BIOINFORMATICS

Vol. 19 Suppl. 1 2003, pages i147–i157
DOI: 10.1093/bioinformatics/btg1019

Using guide trees to construct multiple-sequence evolutionary HMMs

I. Holmes

Department of Statistics, University of Oxford. 1 South Parks Road, Oxford OX1 3TG, UK

Received on January 6, 2003; accepted on February 20, 2000

ABSTRACT

Motivation: Score-based progressive alignment algorithms do dynamic programming on successive branches of a guide tree. The analogous probabilistic construct is an Evolutionary HMM. This is a multiple-sequence hidden Markov model (HMM) made by combining transducers (conditionally normalised Pair HMMs) on the branches of a phylogenetic tree.

Methods: We present general algorithms for constructing an Evolutionary HMM from any Pair HMM and for doing dynamic programming to any Multiple-sequence HMM.

Results: Our prototype implementation, `Handel`, is based on the Thorne-Kishino-Felsenstein evolutionary model and is benchmarked using structural reference alignments.

Availability: `Handel` can be downloaded under GPL from `www.biowiki.org/Handel`

INTRODUCTION

Recent years have seen a surge of interest in the use of probabilistic evolutionary models for multiple sequence alignment (Thorne *et al.*, 1991; Durbin *et al.*, 1998; Hein *et al.*, 2000; Holmes and Bruno, 2001; Holmes and Rubin, 2002). The term 'statistical alignment' has been coined (Hein *et al.*, 2000) to describe this work. Amongst other advantages, the theory scales naturally to any shape or size of phylogenetic tree (Holmes and Bruno, 2001) and offers systematic approaches to problems such as parameterisation (Holmes and Rubin, 2002), homology testing (Hein *et al.*, 2000) and phylogenetic profiling (Durbin *et al.*, 1998).

Much statistical alignment work to date has used a single-residue indel model called the TKF91 model (Thorne *et al.*, 1991). The probability of observing a given pairwise alignment under the TKF91 model is described by a Pair HMM (Durbin *et al.*, 1998). In previous work, we used this TKF91 Pair HMM to develop and implement a multiple alignment package called `Handel`. In doing so, we introduced the concept of a multiple-sequence HMM (or *Multiple HMM*) and showed how a special kind of Multiple HMM, called an *Evolutionary HMM*, could be constructed by associating the TKF91 Pair HMM with each branch of a phylogenetic guide tree (Holmes and Bruno, 2001).

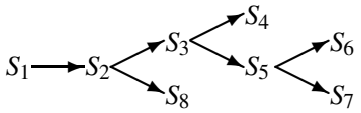

Fig. 1. A multiple alignment with a phylogenetic 'guide tree' can be represented as a Bayesian belief network. The sequences at leaf nodes (i.e. S_4, S_6, S_7 and S_8) are observed, while internal nodes (S_1, S_2, S_3 and S_5) are unobserved.

The idea of using the branches of a guide tree to combine suitable Pair HMMs (henceforth called *Branch HMMs*) follows the intuition that underpins progressive alignment tools such as CLUSTALW (Thompson *et al.*, 1994). The phylogenetic tree here is being used as a 'Bayesian belief network', specifying dependencies between observed (and unobserved) sequences (Friedman *et al.*, 2001). For example, the likelihood function for the tree shown in Figure 1 takes the form $P(S_1 \ldots S_8) = P(S_1) \times P(S_2|S_1) \times P(S_3|S_2) \times P(S_4|S_3) \times P(S_5|S_3) \times P(S_6|S_5) \times P(S_7|S_5) \times P(S_8|S_2)$, where the conditional probabilities $P(S_n|S_m)$ are given by the Branch HMM. This is in contrast to methods that align all sequences to a 'consensus' profile HMM. For such methods, the likelihood function would be $P(S_1 \ldots S_8) = \prod_{n=1}^{8} P(S_n)$, with each $P(S_n)$ given by the consensus HMM.

An Evolutionary HMM (EHMM) is not limited to the TKF91 model as its choice of Branch HMM, and can be generalised (Holmes and Bruno, 2001). The motivation for this is the eventual use of arbitrarily complex Branch HMMs so as to model phenomena such as (taking three examples from CLUSTALW) affine gaps, local alignment and penalisation of indels in hydrophobic regions.

The purpose of this paper is to formalise the EHMM construction algorithm and to report early experimental results. The section entitled 'THE MULTIPLE HMM' defines the general Multiple HMM, extending the formal definitions for 1- and 2-sequence HMMs that

have appeared elsewhere (Durbin *et al.*, 1998). 'THE EVOLUTIONARY HMM' shows how to construct an Evolutionary HMM by combining Branch HMMs on a guide tree. 'RESULTS' describes the implementation of these algorithms in the `Handel` package and gives results of tests on biological data. Finally, 'DISCUSSION' discusses how the work presented here can be used to implement a wide variety of multiple alignment algorithms that can be trained directly from data. A more formal exposition of certain points, including null state removal and dynamic programming, is given in the Appendices.

The `Handel` package is available under the GNU Public License from `www.biowiki.org/Handel/`.

THE MULTIPLE HMM

Suppose that Ω is a finite alphabet, e.g. $\Omega = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ for DNA. We write Ω^L for the set of all sequences of length L, and $\Omega^* = \bigcup_{L=0}^{\infty} \Omega^L$ for the set of all sequences. The length of a sequence S is written $|S|$. When written explicitly, sequences are enclosed by square brackets; e.g. the DNA sequence '`TATAA`' is written [`T`, `A`, `T`, `A`, `A`], while the empty sequence is written []. The concatenation of two sequences $a, b \in \Omega^*$ is written $a \circ b$.

State space

The Multiple HMM (MHMM) is a Markov model, $\mathcal{M}$, with a finite state space, Φ, that emits N separate sequences $S_1, S_2 \ldots S_N$. For simplicity, we will assume that each sequence S_n uses the same alphabet Ω, i.e. $S_n \in \Omega^* \ \forall n$. We can then write the vector of N emitted sequences as $\mathbf{S} \in (\Omega^*)^N$.

In the HMM literature, the states of a Markov model are often sequentially numbered $(1, 2 \ldots M)$ for convenience. In this paper, we use the different convention that states in the MHMM are represented by an abstract *identifier*. This can be a symbol, a string or even a number, but without the restriction that we have to use every number in some range $[1, M]$; instead, we just have to be able to enumerate all the states.

Let $t(i, j)$ be the transition probability from state i to state j. If $t(i, j) = 0$ then we say that there is no transition from i to j. Two special states are the start state, `START`, and the end state, `END`. For all $i \in \Phi$, we require that $t(i, \mathtt{START}) = t(\mathtt{END}, i) = 0$.

Emissions

We use the convention that each state in the MHMM emits symbols to some subset of the N sequences, emitting at most one symbol to each sequence. Thus there are at most $(|\Omega| + 1)^N$ different kinds of state.

An *emission* is an N-dimensional vector $\mathbf{E}$, each entry of which is a sequence of length 0 or 1. If every sequence has length 0 then $\mathbf{E}$ is a *null emission* (written $\mathbf{E} = []^N$).

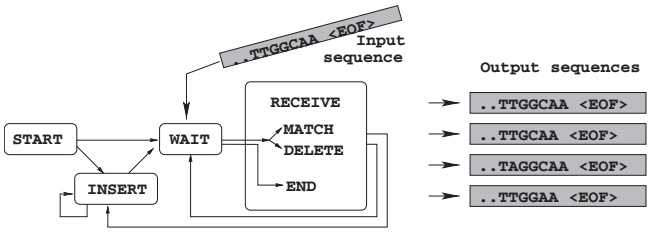

Fig. 2. Schematic illustration of the conditionally normalised MHMM, or stochastic transducer. Here `<EOF>` is a notional end-of-file symbol that propels the transducer into the `END` state once the input sequence is finished.

Each state i of the MHMM has associated with it an emission $\mathbf{E}(i)$. States with null emissions are referred to as *null states*. The special states `START` and `END` are always null.

Transducers

We now introduce a special kind of MHMM called a *transducer MHMM*. Rather than emitting symbols to all N sequences, a transducer MHMM inputs symbols from S_1 and outputs to the remaining sequences (Eskin *et al.*, 2000). We continue to use the term 'emission' to describe the vector $\mathbf{E}(\phi)$ defined in 'Emissions', with the caveat that the first entry of this vector $E(\phi)_1$ represents an absorbed, not an emitted, symbol.

Let us define certain terms. A *wait state* is any null state that is not `START` or `END`, while a *receive state* either inputs a symbol from S_1 or is the `END` state. We impose two requirements on transducers. The first requirement is that wait states can only make transitions into receive states; furthermore, only wait states can make such transitions. The second requirement is that outgoing transitions from wait states are normalised conditional on the next input symbol (or `END`); other transitions are normalised as usual. A more formal treatment can be found in the section entitled 'FORMAL DEFINITIONS'.

An illustration of the transducer MHMM is shown in Figure 2. Conceptually, the transducer pauses in a wait state until it is notified of an incoming symbol or end-of-input, which sends it into an appropriate receive state. If end-of-input has been reached, the appropriate state is `END`; otherwise, the transducer absorbs a symbol from the input sequence and emits either an identical symbol (a match), a different symbol (a mismatch) or no symbol (a deletion) to each output sequence. The machine then rattles around the insert states (potentially emitting more symbols to the output sequences) before returning to a wait state to wait for the next input symbol.

THE EVOLUTIONARY HMM

The Evolutionary HMM is a Multiple HMM that is constructed using two components: a *guide tree* and a *Branch HMM*. The general idea is that each Branch HMM is a two-sequence transducer modeling evolution along a single branch of the guide tree. This section introduces these concepts and shows how the state space of the Evolutionary HMM is a product of the state spaces of the Branch HMMs.

The guide tree

We consider a tree Υ with N nodes. Let $A(n)$ be the parent of node n, let $D(n)$ be the set of all children of node n and let $\mathbf{A}(n)$ and $\mathbf{D}(n)$ be the sets of, respectively, all ancestors and all descendants of node n. For convenience, we take node 1 to be the root, and stipulate that node 2 is its only child ($D(1) = \{2\}$); we call node 2 the *subroot*. An example of such a node numbering is shown in Figure 1.

We assume the nodes are numbered in preorder (i.e. parents before children: $A(n) < n$) and depth-first (suppose $l, m \in D(n)$ are siblings and $l < m$; then $k < m \ \forall k \in \mathbf{D}(l)$). We can also visit the nodes in postorder (children before parents) by iterating backwards from node N to node 1. A node that is visited before node n in such a postorder traversal, but is not a descendant of n, is said to be *eclipsed* by n.

We also require that each branch of the guide tree has a length. This is a time parameter, T, drawn from some set $\mathcal{T}$. Let T_n be the length of branch $A(n) \rightarrow n$. For the present work, we will assume that the set $\mathcal{T}$ is the continuous interval $[0, \infty)$, but it could also be a discrete range.

The MHMM generates a sequence for each of the N nodes. When we use this MHMM for multiple alignment we will generally have observed only the sequences at the leaf nodes. The ancestral sequences are 'missing data'; states that only emit symbols to these ancestral sequences must be eliminated ('ELIMINATING STATES').

The Branch HMM

The Branch HMM (BHMM) is a two-sequence transducer MHMM with state space Ψ whose transition probabilities $\tau(i, j|T)$ are functions of some time parameter, $T \in \mathcal{T}$. Let $\varepsilon(i)$ be the emission for state i of the Branch HMM; we assume these are independent of T.

We refer to sequence S_1 as the ancestor and sequence S_2 as the descendant. The ancestor is regarded as an 'input' and the descendant as an 'output' (c.f. Fig. 2). We will usually assume that the Branch HMM is conditionally normalised ('Normalisation'). It is possible to use an subnormalised BHMM; the consequence of this is that the associated EHMM is also subnormalised. Since we will be constructing the EHMM by combining Branch HMMs, and the EHMM itself has `START` and `END` states, we refer to the start and end states for an individual Branch HMM as `SBR` and `EBR` to avoid ambiguity.

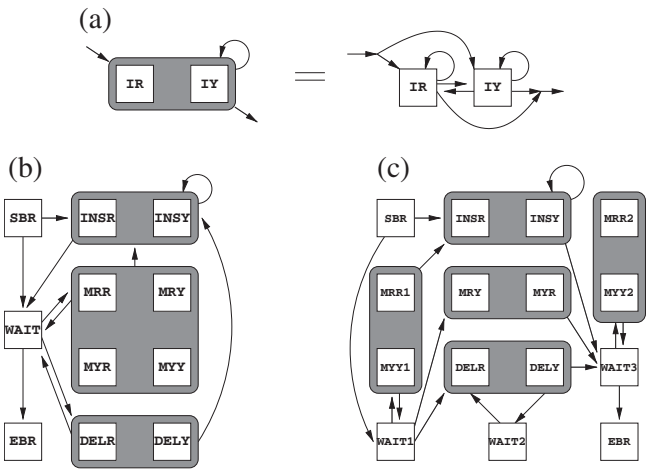

Fig. 3. Example Branch HMMs ('Normalisation'). These HMMs use the binary alphabet $\Omega = \{\mathtt{R}, \mathtt{Y}\}$ (purine/pyrimidine). The state types can be deduced from the state names: thus $\chi(\mathtt{MRY}) = \overset{\mathtt{RY}}{\mathtt{M}}$, $\chi(\mathtt{INSR}) = \overset{\mathtt{R}}{\mathtt{I}}$, $\chi(\mathtt{WAIT1}) = \mathtt{W}$ etc. (a) A shaded, rounded box is used to aggregate zeroth-order emit states. (b) Topology of the Branch HMM for the TKF91 model (Thorne *et al.*, 1991). (c) Topology of a Branch HMM that can be used to propose single-event substitution/insertion/deletion trajectories.

To classify the states of the BHMM, we define a function $\chi(\psi)$ denoting the type of a BHMM state ψ. The permitted state types are $\mathtt{W}$, $\overset{wx}{\mathtt{M}}$, $\overset{w}{\mathtt{D}}$, $\overset{x}{\mathtt{I}}$, $\mathtt{S}$ and $\mathtt{E}$, where $w, x \in \Omega$. The meaning of these state types is as follows: $\mathtt{W}$ is a wait state; $\overset{wx}{\mathtt{M}}$ is a match state with input symbol w and output symbol x; $\overset{w}{\mathtt{D}}$ is a delete state with input symbol w; $\overset{x}{\mathtt{I}}$ is an insert state with output symbol x; $\mathtt{S}$ is the start state (`SBR`); and $\mathtt{E}$ is the end state (`EBR`). Formal definitions are given in 'FORMAL DEFINITIONS'. We sometimes refer to $\overset{wx}{\mathtt{M}}$- and $\overset{x}{\mathtt{I}}$-states collectively as *output states*, since they output a symbol to the descendant sequence.

Two example Branch HMM topologies are shown in Figure 3 for a binary purine/pyrimidine alphabet $\Omega = \{\mathtt{R}, \mathtt{Y}\}$. Figure 3b describes branch evolution for the TKF91 single-residue indel model (Thorne *et al.*, 1991), while Figure 3c describes single-mutation trajectories in a 'long indel' model that allows indel events of geometrically distributed length (Miklos, Lunter and Holmes; in prep). The transition probabilities for these Branch HMMs are not shown. Note that Figure 3c lacks transitions to `EBR` from states `WAIT1` and `WAIT2`, and

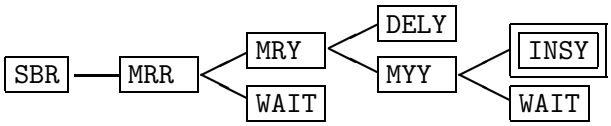

Fig. 4. Example of an EHMM state for the guide tree in Figure 1 and the Branch HMM in Figure 3b. This state would be written $\phi = \{\psi_1, \psi_2 \ldots \psi_8\}$ with $\psi_1 = \texttt{SBR}$, $\psi_2 = \texttt{MRR1}$, $\psi_3 = \texttt{MRY}$ etc. The active node $\alpha(\phi)$ is indicated with a double-bordered box.

so is subnormalised; in fact, in this Branch HMM, the Forward sum $\mathcal{F}(\mathbf{S})$ represents the total mutation rate of the sequence.

The inflated EHMM

We construct the Evolutionary HMM in two stages. In this section, we develop an inflated form of the model where each transition corresponds to a single event on some branch of the guide tree. In 'THE COLLAPSED EHMM', we describe a 'collapsed' state space for the model from which most of the null states have been removed. Even further collapsing of the state space is possible, e.g. the use of Felsenstein's algorithm to marginalise unobserved emissions at internal nodes (Durbin *et al.*, 1998).

In the Evolutionary HMM, each branch of the tree Υ is associated with a distinct Branch HMM (Fig. 4). For $n > 1$, let $\psi_n \in \Psi$ be the state of the BHMM associated with branch $A(n) \rightarrow n$; we can also think of this BHMM as being associated with node n. Just so that ψ_n is defined for all n, we also set $\psi_1 = \texttt{SBR}$.

The vector of states $\phi = \{\psi_1, \psi_2 \ldots \psi_N\}$ represents an individual state of the Evolutionary HMM. The start and end states of the EHMM are defined as we might expect: in the start state $\phi = \texttt{START}$, all nodes are labeled SBR, while in the end state $\phi = \texttt{END}$, all nodes are labeled EBR.

Let $\chi_n \equiv \chi(\psi_n)$ be the Branch HMM state type for branch $A(n) \rightarrow n$. If $\chi_n = \texttt{W}$, we say that node n is a 'wait node'; if $\chi_n = \overset{x}{\texttt{I}}$ for some $x \in \Omega$, we say node n is an 'insert node'; and so on.

For any given ϕ, the *active node* $\alpha(\phi)$ is defined to be the highest-numbered non-wait node; or (if there are no non-wait nodes) the sub-root. In a given EHMM state ϕ, only the active node and its descendants are allowed to change their branch state (Fig. 5). The descendants of the active node only change (from wait to receive states) if the active node emits a symbol or goes into the EBR state, in which case this event is propagated down the tree.

To make this explicit, suppose that the active node changes from state ψ_n to state ψ'_n where $n = \alpha(\phi)$. What happens next depends on the new state type of node n. If ψ'_n is a wait or delete state ($\chi(\psi'_n) \in \{\texttt{W}, \overset{w}{\texttt{D}}\}$), then the transition is complete. If ψ'_n is the EBR state, then all descendants are forced into the EBR state; this can only

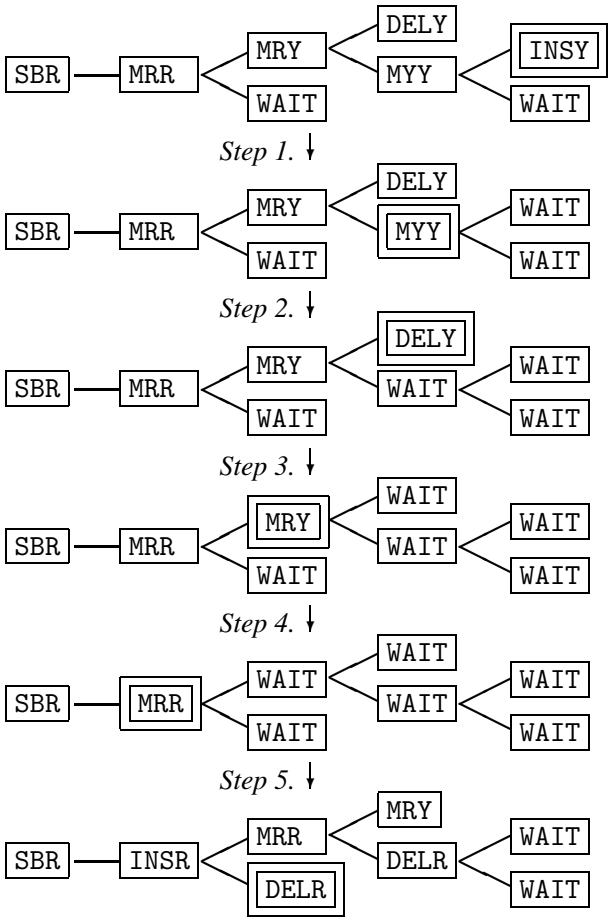

Fig. 5. Example path through the EHMM of Figure 4. *Step 1:* The initial state of the EHMM is one in which node 6 has just entered an INSY (insert) state (and so has just emitted a Y to sequence S_6). For this to have happened, nodes 7 and 8 must be in WAIT states. Node 6 is the highest non-WAIT node, and so is the active node ($\alpha(\phi) = 6$); since it is where the last emission event was initiated, node 6 is also the emitter node ($\beta(\phi) = 6$). In the first step of this path, node 6 goes into a WAIT state, leaving node 5 as the new active node. No symbols are emitted in this step. *Step 2:* Node 5 goes from an MYY (match) state to a WAIT state, leaving node 4 as the new active node. No symbols are emitted. *Step 3:* Node 4 goes from a DELY (delete) state to a WAIT state, leaving node 3 as the new active node. No symbols are emitted. *Step 4:* Node 3 goes from an MRY (mismatch) state to a WAIT state, leaving node 2 as the new active node. No symbols are emitted. *Step 5:* Node 2 goes into an INSR (insert) state, emitting an R to sequence S_2. Since node 2 has entered an output state, nodes 3 and 8 are forced into receive states: node 3 goes into an MRR (match) state (emitting a R to sequence S_3), while node 8 goes into a DELR (delete) state (and therefore emits nothing). Since node 3 has entered an output state, nodes 4 and 5 are forced into receive states: node 4 goes into an MRY (match) state (emitting a Y to sequence S_4), while node 5 goes into a DELR (delete) state (and emits nothing). Since node 5 is not in an output state, the wave of updates ends here; in this one step, an R was emitted to S_2 and S_3, while a Y was emitted to S_4. The emitter node, i.e. the node at which the wave originated, is node 2. Node 8 is now the highest-numbered non-WAIT state and so is the new active node, i.e. it will be the next to be updated.

Table 1. Relative performance of alignment strategies.

Treatment of internal sequences	Accuracy		
	Progressive	Refined	Resampled
Sampled	55.2%	54.9%	57.4%
Marginalised	63.4%	65.4%	66.0%

happen if $n = 2$. If ψ'_n is an output state ($\chi(\psi'_n) \in \{\overset{x}{\mathtt{I}}, \overset{wx}{\mathtt{M}}\}$), then all children of the active node are forced into corresponding receive states ($\overset{xy}{\mathtt{M}}$ or $\overset{x}{\mathtt{D}}$); if any of these receive states are $\overset{xy}{\mathtt{M}}$-states, then the children of these nodes are forced into receive states; and so on down the tree (see e.g. Fig. 5).

Emission states have characteristic properties. Suppose that $\phi = \{\psi_1 \ldots \psi_N\}$ is a general EHMM state. If ϕ is an emission state, we can identify the *emitter node* $\beta(\phi)$ where the emission event was initiated: it must have been at the highest-numbered insert node or (if there are no insert nodes) at the sub-root. If ϕ is an emission state, then the input symbol of every node descended from $\beta(\phi)$ will match the output symbol of its parent. We call this property *synchronisation*. Formal definitions of this and other concepts introduced in this section are given in 'FORMAL DEFINITIONS'.

This concludes our definition of the inflated Evolutionary HMM. It is easy to verify that the EHMM, thus constructed, is a conditionally normalised Multiple HMM as defined in 'Normalisation'. It is also evident that the EHMM models the action of evolution along each branch as an independent Branch HMM, since all we have effectively done is to specify an order for updating the states of these Branch HMMs. In 'THE COLLAPSED EHMM', we describe how to eliminate all unnecessary null states from the EHMM; that is, all states except for `START`, `END`, the emission states, and *all-wait states* (where the sub-root and all descendants are `W`-nodes).

Sampling issues

The EHMM jointly models all sequences and alignments in the tree. When sampling multiple alignments, considerations of memory and time complexity may dictate that we restrict ourselves to resampling the alignment for a small part of the tree at any individual step. To do this, we construct an EHMM for the subtree of interest; e.g. for an individual branch (→) or the neighbourhood of a node (). These procedures—called branch sampling and node sampling—are detailed elsewhere (Holmes and Bruno, 2001).

Unobserved internal sequences can be removed from the observed sequence vector $\mathbf{S} = \{S_1, S_2 \ldots S_N\}$ by discarding some emission indices and renumbering the rest. For example, in the tree of Figure 1, we would discard emission indices S_1, S_2, S_3 and S_5 and renumber $S_4 \rightarrow S'_1$, $S_6 \rightarrow S'_2$, $S_7 \rightarrow S'_3$ and $S_8 \rightarrow S'_4$. Some emit states of the EHMM only emit symbols to internal sequences; these states can be marginalised using the state elimination algorithm described in 'ELIMINATING STATES'. If the Branch HMM is zeroth-order (Fig. 3), one can use Felsenstein's algorithm to sum out internal residues (Durbin *et al.*, 1998).

RESULTS

We have implemented the EHMM construction and alignment algorithms for the TKF91 model (Fig. 3b), including the branch and node sampling techniques and internal-sequence elimination mentioned in 'The Branch HMM', in the `Handel` package (Holmes and Bruno, 2001). The TKF91 model describes the evolution of a whole sequence under the influence of single-residue indels and substitutions, and so is roughly the phylogenetic equivalent of a global alignment algorithm with linear gap penalties.

The EHMM construction algorithm described here works for any choice of Branch HMM. By suitable choice of HMM states and transitions, we could (for example) use a Branch HMM with affine gaps and 'ragged ends' (i.e. local alignment), rather than the TKF91 model (which is closer to global alignment with a linear gap model). At the time of writing, the only EHMM implemented in Handel is based on the TKF91 model, as this was the only rate-based model whose finite-time Branch HMM had been analytically determined. Recently, we have developed a long indel model allowing multi-residue insertions and deletions, using a generalised Branch HMM (Miklos, Lunter & Holmes; in preparation). A more systematic heuristic for finding a Branch HMM that approximates a complex evolutionary model is to simulate a large number of pairwise alignments under the model, then to fit a Branch HMM to these pairwise alignments using the Baum–Welch algorithm (Durbin *et al.*, 1998).

Despite the simplicity of the TKF91 model, it performs reasonably well at multiple alignment (Holmes and Bruno, 2001), approaching the accuracy of CLUSTALW on some parts of BAliBase, the test set for which CLUSTALW was optimised. This benchmarking suggests that the clearest improvements to `Handel` involve updating the Branch HMM to address (e.g.) local alignments, affine gaps, gap-free blocks and hydrophobic core modeling: there is only so much accuracy to be gotten from global alignment with linear gaps. Here, we focus on the general behaviour of the EHMM rather than on comparative analyses with other multiple alignment tools.

```
root                    ---*-****-*---*--****-*-***-*---*-*--**********************-***************
thyg_bovin::bal_human   ---*-****-*---*--****-*-***-*---*-*--********************-***-***************
thyg_bovin::est1_caebr  ---*-****-*---*--****-*-***-*---*-*--**********************-***************
est1_caebr::crys_dicdi  ---*-****-*---*--****-*-***-*---*-*--**********************-***************
thyg_bovin              SPSVPIATH-GQLLGR-SQAI-Q-VG--T-SW-KPVDQFLGVPYAAPPLGEKRFRAPE-HLN-WTGSWEATKPRARCWQ
bal_human               ---V-Y-TE-G---G-FVEGVNKKLGLLGDS-VD-I--FKGIPFA-APTKAL--ENPQ-PHPGWQGTLKAKNFKKRCLQ
est1_caebr              ---V-VNTNYG---K--VEGF-E-YG--A-A--E-V--FLAIPFAKPPVDNLRFEKPEAPEP-WEDVYQATQFRNDC-T
crys_dicdi              ---V-LLSD-G---A--IRGT-V-TD--T-H--R-V--FYGIPFARPPIDELRYEDPQPPKP-WSYVRDGTKQRDQCIQ
est1_culpi              ESLT-VQTKYG---P--VRGK-RNVSLLG-Q--EYVS-FQGIPYARAPEGELRFKAPVPPQK-WTETLDCTQQCEPCYH

thyg_bovin   SPSVPIATHGQLLG-RSQAIQVGTSWKPVDQFLGVPYAAPPLGEKRFRAPE-HLNWTGSWEATKPRARCWQPGI---RTPTPPGVSEDCLY
bal_human    ----VYTEGGFVEGVNKKLGLLG---DSVDIFKGIPFAAPT---KALENPQPHPGWQGTLKAKNFKKRCLQATI---TQDSTYG-DEDCLY
est1_caebr   ---VVNTNYGKVEG-----FEYG----AAEVFLAIPFAKPPVDNLRFEKPEAPEPWEDVYQATQFRNDCTPHYRLVAQFSSYSG--EDCLT
crys_dicdi   ---VLLSD-GAIRG---------TVTDTHRVFYGIPFARPPIDELRYEDPQPPKPWSYVRDGTKQRDQCIQDCKLGKGSCSEVGTSEDCLY
est1_culpi   ESLTVQTKYGPVRG-KRNVSLLG---QEYVSFQGIPYARAPEGELRFKAPVPPQKWTETLDCTQQCEPCYHFDR---RLQKIVG-CEDSLK
```

Fig. 6. Top: Part of a `Handel` alignment of esterase sequences from BAliBase. Asterisks are *Felsenstein wildcards*, used when `Handel` infers that an ancestral residue existed at a given position but has used Felsenstein's algorithm to sum out the substitution likelihood (Holmes and Bruno, 2001). The notation `A::B` indicates a common ancestor of `A` and `B`. For reference, the inferred phylogeny in Newark format was `((thyg_bovin:0.72,bal_human:0.69):0.032, (est1_caebr:0.71, crys_dicdi:0.70):0.0070, est1_culpi:0.70)`. Bottom: Part of the BAliBase structural alignment of the same sequences.

We tried three different alignment strategies: progressive, refined alignment (using the Viterbi algorithm) and resampled alignment (using the Forward algorithm). We also tried both marginalising and sampling the unobserved sequences at internal nodes of the tree; in other words, we tried running the program both with and without Felsenstein wildcards. The results of these tests are summarised in Table 1.

When internal sequences are marginalised, refining the alignment improves the alignment quality in 57% of the BAliBase datasets, while resampling the alignment leads to a further improvement over the refined alignment in 35% of the datasets (with a further 37% of datasets showing no change). Considering BAliBase as a whole, resampling leads to a net increase in accuracy of 1% over refinement.

When we sample internal sequences instead of marginalising them, refinement leads to worse alignments in the majority (67%) of cases, although the net loss of accuracy over all of BAliBase is small ($< 1\%$). When we resample the alignments, 46% of the datasets show improved accuracy over the refined alignments, with a further 45% showing no change. In general, marginalisation seems to be an effective strategy: in 88% of the BAliBase datasets, marginalisation yielded a more accurate multiple alignment, with a total increase in accuracy of around 9%.

`Handel` and BAliBase alignments for a family of acetylcholinesterase-related proteins are shown in Figure 6. It can be seen that the `Handel` alignment is generally more 'gappy', an artefact of the TKF91 model. `Handel`'s sampling behaviour for this family is illustrated in Figure 7.

DISCUSSION

We defined a number of new algorithmic concepts including the *Multiple HMM*, a hidden Markov model for multiple sequence alignment; the *Branch HMM*, a stochastic transducer modeling the evolution of a single phylogenetic branch; and the *Evolutionary HMM*, a Multiple HMM made by combining Branch HMMs on a guide tree. We have given algorithms for construction of and alignment to such models. For a specific Branch HMM (the TKF91 model) we have implemented these algorithms and analysed their performance on biological sequences. Our results suggest alignment resampling helps avoid local minima especially when unobserved ancestral sequences are sampled rather than marginalised. The fully general EHMM construction algorithm is currently under implementation in our group.

Although the EHMM we have described is normalised conditional on the root sequence S_1, it is possible to construct an EHMM that is jointly normalised for an entire sequence set. To do this, we simply take the limit $T_2 \to \infty$; i.e. we let the length of the branch from the root to the subroot become infinitely long. We then set the root sequence equal to the empty sequence, $S_1 = []$. The EHMM then gives a joint probability distribution for the remaining sequences $S_2 \ldots S_N$.

It is possible to 'nest' one (Evolutionary) Multiple HMM inside another: given a Multiple HMM, we can replace each emit state ϕ with a sub-model constructed on the set of sequences to which ϕ emits. This allows, for example, a version of the TKF91 links model where each link is itself a miniature TKF91 model. Full exploitation of such models may, in some cases, demand specialised dynamic programming algorithms.

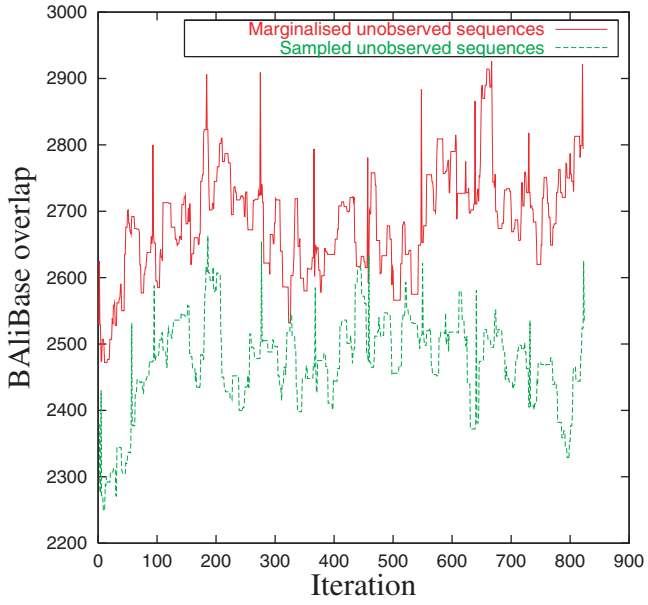

Fig. 7. Behaviour of `Handel` on a dataset of low-identity (< 25%) esterase proteins from BAliBase. The graph shows the accuracy throughout repeated node and branch resampling (section: 'The Branch HMM'). The accuracy is measured in the residue-pair overlap with a structural BAliBase alignment, out of a maximum of 4407 residue-pairs. Periodically, `Handel` refines the alignment by iterating the Viterbi algorithm (appendix: 'DYNAMIC PROGRAMMING') on nodes and branches; this causes the regular spikes on the graph. In the upper plot, the unobserved sequences at internal nodes of the tree were marginalised using Felsenstein's algorithm (section: 'Sampling issues'); in the lower plot, these sequences were sampled.

An obvious adjunct to the EHMM construction algorithm would be an algorithm to construct evolutionary stochastic context-free grammars (SCFGs) from pairwise 'Branch SCFGs'. Such an algorithm, the probabilistic counterpart of Sankoff *et al*'s algorithm for multiple sequence alignment and RNA structure prediction (reviewed by Durbin *et al*), would allow the evolutionary modeling of changes in RNA structure. Given the current emphasis on comparative genomic analysis, probabilistic evolutionary methods such as those presented here are timely and relevant.

ACKNOWLEDGEMENTS

The author acknowledges E.Z.Holmes for graphics support. This work was supported by EPSRC (code HAMJW) and MRC (code HAMKA).

REFERENCES

Durbin,R., Eddy,S., Krogh,A. and Mitchison,G. (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, UK.

Eskin,E., Grundy,W.N. and Singer,Y. (2000) Protein family classification using sparse Markov transducers. In Bourne,P., Gribskov,M., Altman,R., Jensen,N., Hope,D., Lengauer,T., Mitchell,J., Scheeff,E., Smith,C., Strande,S. and Weissig,H. (eds), *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Menlo Park, CA, pp. 134–135.

Friedman,N., Ninio,M., Pe'er,I. and Pupko,T. (2001) A structural EM algorithm for phylogenetic inference. In Lengauer,T., Sankoff,D., Istrail,S., Pevzner,P. and Waterman,M. (eds), *Proceedings of the Fifth Annual International Conference on Computational Biology*. Association for Computing Machinery, New York.

Hein,J., Wiuf,C., Knudsen,B., Moller,M.B. and Wibling,G. (2000) Statistical alignment: computational properties, homology testing and goodness-of-fit. *J. Mol. Biol.*, **302**, 265–279.

Holmes,I. and Bruno,W.J. (2001) Evolutionary HMMs: a Bayesian approach to multiple alignment. *Bioinformatics*, **17(9)**, 803–820.

Holmes,I. and Rubin,G.M. (2002) An expectation maximization algorithm for training hidden substitution models. *J. Mol. Biol.*, **317(5)**, 757–768.

Thompson,J.D., Higgins,D.G. and Gibson,T.J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.

Thorne,J.L., Kishino,H. and Felsenstein,J. (1991) An evolutionary model for maximum likelihood alignment of DNA sequences. *J. Mol. Evol.*, **33**, 114–124.

THE COLLAPSED EHMM

The inflated state space has a lot of unnecessary null states (Fig. 5). We can eliminate many of these by allowing direct transitions between accessible emit states, as well as `START`, `END` and all-wait states. These states constitute the *collapsed state space* $\Phi'' \subseteq \Phi$. In this appendix, we give an algorithm to construct Φ'' explicitly. This is the EHMM construction algorithm that will primarily be used in practise.

We first suppose that, excluding branch $1 \rightarrow 2$, the guide tree Υ is a binary tree. This is without loss of generality, since we can always convert a non-binary tree into a binary tree by suitable manipulations.

For a generic BHMM state type $\mathtt{X} \in \left\{ \overset{wx}{\mathtt{M}}, \overset{w}{\mathtt{D}}, \overset{x}{\mathtt{I}}, \mathtt{W}, \mathtt{S}, \mathtt{E} \right\}$:

Let $A_m(\mathtt{X})$ be the set of all partial labelings for the subtree rooted at node m, such that (i) node m is in an `X`-state, (ii) no nodes are in `I`-states.

Let $B_m(\mathtt{X})$ be the set of all partial labelings for the subtree rooted at node m, such that (i) node m is in an `X`-state, (ii) at least one node is in an `I`-state.

If m is an internal node of Υ, then let u and v represent the two child nodes, with $u < v$. Suppose that U and V are partial labelings rooted at nodes u and v respectively. Let

$\text{—X}\overset{U}{<}V$ represent a partial labeling rooted at node m, with node m itself having type X and with the subtrees rooted at u and v labeled according to U and V, respectively.

If m is a leaf node of Υ, then let —X represent a partial labeling of node m, so that the state of node m has type X.

Note that $A_m\left(\overset{x}{\text{I}}\right) = B_m\left(\overset{w}{\text{D}}\right) = B_m(\text{W}) = B_m(\text{E}) = \emptyset\ \forall m$. We construct the remaining $A_m(\text{X})$, $B_m(\text{X})$ recursively. For internal nodes ($|\mathbf{D}(m)| > 0$):

$$A_m\left(\overset{wx}{\text{M}}\right) = \left\{ \text{—}\overset{wx}{\text{M}}\overset{U}{<}V : \begin{array}{l} U \in A_u\left(\overset{xy}{\text{M}}\right) \cup A_u\left(\overset{x}{\text{D}}\right), \\ V \in A_v\left(\overset{xz}{\text{M}}\right) \cup A_v\left(\overset{x}{\text{D}}\right), \\ y, z \in \Omega \end{array} \right\}$$

$$A_m(\text{W}) = \left\{ \text{—W}\overset{U}{<}V : \begin{array}{l} U \in A_u(\text{W}), \\ V \in A_v(\text{W}) \end{array} \right\}$$

$$A_m\left(\overset{w}{\text{D}}\right) = \left\{ \text{—}\overset{w}{\text{D}}\overset{U}{<}V : \begin{array}{l} U \in A_u(\text{W}), \\ V \in A_v(\text{W}) \end{array} \right\}$$

$$A_m(\text{S}) = \left\{ \text{—S}\overset{U}{<}V : \begin{array}{l} U \in A_u(\text{S}), \\ V \in A_v(\text{S}) \end{array} \right\}$$

$$A_m(\text{E}) = \left\{ \text{—E}\overset{U}{<}V : \begin{array}{l} U \in A_u(\text{E}), \\ V \in A_v(\text{E}) \end{array} \right\}$$

$$B_m(\text{S}) = \left\{ \text{—S}\overset{U}{<}V : \begin{array}{l} U \in B_u(\text{S}) \cup B_u\left(\overset{y}{\text{I}}\right), \\ V \in A_v(\text{W}), \\ y \in \Omega \end{array} \right\} \cup \left\{ \text{—S}\overset{U}{<}V : \begin{array}{l} U \in A_u(\text{S}), \\ V \in B_v(\text{S}) \cup B_v\left(\overset{z}{\text{I}}\right), \\ z \in \Omega \end{array} \right\}$$

$$B_m\left(\overset{wx}{\text{M}}\right) = \left\{ \text{—}\overset{wx}{\text{M}}\overset{U}{<}V : \begin{array}{l} U \in B_u\left(\overset{xy}{\text{M}}\right) \cup B_u\left(\overset{y}{\text{I}}\right), \\ V \in A_v(\text{W}), \\ y \in \Omega \end{array} \right\} \cup \left\{ \text{—}\overset{wx}{\text{M}}\overset{U}{<}V : \begin{array}{l} U \in A_u\left(\overset{xy}{\text{M}}\right) \cup A_u\left(\overset{x}{\text{D}}\right), \\ V \in B_v\left(\overset{xz}{\text{M}}\right) \cup B_v\left(\overset{z}{\text{I}}\right), \\ y, z \in \Omega \end{array} \right\}$$

$$B_m\left(\overset{x}{\text{I}}\right) = \left\{ \text{—}\overset{x}{\text{I}}\overset{U}{<}V : \begin{array}{l} U \in B_u\left(\overset{xy}{\text{M}}\right) \cup B_u\left(\overset{y}{\text{I}}\right), \\ V \in A_v(\text{W}), \\ y \in \Omega \end{array} \right\} \cup \left\{ \text{—}\overset{x}{\text{I}}\overset{U}{<}V : \begin{array}{l} U \in A_u\left(\overset{xy}{\text{M}}\right) \cup A_u\left(\overset{x}{\text{D}}\right), \\ V \in A_v\left(\overset{xz}{\text{M}}\right) \cup A_v\left(\overset{x}{\text{D}}\right) \\ \qquad \cup B_v\left(\overset{xz}{\text{M}}\right) \cup B_v\left(\overset{z}{\text{I}}\right), \\ y, z \in \Omega \end{array} \right\}$$

For leaf nodes ($\mathbf{D}(m) = \emptyset$):

$$\begin{array}{rcll} A_m(\text{X}) & = & \left\{\text{—}\overset{wx}{\text{M}}\right\} & \forall\, \text{X} \in \{\overset{wx}{\text{M}}, \overset{w}{\text{D}}, \text{W}, \text{S}, \text{E} : w, x \in \Omega\} \\ B_m\left(\overset{x}{\text{I}}\right) & = & \left\{\text{—}\overset{x}{\text{I}}\right\} & \\ B_m(\text{S}) & = & \emptyset & \forall\, \text{X} \in \{\overset{wx}{\text{M}}, \text{S} : w, x \in \Omega\} \end{array}$$

The collapsed state space is thus

$$\Phi'' = \bigcup_{w,x\in\Omega} \left(\bigcup_{\text{X}\in\{\overset{wx}{\text{M}},\overset{w}{\text{D}},\text{S},\text{E}\}} A_2(\text{X}) \right) \cup \left(\bigcup_{\text{X}\in\{\overset{wx}{\text{M}},\overset{w}{\text{D}},\overset{x}{\text{I}},\text{S}\}} B_2(\text{X}) \right)$$

In the collapsed state space, the likelihood of inflated paths such as the one shown in Figure 5 are collapsed into a single 'effective' transition probability. Suppose that $\phi, \phi' \in \Phi''$ are states in the collapsed state space, with $\phi = \{\psi_1 \ldots \psi_N\}$ and $\phi' = \{\psi'_1 \ldots \psi'_N\}$. Thus ψ_n is the Branch HMM state of node n in EHMM state ϕ, and ψ'_n is the Branch HMM state of node n in EHMM state ϕ'.

For the collapsed transition probability $t(\phi, \phi')$ to be nonzero, it is required firstly that **either** $\beta(\phi') \geq \beta(\phi)$ **or** $\beta(\phi') \in \mathbf{A}(\beta(\phi))$. That is, the emitter can sweep forward (in a preorder traversal) or back up (to an ancestor).

FORMAL DEFINITIONS

'Emissions': An emission is a vector of sequences $\mathbf{E} \in (\Omega^0 \cup \Omega^1)^N$. A null emission $\mathbf{E} = []^N$ has $E_n = []\ \forall n$. START and END always have null emissions, $\mathbf{E}(\text{START}) = \mathbf{E}(\text{END}) = []^N$. The emit set for a state is defined by $\mathcal{B}(\phi) = \{n : E_n \neq []\}$.

'Normalisation': the set of wait states is $\Phi_W = \{\phi \in \Phi : \phi \notin \{\text{START}, \text{END}\}, \mathbf{E}(\phi) = []^N\}$ while the set of receive states is $\Phi_R = \{\phi \in \Phi : |E(\phi)_1| = 1\} \cup \{\text{END}\}$. The first requirement of transducers is, for all nonzero transitions $\phi, \phi' : t(\phi, \phi') > 0$, we have $\phi \in \Phi_W \Leftrightarrow \phi' \in \Phi_R$. For the second requirement, let $\Phi_R(\omega) = \{\phi \in \Phi : E(\phi)_1 = \omega\}$ be the set of receive states that absorb symbol ω. The conditional normalisation requirement for wait states is $\sum_{\phi' \in \Phi_R(\omega)} t(\phi, \phi') = 1\ \forall \phi \in \Phi_W, \omega \in \Omega$, and $t(\phi, \text{END}) = 1\ \forall \phi \in \Phi_W$. The joint normalisation requirement for non-wait states is $\sum_{\phi' \in \Phi} t(\phi, \phi') = 1\ \forall \phi \in \Phi, \phi \notin \Phi_W, \phi \neq \text{END}$.

'Normalisation': $\chi(\psi)$ is defined as follows. For $w, x \in \Omega$

$$\chi(\psi) = \begin{cases} \text{S} & \text{if } \psi = \text{SBR} \\ \text{E} & \text{if } \psi = \text{EBR} \\ \text{W} & \text{if } \varepsilon(\psi)_1 = \varepsilon(\psi)_2 = [] \text{ and } \psi \notin \{\text{SBR}, \text{EBR}\} \\ \overset{wx}{\text{M}} & \text{if } \varepsilon(\psi)_1 = [w] \text{ and } \varepsilon(\psi)_2 = [x] \\ \overset{w}{\text{D}} & \text{if } \varepsilon(\psi)_1 = [w] \text{ and } \varepsilon(\psi)_2 = [] \\ \overset{x}{\text{I}} & \text{if } \varepsilon(\psi)_1 = [] \text{ and } \varepsilon(\psi)_2 = [x] \end{cases}$$

'The inflated EHMM': The active node $\alpha(\phi)$ is defined as follows: let $\mathcal{S}(\phi) = \{n : 1 < n \leq N, \chi(\psi_n) \neq \mathtt{W}\}$. If $|\mathcal{S}(\phi)| > 0$ then $\alpha(\phi) = \max(n \in \mathcal{S}(\phi))$; otherwise, $\alpha(\phi) = 2$.

If $\phi = \{\psi_1 \dots \psi_N\}$ and $\phi' = \{\psi'_1 \dots \psi'_N\}$ are two EHMM state identifiers with $\phi' \neq \mathtt{END}$, then the inflated transition probability $t'(\phi, \phi')$ is given by

$$t'(\phi, \phi') = \tau(\psi_{a(\phi)}, \psi'_{a(\phi)} | T_{a(\phi)}) \times \prod_{n \in \mathbf{D}(a(\phi))} \sigma(\psi'_{A(n)}, \psi_n, \psi'_n, T_n) \times \prod_{\substack{n \neq a(\phi), \\ n \notin \mathbf{D}(a(\phi))}} \delta(\psi_n, \psi'_n) \quad (1)$$

where $\sigma(p', n, n', T)$ is a 'sync factor' ensuring that a node's state (n) and its parent's state (p) remain synchronised (here T is the branch length)

$$\sigma(p', n, n', T) = \begin{cases} \delta(\varepsilon(p')_2, \varepsilon(n')_1) & \\ \quad \tau(n, n'|T) & \text{if } |\varepsilon(p')_2| > 0 \\ \delta(n, n') & \text{otherwise.} \end{cases}$$

The transition from ϕ to END has probability $t'(\phi, \mathtt{END}) = \delta(a(\phi), 2)\delta(\chi(\psi_2), \mathtt{W})$.

The emitter $\beta(\phi)$ is defined as follows: if the set $\mathcal{I}(\phi) = \{n : \varepsilon(\psi_n)_1 = [], \varepsilon(\psi_n)_2 \neq []\}$ is nonempty, then $\beta(\phi) = \max(n \in \mathcal{I}(\phi))$; otherwise, $\beta(\phi) = 2$.

The synchronisation condition requires that $\varepsilon(\psi_n)_1 = \varepsilon(\psi_{A(n)})_2 \ \forall n \in \mathbf{D}(\beta(\phi))$. Synchronisation is clearly a necessary condition for ϕ to be an emission state, due to the 'sync factor' $\sigma(\dots)$ of Equation 1. To prove that it is also a sufficient condition note that, on exiting the emission state, either $\beta(\phi)$ or one of its descendants must change state, destroying synchronisation.

The emit set $\mathcal{B}(\phi)$ is given by $\mathcal{B}(\phi) = \{\beta(\phi)\} \cup \{n : n \in \mathbf{D}(\beta(\phi)), |\varepsilon(\psi_n)_1| = |\varepsilon(\psi_n)_2| = 1\}$. If ϕ is synchronised, the associated emission $\mathbf{E}(\phi)$ is defined by $E(\phi)_n = \varepsilon(\psi_n)_2$ for $n \in \mathcal{B}(\phi)$ and $E(\phi)_n = []$ otherwise. If ϕ is not synchronised then $\mathbf{E}(\phi) = []^N$.

ELIMINATING STATES

It is sometimes useful to be able to eliminate some subset of states from the state space Φ of an MHMM, retaining some subset $Y \subset \Phi$ and incorporating the likelihood of all paths through the eliminated states $X = \overline{Y}$ into effective transition probabilities between the remaining states in Y. Conversely, we may want to generate sample paths through Φ consistent with a given path through Y. For example, we often want to remove null states (and null cycles) from an HMM before dynamic programming, then restore them after traceback. For the purposes of this section, we write the transition function as a $|\Phi| \times |\Phi|$ matrix $\mathbf{t}$ with entries $t_{ij} \equiv t(i, j)$.

Let $y_k = 1$ if $k \in Y$ and 0 if $k \in X$. Suppose that $\mathbf{t} = \mathbf{a}+\mathbf{b}+\mathbf{c}+\mathbf{d}$ where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ are $|\Phi| \times |\Phi|$ matrices with entries $a_{ij} = t_{ij} y_i y_j$, $b_{ij} = t_{ij} y_i (1 - y_j)$, $c_{ij} = t_{ij}(1 - y_i) y_j$ and $d_{ij} = t_{ij}(1 - y_i)(1 - y_j)$. The effective transition probability q_{ij} between two states $i, j \in Y$ is given by $q_{ij} = a_{ij} + \sum_k b_{ik} c_{kj} + \sum_{k,l} b_{ik} d_{kl} c_{lj} + \sum_{k,l,m} b_{ik} d_{kl} d_{lm} c_{mj} + \dots$ or, in matrix notation, $\mathbf{q} = \mathbf{a} + \sum_{n=0}^{\infty} \mathbf{b}\mathbf{d}^n\mathbf{c} = \mathbf{a} + \mathbf{b}(\mathbf{1} - \mathbf{d})^{-1}\mathbf{c}$ where $\mathbf{1}$ is the identity matrix. Here $\mathbf{a}$ describes transitions within Y, while the summand $\mathbf{b}\mathbf{d}^n\mathbf{c}$ describes the probability of making a transition from Y into the eliminated states X, followed by n transitions within the eliminated states, and finally a transition back into Y.

The converse problem is the probabilistic sampling of a path segment $\pi = [\pi_1, \pi_2 \dots \pi_{|\pi|}] \in \Phi^*$ consistent with a given transition $i \to j$ where $i, j \in Y$. Here $\pi_1 = i$, $\pi_n = j$ and $\pi_m \in X \ \forall m : 1 < m < n$. To generate this sample we need to know the probability r_{ij} of all paths from $i \in Y$ to $j \in X$, starting with a transition from Y into X, and remaining in X until reaching j after any number of steps. In matrix notation, $\mathbf{r} = \sum_{n=0}^{\infty} \mathbf{b}\mathbf{d}^n = \mathbf{b}(\mathbf{1} - \mathbf{d})^{-1}$. The algorithm for sampling π then uses variables $k, l \in \Phi$ as follows

- Begin by setting $k \leftarrow j$, $\pi \leftarrow [j]$.
- **Loop:** Sample $l \in \Phi$ from the distribution $P(l|k, i) = r_{il} p_{lk} / \sum_{x \in \Phi} r_{ix} p_{xk}$.
- Set $\pi \leftarrow [l] \circ \pi$ and $k \leftarrow l$.
- If $k = i$, then terminate; otherwise, goto **Loop**.

This algorithm can be used to sample a path in Φ consistent with an observed path in Y, by repeated application to each transition in the observed path.

Given an effective transition $i \to j$ where $i, j \in Y$, we may want to compute $n(k \to l | i \to j)$, the posterior expected number of null transitions $k \to l$. For this, we need $\mathbf{s} = (\mathbf{1} - \mathbf{d})^{-1}\mathbf{c}$ as well as the previously-calculated $\mathbf{r}$ and $\mathbf{q}$. Then the expectation is given by $n(k \to l | i \to j) = r_{ik} d_{kl} s_{lj} / q_{ij}$. Related expectations are $n(i \to k | i \to j) = b_{ik} s_{kj} / q_{ij}$ and $n(l \to j | i \to j) = r_{il} c_{lj} / q_{ij}$. Finally, the posterior expected number of actual $i \to j$ transitions given an observed effective $i \to j$ transition is a_{ij} / q_{ij}.

DYNAMIC PROGRAMMING

We first introduce some extra notation. For $S \in \Omega^*$ define the *left and right subsequence* functions $\mathcal{L} : \Omega^* \times \mathbf{Z} \to \Omega^*$ and $\mathcal{R} : \Omega^* \times \mathbf{Z} \to \Omega^*$ such that $\mathcal{L}(a \circ b, |a|) = \mathcal{R}(b \circ a, |a|) = a$ for $a, b \in \Omega^*$, while $\mathcal{L}(a, n) = \mathcal{R}(a, n) = a$ for out-of-range subscripts $n > |a|$.

For $a, b \in \Omega^*$, define the relation $a \lhd b$ as $\{(a, b) : \exists c \in \Omega^*, a \circ c = b\}$, meaning '$a$ is a left subsequence of b'.

The serial concatenation of a sequence of K sequences $[X_1, X_2 \ldots X_K] \in (\Omega^*)^*$ is written $\bigcirc_{k=1}^{K} X_k$.

Let $\pi = [\pi_1, \pi_2 \ldots \pi_K] \in \Phi^*$ be a path in the state space, with $|\pi| = K$. Define $\varrho(\pi) = \prod_{k=2}^{|\pi|} t(\pi_{k-1}, \pi_k)$ to be the *path product* and and $\mathbf{G} : \Phi^* \rightarrow (\Omega^*)^N$, $G(\pi)_n = \bigcirc_{k=1}^{|\pi|} E(\pi_k)_n$ to be the *path emission*. Let $\Pi(\alpha, \beta, \mathbf{X}) = \{\pi : \pi_1 = \alpha, \pi_{|\pi|} = \beta, \mathbf{G}(\pi) = \mathbf{X}\}$ be the set of all paths from α to β emitting $\mathbf{X}$, where $\alpha, \beta \in \Phi$ and $\mathbf{X} \in (\Omega^*)^N$.

Define the *Forward sum* $\mathcal{F}(\mathbf{S}) = \sum_{\pi \in \Pi(\mathtt{START}, \mathtt{END}, \mathbf{S})} \varrho(\pi)$. For models normalised by the criteria of 'Normalisation', $\mathcal{F}(\mathbf{S})$ is the likelihood of the sequences $\mathbf{S}$. The Forward sum is computed by dynamic programming (DP) as follows.

Let $\mathbf{M} : (\Omega^*)^N \rightarrow \mathbf{Z}^N$ be a function returning an N-dimensional vector of sequence lengths (so $M(\mathbf{S})_n = |S_n|$), let $\mathbf{C} \in \mathbf{Z}^N$ be an N-dimensional vector of *cell co-ordinates* and let $\mathbf{L} : (\Omega^*)^N \times \mathbf{Z}^N \rightarrow (\Omega^*)^N$ be a function returning the *left subsequence vector* for a given sequence vector and cell co-ordinates (defined by $L(\mathbf{S}, \mathbf{C})_n = \mathcal{L}(S_n, C_n)$). Further, let $\mathbf{J} : (\Omega^*)^N \times \mathbf{Z}^N \rightarrow (\Omega^0 \cup \Omega^1)^N$ be a function returning the *maximal forward emission* into a cell (defined by $J(\mathbf{S}, \mathbf{C})_n = \mathcal{R}(L(\mathbf{S}, \mathbf{C})_n, 1)$). Finally, let $\Gamma : (\Omega^*)^N \rightarrow \{V : V \subseteq \Phi\}$ be a function returning the set of *valid states* for a given maximal emission, defined by $\Gamma(\mathbf{X}) = \{\phi : \phi \in \Phi, E(\phi)_n \lhd X_n \forall n, 1 \le n \le N\}$.

Define the *Forward DP matrix*

$$\mathcal{F}_{DP}(\mathbf{S}, \mathbf{C}, \phi) = \sum_{\pi \in \Pi(\mathtt{START}, \phi, \mathbf{L}(\mathbf{S}, \mathbf{C}))} \varrho(\pi)$$

Then

$$\mathcal{F}_{DP}(\mathbf{S}, \mathbf{C}, \phi) = \sum_{\phi' \in \Phi} \mathcal{F}_{DP}(\mathbf{S}, \mathbf{C} - \mathbf{M}(\mathbf{E}(\phi)), \phi') t(\phi', \phi) \tag{2}$$

if $\phi \in \Gamma(\mathbf{J}(\mathbf{S}, \mathbf{C}))$, and 0 otherwise. The boundary condition is $\mathcal{F}_{DP}(\mathbf{S}, \mathbf{0}, \mathtt{START}) = 1$. The Forward sum is $\mathcal{F}(\mathbf{S}) = \mathcal{F}_{DP}(\mathbf{S}, \mathbf{M}(\mathbf{S}), \mathtt{END})$. Resolution of circular dependencies in (2) is achieved by eliminating null states as described in 'ELIMINATING STATES'. The recursion can be accelerated by tabulating Γ and, of course, $\mathcal{F}_{DP}$.

Define the *Viterbi max* $\mathcal{V}(\mathbf{S}) = \max_{\pi \in \Pi(\mathtt{START}, \mathtt{END}, \mathbf{S})} \varrho(\pi)$. Again, this is computed using a *Viterbi DP matrix*

$$\mathcal{V}_{DP}(\mathbf{S}, \mathbf{C}, \phi) = \max_{\pi \in \Pi(\mathtt{START}, \phi, \mathbf{L}(\mathbf{S}, \mathbf{C}))} \varrho(\pi)$$

Then

$$\mathcal{V}_{DP}(\mathbf{S}, \mathbf{C}, \phi) = \max_{\phi' \in \Phi} \mathcal{V}_{DP}(\mathbf{S}, \mathbf{C} - \mathbf{M}(\mathbf{E}(\phi)), \phi') t(\phi', \phi) \tag{3}$$

if $\phi \in \Gamma(\mathbf{J}(\mathbf{S}, \mathbf{C}))$, and 0 otherwise, with boundary condition $\mathcal{V}_{DP}(\mathbf{S}, \mathbf{0}, \mathtt{START}) = 1$. Then $\mathcal{V}(\mathbf{S}) = \mathcal{V}_{DP}(\mathbf{S}, \mathbf{M}(\mathbf{S}), \mathtt{END})$.

Given a DP matrix and a probabilistic way of tracing back through its dependencies, we can sample a path $\pi \in \Pi(\mathtt{START}, \mathtt{END}, \mathbf{S})$ via the following *traceback algorithm*, using temporary variables $\mathbf{C} \in \mathbf{Z}^N$ and $i, j \in \Phi$:

- Begin by setting $j \leftarrow \mathtt{END}$, $\pi \leftarrow [j]$, $\mathbf{C} \leftarrow \mathbf{M}(\mathbf{S})$.
- **Loop:** Sample $i \in \Phi$ from the distribution $\rho(i|j, \mathbf{C})$ (to be defined below).
- Set $\pi \leftarrow [i] \circ \pi$, $j \leftarrow i$ and $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{M}(\mathbf{E}(i))$.
- If $i = \mathtt{START}$, then terminate; otherwise, goto **Loop**.

For Forward traceback, we use

$$\rho(i|j, \mathbf{C}) = \mathcal{F}_{DP}(\mathbf{S}, \mathbf{C}, i) t(i, j) / \sum_{k \in \Phi} \mathcal{F}_{DP}(\mathbf{S}, \mathbf{C}, k) t(k, j)$$

For Viterbi traceback, we use

$$\rho(i|j, \mathbf{C}) = \delta(i, \mathrm{argmax}_{k \in \Phi} \mathcal{V}_{DP}(\mathbf{S}, \mathbf{C}, k) t(k, j))$$

In the case of Forward traceback, the returned path is an unbiased sample from $P(\pi|\mathbf{S}) = \varrho(\pi)/\mathcal{F}(\mathbf{S})$; in the case of Viterbi traceback, the path is the one with the highest path product.

We can also compute partial derivatives $c(\mathbf{S}, i, j) = \delta(\log \mathcal{F}(\mathbf{S}))/\delta(\log t(i, j))$, which can be interpreted as probabilistic 'counts' of the number of time transition $i \rightarrow j$ was used, using the following *Forward-Backward algorithm*. Let $\mathbf{R} : (\Omega^*)^N \times \mathbf{Z}^N \rightarrow (\Omega^*)^N$ be a function returning the *right subsequence vector* for a given sequence vector and cell co-ordinates, defined such that $L(\mathbf{S}, \mathbf{C})_n \circ R(\mathbf{S}, \mathbf{C})_n = S_n$. Further, let $\mathbf{K} : (\Omega^*)^N \times \mathbf{Z}^N \rightarrow (\Omega^0 \cup \Omega^1)^N$ be a function returning the *maximal backward emission* from a cell (defined by $K(\mathbf{S}, \mathbf{C})_n = \mathcal{L}(R(\mathbf{S}, \mathbf{C})_n, 1)$).

Define the *Backward DP matrix*

$$\mathcal{B}_{DP}(\mathbf{S}, \mathbf{C}, \phi) = \sum_{\pi \in \Pi(\phi, \mathtt{END}, \mathbf{R}(\mathbf{S}, \mathbf{C}))} \varrho(\pi)$$

Then

$$\mathcal{B}_{DP}(\mathbf{S}, \mathbf{C}, \phi) = \sum_{\phi' \in \Phi} \mathcal{B}_{DP}(\mathbf{S}, \mathbf{C} + \mathbf{M}(\mathbf{E}(\phi)), \phi') t(\phi, \phi') \tag{4}$$

if $\phi \in \Gamma(\mathbf{K}(\mathbf{S}, \mathbf{C}))$, and 0 otherwise, with boundary condition $\mathcal{B}_{DP}(\mathbf{S}, \mathbf{M}(\mathbf{S}), \mathtt{END}) = 1$. The counts $c(\mathbf{S}, i, j)$ are then given by

$$c(\mathbf{S}, i, j) = \sum_{\mathbf{C}=\mathbf{0}}^{\mathbf{M}(\mathbf{S})} \frac{\mathcal{F}_{DP}(\mathbf{S}, \mathbf{C}, i) t(i, j) \mathcal{B}_{DP}(\mathbf{S}, \mathbf{C}, j)}{\mathcal{F}(\mathbf{S})} \tag{5}$$

The Baum–Welch training algorithm sets $t(i, j) \propto c(\mathbf{S}, i, j)$ and iterates to convergence (Durbin *et al.*, 1998). The counts $c(\mathbf{S}, i, j)$ can also be used by other flavours of EM algorithm (Holmes and Rubin, 2002).

The range of $\mathbf{C}$ in Equations (2)–(5) is from $\mathbf{0}$ to $\mathbf{M}(\mathbf{S})$. Note that the mapping from cell co-ordinates $\mathbf{C}$ to sequences $\mathbf{S}$ is defined entirely by the left subsequence vector function $\mathbf{L}(\mathbf{S}, \mathbf{C})$. This means we can easily restrict the DP to a subset $\Pi' \in \Pi$ of all paths (e.g. to a slice of the DP matrix corresponding to a given alignment) by changing the range of $\mathbf{C}$ and the definition of $\mathbf{L}$. Thus, if $A \in ((\Omega^0 \cup \Omega^1)^N \cap \overline{\{[]^N\}})^*$ is an *alignment* of $\mathbf{S}$, i.e. a sequence of non-null emissions $[A_1, A_2 \ldots A_{|A|}]$ (so $(A_k)_n$ is the k'th emission to the n'th sequence) satisfying $\bigcirc_{k=1}^{|A|}(A_k)_n = S_n \forall 1 \leq n \leq N$, and $H(\pi) = \bigcirc_{i:\mathbf{E}(\pi_i)\neq[]^N}[\mathbf{E}(\pi_i)]$ is the alignment for path π, then we can do dynamic programming exclusively over alignment A, i.e. over the set of paths $\Pi'(\alpha, \beta, \mathbf{S}) = \{\pi : \pi \in \Pi(\alpha, \beta, \mathbf{S}), H(\pi) = A\}$, simply by using a one-dimensional cell co-ordinate $C \in \mathbb{Z}, 0 \leq C \leq |A|$ and using $\mathbf{L}(\mathbf{S}, C)_n = \bigcirc_{k=1}^{C}(A_k)_n$ for our definition of the left subsequence vector.