1978). A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure* (Dayhoff, M. O., ed.), vol. 5, suppl. 3, pp. 345-352, National Biomedical Research Foundation, Washington, DC.
3. Karlin, S. & Taylor, H. (1975). *A First Course in Stochastic Processes*, Academic Press, San Diego, CA.
4. Henikoff, S. & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA,* **89**, 10915-10919.
5. Durbin, R., Eddy, S., Krogh, A. & Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, Cambridge, UK.
6. Felsenstein, J. & Churchill, G. A. (1996). A hidden Markov model approach to variation among sites in rate of evolution. *Mol. Biol. Evol.* **13**, 93-104.
7. Yang, Z. (1995). A space-time process model for the evolution of DNA sequences. *Genetics,* **139**, 993-1005.
8. Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W. & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acids Res.* **25**, 3389-3402.
9. Bruno, W. J. (1996). Modelling residue usage in aligned protein sequences via maximum likelihood. *Mol. Biol. Evol.* **13**, 1368-1374.
10. Halpern, A. L. & Bruno, W. J. (1998). Evolutionary distances for protein-coding sequences: modeling site-specific residue frequencies. *Mol. Biol. Evol.* **15**, 910-917.
11. Dimmic, M. W., Mindell, D. P. & Goldstein, R. A. (2000). Modeling evolution at the protein level using an adjustable amino acid fitness model. In *Proceedings of the Fifth Pacific Symposium on Biocomputing*, pp. 18-29, Stanford University, Palo Alto, CA.
12. Koshi, J. M. & Goldstein, R. A. (2001). Analyzing rate heterogeneity during protein evolution. In *Proceedings of the Sixth Pacific Symposium on Biocomputing*, pp. 191-202, Stanford University, Palo Alto, CA.
13. Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statistical Soc. sect. B,* **39**, 1-38.
14. Brown, M., Hughey, R., Krogh, A., Mian, I. S., Sjölander, K. & Haussler, D. (1993). Using Dirichlet mixture priors to derive hidden Markov models for protein families. In *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology* (Hunter, L., Searls, D. B. & Shavlik, J., eds), pp. 47-55, AAAI Press, Menlo Park, CA.
15. Michalek, S. & Timmer, J. (1999). Estimating rate constants in hidden Markov models by the EM algorithm. *IEEE Trans. Signal Processing,* **47**, 226-228.
16. Friedman, N., Ninio, M., Pe'er, I. & Pupko, T. (2001). A structural EM algorithm for phylogentic inference. In *Proceedings of the Fifth Annual International Conference on Computational Biology* (Lengauer, T., Sankoff, D., Istrail, S., Pevzner, P. & Waterman, M., eds), pp. 132-140, Association for Computing Machinery, New York.
17. Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann Publishers, San Mateo, CA.
18. Bateman, A., Birney, E., Durbin, R., Eddy, S. R., Howe, K. L. & Sonnhammer, E. L. L. (2000). The Pfam protein families database. *Nucl. Acids Res.* **28**, 263-266.
19. Kimura, M. (1980). A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.* **16**, 111-120.
20. Branden, C. & Tooze, J. (1991). *Introduction to Protein Structure*, Garland, New York.
21. Thompson, J. D., Plewniak, F. & Poch, O. (1999). A comprehensive comparison of multiple sequence alignment programs. *Nucl. Acids Res.* **27**, 2682-2690.
22. Holmes, I. & Bruno, W. J. (2001). Evolutionary HMMs: a Bayesian approach to multiple alignment. *Bioinformatics,* **17**, 803-820.

† http://wol.ra.phy.cam.ac.uk/mackay

23. Thorne, J. L., Kishino, H. & Felsenstein, J. (1991). An evolutionary model for maximum likelihood alignment of {DNA} sequences. *J. Mol. Evol.* **33**, 114-124.
24. Sjölander, K., Karplus, K., Brown, M., Hughey, R., Krogh, A., Mian, I. S. & Haussler, D. (1996). Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *Comput. Appl. Biosci.* **12**, 327-345.
25. Knudsen, B. & Hein, J. (1999). RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics,* **15**, 446-454.

Edited by F. Cohen

(Received 13 September 2001; received in revised form 11 December 2001; accepted 10 January 2002)