5 Reconstruction

The subject of this chapter is the *reconstruction problem*, which has two versions. In both, the objective is to construct a presentation for some process given certain information about that process. In the first, which we will call *reconstruction from probabilities*, the given information is the probability the process assigns to every word in $\mathcal{X}^*$. In the second, which we will call *reconstruction from a sample*, the given information is a sample of the process's output. We will use this sample data solely to estimate probabilities of words, so reconstruction from a sample may be viewed as a form of reconstruction from probabilities in which the probabilities are only approximately known. Alternatively, reconstruction from probabilities may be thought of as an idealized form of reconstruction from a sample, in which the sample is infinitely large. In both versions, we make the assumption that the span of the process states is finite-dimensioned. In fact, we will show how to construct a GHMM presentation for any process which satisfies this condition.

A substantial body of research has accumulated around the problem of reconstruction from a sample for HMMs, for example [23–26]. Most of it involves versions of an algorithm known as forward-backward or Baum-Welsh [17,24]. These are forms of the expectation-maximization (EM) algorithm [27]. With all of these algorithms, a number of structural assumptions are required — the number of states and some choice about what transitions will be allowed to occur (for example, all may be allowed). Then a random initial presentation which satisfies the structural assumptions is chosen. The various algorithms then implicitly assume it describes the sample to some degree, and iteratively adjust the parameters while leaving the structure fixed.

In contrast, we will present a solution to the problem of reconstruction from probabilities, which appears not to have been studied before. We will follow this with an adaptation of this solution to reconstruction from a sample. The resulting reconstruction algorithm does not require its user to choose a number of states or a structure of allowed transitions, nor does it depend on any ability of random HMMs to describe the sample. Instead, it operates by estimating conditional distributions (that is, process states) from the sample and then constructs GHMM transition matrices directly from these conditional distributions.

This algorithm is new, and it is the work of the author. It should be noted, however, that other elements of this algorithm have been used before. Gilbert [20] and Dharmadhikari [10,28], consider the rank of the matrix of probabilities we call HT^kF. Given a function of a Markov chain in a certain class, Gilbert constructs a new function of a Markov chain which is conjugate to the original. His technique, like the one used here, derives a set of presentation states form a matrix of word probabilities. Crutchfield [11] uses estimated probabilities to reconstruct presentations of stochastic deterministic finite automata from samples. And the QL algorithm is a standard technique in numerical linear algebra [29].

5.1 Constructing a Presentation for a Process

The key observation in section 4.3 was this: the matrices HF and HT^kF, for all k, do not depend on a presentation. We can rewrite theorem 4.3.10 as follows: if a process $\mathcal{P} = \left(\mathcal{X}^{\mathbb{Z}}, \mathbb{X}, \mathbf{P}\right)$ has a presentation for which W and S are minimal wordlists, then $\left(W, \mathcal{X}, \left\{B^k\right\}, \gamma\right)$ is a presentation for $\mathcal{P}$. That is, we can construct a presentation for $\mathcal{P}$ almost without referring to a preexisting presentation — if we know the probabilities of the right words, we can compute all the components of $\left(W, \mathcal{X}, \left\{B^k\right\}, \gamma\right)$ from those probabilities using 4.3.10. But so far, we still need a preexisting presentation from which to construct the wordlists. To solve the problem of reconstruction from probabilities, then, we need to be able to build minimal wordlists from probabilities — that is, from a process without referring to a presentation.

It turns out that it is possible to construct suitable wordlists from probabilities of words, but no finite algorithm can do so correctly in every case. The reason is simple: a finite algorithm can only examine the probability of a finite number of words. If we build a presentation from the probabilities of a finite number of words, then it is always possible that there is a word, which was not examined, whose probability is not correctly extrapolated from the words which were examined by the presentation. Thus, the solution to the problem of reconstruction from probabilities must have an element which is either infinitary or nonconstructive. Fortunately, there is an algorithm, built on our theoretical framework, that works well in practice as we will see in the next section. And we can find suitable wordlists directly given any upper bound on the maximum word length, such as that which can be derived from the number of presentation states.

In chapter 4, we built our history wordlists so that the set of mixed states induced by their words spanned $\mathcal{H}/K_{\mathcal{F}}$. Recall that $\mathcal{U}$ is defined as the span of the recurrent process states, without reference to a presentation, and that $\mathcal{U}$ is isomorphic to $\mathcal{H}/K_{\mathcal{F}}$. To build history wordlists without referring to presentations, we will choose words that induce a set of process states spanning $\mathcal{U}$. Describing the future wordlists in an analogous fashion is a little more difficult because we do not have a concept analogous to "state" that refers to something induced by the future.* If we recall the definition of a reachable process state — a conditional distribution on the future which is induced by a history word — then the appropriate analog is clear: a conditional distribution on the past which is induced by a future word. Such a conditional distribution would be a process state if we were to reverse the direction of time, so we will call it a *reverse state*. The future wordlists we wish to build, then, consist of words which induce a set of reverse states which has the same span as the set of all reverse states. It will not be necessary to actually write reverse states in any calculations, so we will not define notation for them.

At this point, we shift to more nearly concrete objects. We need to choose an order on $\mathcal{X}^*$ — the order we choose does not matter, so long as it is fixed. A natural choice is to put shorter words before longer words, and put words of the same length in lexicographical order by some order on $\mathcal{X}$. When $\mathcal{X} = \{0, 1\}$, and 0 precedes 1, we have

$$\lambda, 0, 1, 00, 01, 10, 11, 000, \ldots \tag{5.1}$$

Let q_i be the ith word in this ordering, then we have a one-to-one correspondence between $\mathbb{N}$ and $\mathcal{X}^*$.

We can now define the infinite matrix P, whose entries are indexed by $\mathbb{N} \times \mathbb{N}$. Let $w = q_i$, $s = q_j$, and

$$P_{ij} = \begin{cases} \mathbf{P}(s|w) & \mathbf{P}(w) \neq 0 \\ 0 & \mathbf{P}(w) = 0. \end{cases} \tag{5.2}$$

For convenience and clarity, we will use q_i in place of i itself in subscripts of P and write $P_{w,s}$ instead of P_{ij}. Each row of P is labeled with a history word, each column with a future word, and each entry (except for those in all-zero rows) is a conditional probability. The top row, with history word λ, contains the unconditioned probabilities $\mathbf{P}(w|\lambda) = \mathbf{P}(w)$. So in principle the top row alone can generate the whole matrix. Each

*There is a time-symmetric development of our approach that we have chosen not to present here.

row contains a complete description of a reachable process state, in the form of all the conditional probabilities on future words. Each column contains a complete description of a reverse state, although some renormalization is needed to put the column into the right form. Thus, any word w identifies both a unique row $P_{w,\cdot}$ and a unique column $P_{\cdot,w}$. We will denote the row associated with w by $r(w)$ and the column associated with s by $c(s)$.

The following definition is analogous to definition 4.2.4 of sufficient wordlists for a GHMM.

Definition 5.1.1. A history wordlist W is *sufficient for a process* if the rows identified by the words in W span all the rows of P. Similarly, a future wordlist S is sufficient for a process if the columns identified by the words in S span all the columns of P.

Now, a pair of wordlists W and S identifies a submatrix of P in a natural way: if $W = \{w_1, \ldots, w_n\}$ and $S = \{s_1, \ldots, s_k\}$, we define the submatrix G of P by $G_{ij} = P_{w_i,s_j}$, so that $G_{ij} = \mathbf{P}(s_j|w_i)$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, l$. That is, we pick out the elements of P which are both in the rows identified by W and in the columns identified by S. If we had a presentation for $\mathcal{P}$ and we built the wordlists W and S and the matrices H and F for that presentation, we would find that $G = HF$ (see equation 4.40). In the same manner, we define a submatrix C^k of P for each $k \in \mathcal{X}$ by picking out the rows of P corresponding to words in W and the columns corresponding to words ks_j for $s_j \in S$. That is, C^k is defined by $C^k_{ij} = P_{w_i,ks_j} = \mathbf{P}(ks_j|w_i)$. Thus, if we had a presentation for $\mathcal{P}$, we would have $C^k = HT^kF$.

In corollary 4.2.9, we established that if W and S are minimal for a GHMM, then HF is invertible. And we know that if either W or S is sufficient but not minimal, then HF is larger in size — but not in rank — than it would be if W and S were minimal. Thus if W and S are sufficient, HF is invertible if and only if W and S are both minimal. We will use the analogous version of this for the following definition.

Definition 5.1.2. A pair of wordlists W and S are *minimal for a process* $\mathcal{P}$ if they are sufficient for $\mathcal{P}$ and the matrix G they define is invertible.

The next few results establish that sufficiency and minimality for a process have the properties we will need in theorem 5.1.8, which is the main result of this section. These

properties are roughly analogous to the properties of sufficient and minimal wordlists for a GHMM, though we will not attempt to draw precise analogies.

We are interested in the number of linearly independent rows and columns of P. If P were finite, we would refer to its rank. Thus we give the following definition for rank in this infinite context.

Definition 5.1.3. The *rank* of P is defined to be the supremum of the ranks of the finite submatrices of P.

Lemma 5.1.4. The rank of P is equal to the dimension of the span of $\mathcal{U}$.

Because of the close connection between the rows of P and the reachable process states, this is almost automatic. The only difficulties are in dealing with the infinitely many columns in those rows.

Proof. Let $n = \dim(\mathcal{U})$, which we will assume for the moment is finite. We can choose $\{w_1, \ldots, w_n\}$ such that if $\mathbf{A}_i = \mathbf{P}(\cdot|w_i)$ for $i = 1, \ldots, n$, the process states $\mathbf{A}_1, \ldots, \mathbf{A}_n$ span $\mathcal{U}$. Then for all $w \in \mathcal{X}^*$, there are numbers $a_1, \ldots, a_n$ such that

$$\mathbf{P}(\cdot|w) = a_1\mathbf{A}_1 + \ldots + a_n\mathbf{A}_n. \tag{5.3}$$

so for all s, $\mathbf{P}(s|w) = \sum_i a_i\mathbf{A}_i(s)$. This implies that the row $r(w)$ can be written as $r(w) = a_1 r(w_1) + \ldots + a_n r(w_n)$. So the rows $r(w_1), \ldots, r(w_n)$ form a basis for the rows of P. Thus, any collection of more than n rows is linearly dependent, and the same must be true for the rows of submatrices. This shows that rank$(P) \leq n$.

Conversely, because $\mathbf{A}_1, \ldots, \mathbf{A}_n$ are linearly independent and the row $r(w_i)$ determines the distribution $\mathbf{A}_i = \mathbf{P}(\cdot|w_i)$, $r(w_1), \ldots, r(w_n)$ must be linearly independent. So the row space of P has dimension n. What remains to be shown is that P has a finite submatrix of rank n.

For any word s, let $c'(s)$ be the length n column vector consisting of those elements of $c(s)$ which lie in rows $r(w_1), \ldots, r(w_n)$; that is,

$$c'(s) = \begin{pmatrix} \mathbf{P}(s|w_1) \\ \vdots \\ \mathbf{P}(s|w_n) \end{pmatrix} = \begin{pmatrix} P_{w_1,s} \\ \vdots \\ P_{w_n,s} \end{pmatrix}. \tag{5.4}$$

We will call $c'(s)$ a *subcolumn* of P. Choose a wordlist S such that $c'(s_1), \ldots, c'(s_l)$ is a linearly independent basis for the span of all subcolumns $c'(s)$. Let G be the $n \times l$

submatrix of P given by W and S. The columns of G are exactly the subcolumns $c'(s_1), \ldots, c'(s_l)$, so G has rank l. We have shown the rank of G must be less than n, so $l \leq n$.

We now define the subrow r' in a manner analogous to the definition of a subcolumn. The subrow $r'(s)$ is the length l row vector $(P_{s_1,w}, \ldots, P_{s_l,w})$. We will reserve the terms *subrow* and *subcolumn* for these subsets of P, and we will use the terms *row vector* and *column vector* for other row and column vectors, including linear combinations of subrows and subcolumns.

Suppose that $l < n$. Then there exists a row vector $(a_1, \ldots, a_n)$, not all components of which are zero, such that the product $(a_1, \ldots, a_n)G$ is a linear combination of subrows which is the zero row vector. Now, for any $s \in \mathcal{X}^*$, the subcolumn $c'(s)$ is a linear combination of $c'(s_1), \ldots, c'(s_l)$, so if the subrow $r'(w_i)$ is zero, the ith element of each $c'(s_j)$ must be zero, and so the ith element of every subcolumn must be zero. That is, if a subrow is zero, the corresponding entire row must be zero. The same must be true for linear combinations of subrows: for any s the element in column s of the infinite vector $a_1 r(w_1) + \ldots + a_n r(w_n)$ is a linear combination of elements of the row vector $(a_1, \ldots, a_n)G$. Thus, if $(a_1, \ldots, a_n)G$ is all zeros, then every element of $a_1 r(w_1) + \ldots + a_n r(w_n)$ is a linear combination of zeros, and so

$$a_1 P_{w_1,s} + \ldots + a_n P_{w_n,s} = 0, \tag{5.5}$$

for all words s.

But the rows $r(w_1), \ldots, r(w_n)$ have been shown to be linearly independent, so $a_1 r(w_1) + \ldots + a_n r(w_n)$ cannot be equal to a row of zeros. Thus, there is a word s such that

$$a_1 P_{w_1,s} + \ldots + a_n P_{w_n,s} \neq 0. \tag{5.6}$$

This is a contradiction. Therefore, $n = l$ and G is a square submatrix of P which has rank n.†

†We may interpret this in the following way: the process states of the forward and time-reversed processes have span sets of the same dimension. Thus, the minimal GHMM presentations for the forward and time-reversed processes have the same number of presentation states.

Lastly, if $n = \dim(\mathcal{U})$ is infinite, then for any m we can find linearly independent process states $\mathbf{A}_1, \ldots, \mathbf{A}_m$. Proceeding as above, we can construct a submatrix G or P with rank m. Thus there is no upper bound to the ranks of the submatrices of P, so the rank of P is infinite.■

The first part of the following result has essentially already been proven. The second part establishes that minimal wordlists exist.

Proposition 5.1.5. For any process $\mathcal{P}$, let n be the rank of P. If n is finite, then

1. There exists an invertible $n \times n$ submatrix of P, and
2. If G is any such matrix, then the wordlists W and S that produce it are minimal for $\mathcal{P}$.

Proof. In the proof of lemma 5.1.4, we constructed an $n \times l$ matrix G which had rank l, and showed than $l = n$. Thus G is a square matrix of full rank, and must be invertible, which proves part 1. Note that the invertibility of G is also part of what it means to be minimal for a process. Thus, to prove part 2, we need only show that W and S are sufficient, because G is invertible. That is, we need only show that $r(w_1), \ldots, r(w_n)$ span all rows of P, and that $c(s_1), \ldots c(s_n)$ span all columns of P. As in the proof of lemma 5.1.4, we will use $c'(s)$ to denote the subcolumn of P,

$$c'(s) = \begin{pmatrix} P_{w_1,s} \\ \vdots \\ P_{w_n,s} \end{pmatrix}. \tag{5.7}$$

Thus, $c'(s_j)$ is the jth column of G. Likewise, we will use $r'(w)$ to denote the subrow $r'(w) = (P_{ws_1}, \ldots, P_{ws_n})$, so that $r'(w_i)$ is the ith row of G.

Let w be any word not in W. The set of subrows $\{r'(w),\ r'(w_1), \ldots, r'(w_n)\}$ contains $n+1$ vectors of length n, hence they are not linearly independent. But $r'(w_1), \ldots, r'(w_n)$ are linearly independent, so $r'(w)$ must be a linear combination of them. That is, there is a vector a such that $aG = r'(w)$. Because G is invertible, we have $a = r'(w)G^{-1}$; that is, there is exactly one such a.

Now let s be any word not in S, and consider the $(n+1) \times (n+1)$ submatrix M of P given by

$$M = \begin{pmatrix} G & c'(s) \\ r'(w) & P_{ws} \end{pmatrix}. \tag{5.8}$$

We know that rank$(P) = n$, so M must be singular. The first n rows of M are linearly independent, so the last row must be a linear combination of them. That is, there must be a vector b such that $bG = r'(w)$ and $bc'(s) = P_{w,s}$. Now we have just shown that there is a unique vector a such that $aG = r'(w)$, and this a clearly does not depend on s. So we must have $a = b$, and $ac'(s) = P_{w,s}$. Furthermore, this must hold for all s. Thus the entire row $r(w)$ of P satisfies

$$r(w) = a_1 r(w_1) + \ldots + a_n r(w_n), \tag{5.9}$$

so we have shown that all rows of P lie in the span of $r(w_1), \ldots, r(w_n)$.

A similar argument shows that the columns $c(s_1), \ldots, c(s_n)$ span all columns of P.■

The next lemma completes the development of minimal wordlists for a process.

Lemma 5.1.6. Let $\mathcal{P}$ be any process such that the rank n of P is finite. If W and S are minimal wordlists for $\mathcal{P}$, then both W and S have length n.

Proof. If W and S are minimal wordlists for $\mathcal{P}$, then G is an invertible submatrix of P. The rank of G is at most n, and so W and S (which must have the same length because invertible matrices must be square) have length at most n. Furthermore, there are sets of n rows which are linearly independent, all of which must lie in the span of the rows identified by W. So W must have length n.■

We need one more lemma before we are ready for theorem 5.1.8. This one is a calculation, most of which appeared in the proof of proposition 5.1.5.

Lemma 5.1.7. Given a process $\mathcal{P}$ and a matrix G defined by minimal wordlists, for any $w, s \in \mathcal{X}^*$, we have

$$r'(w)G^{-1}c'(s) = \mathbf{P}(s|w). \tag{5.10}$$

Proof. As in the proof of proposition 5.1.5, let

$$M = \begin{pmatrix} G & c'(s) \\ r'(w) & P_{w,s} \end{pmatrix}. \tag{5.11}$$

M must be singular in such a way that there is a vector a satisfying

1. $aG = r'(w)$, and
2. $ac'(s) = P_{w,s}$.

And because G is invertible, we must have $a = r'(w)G^{-1}$. Thus we have

$$r'(w)G^{-1}c'(s) = P_{w,s} = \mathbf{P}(s|w). \blacksquare \tag{5.12}$$

Note that if we let $w = x$ and $s = \lambda$, lemma 5.1.7 gives us $r'(x)G^{-1}c'(\lambda) = \mathbf{P}(\lambda|x)$. But $c'(\lambda) = \vec{1}$ and $\mathbf{P}(\lambda|x) = 1$ for any x, so for all x we have

$$r'(x)G^{-1}\vec{1} = 1. \tag{5.13}$$

We have now completed the minor results concerning minimal wordlists for processes, and are ready to state and prove the main result of this section. We will use δ_i to represent a vector with a one in the ith position and zeros in all other positions. This symbol represents a row vector when it appears to the left of a matrix and a column vector when it appears to the right of a matrix.

Recall that we have defined C^k by $C^k_{ij} = P_{w_i,ks_j}$, and that if we have arbitrary wordlists and any presentation for $\mathcal{P}$, we have $G = HF$ and $C^k = HT^kF$. Note that C^k satisfies the following:

$$\begin{aligned} C^k_{ij} &= \mathbf{P}(ks_j|w_i) \\ &= \mathbf{P}(k|w_i)\mathbf{P}(s_j|w_ik) \\ &= \mathbf{P}(k|w_i)P_{w_ik,s_j}. \end{aligned} \tag{5.14}$$

so the ith row of C^k satisfies

$$\delta_i C^k = \mathbf{P}(k|w)r'(w_ik) \tag{5.15}$$

Theorem 5.1.8. If $\mathcal{P}$ is any process for which the dimension of the span of $\mathcal{U}$ is finite, and W and S are minimal wordlists for $\mathcal{P}$, then $\left(W, \mathcal{X}, \left\{B^k\right\}, \gamma\right)$ is a GHMM presentation for $\mathcal{P}$, where $B^k = C^kG^{-1}$ and $\gamma = (\mathbf{P}(s_1), \ldots \mathbf{P}(s_n))G^{-1}$.

Proof. To prove that $\left(W, \mathcal{X}, \left\{B^k\right\}, \gamma\right)$ is a presentation for $\mathcal{P}$, we must show $\sum_k B^k$ is unit–sum and that $\gamma B^x\vec{1} = \mathbf{P}(x)$ for all $x \in \mathcal{X}^*$. The first of these is a calculation. By manipulating the definition of the B^ks, we get

$$\left(\sum_k B^k\right)\vec{1} = \left(\sum_k C^k\right)G^{-1}\vec{1}. \tag{5.16}$$

We will work with the ith row alone, and use equation 5.15:

$$\delta_i \left(\sum_k B^k \right) \vec{1} = \delta_i \left(\sum_k C^k \right) G^{-1} \vec{1} = \sum_k \mathbf{P}(k|w_i) r'(w_i k) G^{-1} \vec{1}. \tag{5.17}$$

Applying equation 5.13, we get

$$\begin{aligned} \delta_i \left(\sum_k B^k \right) \vec{1} &= \sum_k \mathbf{P}(k|w_i) \\ &= 1. \end{aligned} \tag{5.18}$$

Since this is true for each i, we have shown that $\sum_k B^k$ is unit-sum: $\left(\sum_k B^k \right) \vec{1} = \vec{1}$.

Showing that

$$\gamma B^x \vec{1} = \mathbf{P}(x) \tag{5.19}$$

holds for all x is more involved. We will prove this by proving that for all x,

$$\gamma B^x = \mathbf{P}(x) r'(x) G^{-1}. \tag{5.20}$$

From this multiplying on the right by $\vec{1}$ and applying equation 5.13 gives us equation 5.19.

We will establish equation 5.20 by inductively concatenating symbols to form an arbitrary word x. The base case is trivial — B^γ is the identity matrix, $\mathbf{P}(\lambda) = 1$, and $\gamma = r' G^{-1}$ by definition. So what remains to be shown is the induction case: given a word x and a symbol k, assume that $\gamma B^x = \mathbf{P}(x) r'(x) G^{-1}$ and prove that $\gamma B^x B^k = \mathbf{P}(xk) r'(xk) G^{-1}$.

Consider the jth coordinate of $\mathbf{P}(k|x) r'(xk)$: we have

$$\begin{aligned} \mathbf{P}(k|x) r'(xk) \delta_j &= \mathbf{P}(k|x) P_{xk,s_j} \\ &= \mathbf{P}(k|x) \mathbf{P}(s_j|xk) \\ &= \mathbf{P}(k s_j|x). \end{aligned} \tag{5.21}$$

Replacing the right hand side using lemma 5.17, we have

$$\mathbf{P}(k|x) r'(xk) \delta_j = r'(x) G^{-1} c'(k s_j). \tag{5.22}$$

Now, note that $C^k_{ij} = P_{w_i, k s_j}$ is the ith row of $c'(k s_j)$, so that for all i,

$$\delta_i c'(k s_j) = C^k_{ij} = \delta_i C^k \delta_j. \tag{5.23}$$

Thus, we have $C^k\delta_j = c'(ks_j)$, and equation 5.22 becomes

$$\mathbf{P}(k|x)r'(xk)\delta_j = r'(x)G^{-1}C^k\delta_j. \tag{5.24}$$

This holds for all j, so we have

$$\mathbf{P}(k|x)r'(xk) = r'(x)G^{-1}C^k. \tag{5.25}$$

Next we multiply both sides by $\mathbf{P}(x)$ on the left and G^{-1} on the right and then simplify:

$$\begin{aligned} \mathbf{P}(x)\mathbf{P}(k|x)r'(xk)G^{-1} &= \mathbf{P}(x)r'(x)G^{-1}C^kG^{-1} \\ \mathbf{P}(xk)r'(xk)G^{-1} &= \mathbf{P}(x)r'(x)G^{-1}B^k. \end{aligned} \tag{5.26}$$

Finally, we use the induction hypothesis $\gamma B^x = \mathbf{P}(x)r'(x)G^{-1}$ and we get

$$\mathbf{P}(xk)r'(xk)G^{-1} = \gamma B^xB^k, \tag{5.27}$$

and the proof is complete.■

We summarize the constructive portion of the preceding development as follows.

Algorithm 5.1.9. Let $\mathcal{P} = \left(\mathcal{X}^{\mathbb{Z}}, \mathbb{X}, \mathbf{P}\right)$ be a process satisfying the condition that the span of its process states is finite-dimensional. A GHMM presentation for $\mathcal{P}$ may be constructed by the following steps.

1. Construct the infinite matrix P with entries $P_{w,s} = \mathbf{P}(s|w)$.
2. Find a nonsingular minor G of P such that no other minor of P has rank greater than the rank of G.
3. Build the history wordlist $W = \{w_1, \ldots, w_n\}$ by defining w_i to be the word associated with the ith row of P. Build the future wordlist $s = \{s_1, \ldots, s_n\}$ by defining s_j to be the word associated with the jth column of P.
4. For each $k \in \mathcal{X}$, construct the matrix C^k with entries $C^k_{ij} = \mathbf{P}(ks_j|w_i)$.
5. For each $k \in \mathcal{X}$, compute $B^k = C^kG^{-1}$.
6. Compute $\gamma = (\mathbf{P}(s_1), \ldots, \mathbf{P}(s_n))G^{-1}$.

As we will see in section 5.2, a reconstruction program based on this theoretical framework has been developed. When it is given a set of word probabilities produced by a GHMM it reliably constructs a GHMM which accurately reproduces those word probabilities.

We conclude this section with the converse of theorem 4.16, which says that every process that has a GHMM presentation satisfies $\dim(\mathcal{U}) < \infty$. And if conjecture 6.1.1 is true, it is also a converse of theorem 3.6.1, which says the same thing for HMMs.

Corollary 5.1.10. Every process such that the dimension of the span of its reachable process states is finite has a GHMM presentation.

Proof. Let $\mathcal{P}$ be any process for which the span of $\mathcal{U}$ is finite dimensional. Lemma 5.1.4 tells us that P has finite rank, and proposition 5.1.5 then establishes that finite minimal wordlists exist. Theorem 5.1.8 gives us a GHMM presentation in terms of these wordlists.■

Together, theorem 4.16 and corollary 5.1.10 prove the following characterization of the class of processes represented by GHMMs.

Theorem 5.1.11. A process has a GHMM presentation if and only if the dimension of the span of its reachable process states is finite.

5.2 Reconstruction from a Sample

Suppose we are given a finite sequence of symbols and we are told that it is a sample of output from a process $\mathcal{P}$. How can we construct a presentation for this process? This is the problem of *reconstruction from a sample.* We will consider samples which consist of a single sample sequence of length L, such as

$$01101\ldots10, \tag{5.28}$$

and also samples which consist of several sample sequences of total length L, like

$$11110\ldots11,\ 0111\ldots0,\ \text{and}\ 010110\ldots001. \tag{5.29}$$

It should be apparent to the reader that this problem is of a different character than the problem of reconstruction from probabilities. Any given finite sample could have been generated by any one of an infinite number of processes, so the problem cannot be solved in any absolute sense. The best we can possibly do is to give a presentation for a process which would be likely to generate this sample, and consider this presentation to represent a new process that is an approximation to $\mathcal{P}$. Thus, we

must heed statistical issues such as variances of parameter estimates, and how much data is available. Further, because this is a question which can be asked for real data in a practical setting, we will be interested in the computational issues of operation counts and storage needs. These issues are discussed in subsections 2 and 3, following the description of the reconstruction algorithm.

The Algorithm Our approach to reconstruction from a sample estimates conditional probabilities of various words from the relative frequencies of those words in the data. It then assembles these probabilities into a truncated (finite size) estimate $\hat{P}$ of P. From here we will proceed as in algorithm 5.1.9, substituting $\hat{P}$ for P and adapting the algorithm so that it works with the finite size and imperfect estimation of $\hat{P}$.

Our first task, then, is to construct $\hat{P}$, for which we need an estimator and a pair of wordlists. Let r and r' be the lengths of the longest history and future words which we will consider, respectively. Define a *cutpoint* to be a position between to consecutive symbols in a sample sequence which is at least r symbols from the beginning and r' symbols from the end of the sample sequence. That is, a cutpoint is a time at which we know the immediate history and future words of lengths at least r and r', respectively. For any pair of words w and s, let b be the number of cutpoints in all sample sequences preceded by w, and let a be the number of cutpoints preceded by w and followed by s. Our estimate for $\mathbf{P}(s|w)$, which we will denote $\pi(s|w)$, is given by $\pi(s|w) = a/b$. We use the conventions that $\pi(s|w) = 0$ if $b = 0$ and that if $w = \lambda$, b is the number of cutpoints in the sample. This is simply a frequency substitution estimate, and it has mean $\mathbf{P}(s|w)$. For any sample, and for any choice of r, the estimates $\pi(s|w)$ for different pairs of words w and s are consistent with each other. (For example, for any w, s, and x, $\pi(ws|x) = \pi(w|x) \cdot \pi(s|w)$.)

The wordlists we will use to construct $\hat{P}$ will not be minimal in any sense. We will make them as large as possible to make sure they are sufficient. We are limited by our data in what words we can include. (If we fix b, the variance of $\pi(s|w)$ can be shown to be $\mathbf{P}(s|w)(1 - \mathbf{P}(s|w))/b$, so the estimate is of little value if b is too small. If a is too small, on the other hand, the standard deviation becomes large relative to a/b even if b is large.) Thus, we will need to select a large set of words which occur reasonably often. The precise method by which we do this may be rather ad-hoc, as

all reasonable methods will produce similar sets of words. This is because they will all include short words with well estimated probabilities and will exclude long words with poorly estimated probabilities. The essential requirement is that a balance must be struck between making the wordlists large and making the variances of the estimates small. For simplicity, we will use the following heuristic: let l be the largest natural number such that K times the total number of words of length $2l+1$ that occur in the sample is less than the total number of cutpoints in the sample, for some fixed constant K. This means that those words of length $2l+1$ that have at least one occurrence in the sample occur an average of more than K times. We choose both of our wordlists to be the set of all words of length l or less that occur in the sample. Let Y denote this set.

Now, with our wordlists chosen, we construct the $|Y| \times |Y|$ matrix $\overline{P}$ from history wordlist Y and future wordlist Y just as we constructed G from the minimal wordlists W and S in section 5.1. That is, let $\overline{P}$ consist of one row and one column for each word in Y, so that for all $w, s \in Y$, $\overline{P}$ has an element $\overline{P}_{w,s} = P_{w,s} = \mathbf{P}(s|w)$. $\overline{P}$ contains those unknown true probabilities we want to estimate. We define $|Y| \times |Y|$ matrix $\hat{P}$ by replacing each conditional probability $\overline{P}_{w,s} = \mathbf{P}(s|w)$ with its estimate $\hat{P}_{w,s} = \pi(s|w)$ for all $w, s \in Y$. Thus, $\hat{P}$ is a stochastic matrix which may be thought of as an estimate of $\overline{P}$.

Hopefully $\hat{P}$ is large enough, by which we mean that Y contains sufficient history and future wordlists for the process $\mathcal{P}$. If it is not, then the presentation we produce will represent only a poor approximation to $\mathcal{P}$, and the available data is probably insufficient to induce a good approximation. In this case, the estimates in $\hat{P}$ have sufficiently small variances, but some essential behavior of the process cannot be deduced from the probabilities we have estimated. If we make Y larger, the probabilities we attempt to estimate capture the essential behavior, but our estimates might have variance large enough to make them meaningless. If we followed section 5.1 exactly, the next step would be to find a minor G of $\hat{P}$ which has the same rank as $\hat{P}$. Because we have chosen Y as large as possible, we expect it to contain minimal wordlists W and S as proper subsets, and we expect $\hat{P}$ to have a rank much less than its size.

However, this is probably not the case. If Y is big enough, $\overline{P}$ will have a rank much less than its size; but $\hat{P}$ will not. Because each $\hat{P}_{w,s}$ is a random variable with nonzero variance, $\hat{P}$ is a random matrix close to the singular matrix $\overline{P}$. Generically,

such a matrix will be nonsingular, but ill-conditioned. We want G to have the same rank as $\overline{P}$ and to be well-conditioned; because the rank of $\overline{P}$ is unknown, we will make G as large a well-conditioned submatrix of $\hat{P}$ as possible. For this reason, the problem of finding a suitable G is itself an estimation problem.

We solve it as follows. We decompose $\hat{P}$ by the QL algorithm with pivoting. The QL decomposition is related by transposes to the more familiar QR decomposition, and both are standard techniques in numerical linear algebra [29]. This algorithm builds a basis for the row space of $\hat{P}$, starting with an empty basis, and adding one vector at a time. At each step, it computes the distance from each row vector to the subspace spanned by the developing basis. It selects the row with the greatest distance, and adds a vector derived from it to the basis, in such a way that the basis now spans the selected row. As the reader may deduce, each row is selected at most once and the distances for the selected rows decrease as more rows are selected.

As a by-product, the QL algorithm lists the rows of the matrix in the order in which they were selected, and it gives the distance computed for each row at the time it was selected. These distances tell us how significant each row is and thus, how significant each basis vector is. We choose a threshold θ for these distances and discard the rows with distances less than this threshold. These discarded rows with their small distances make $\hat{P}$ ill-conditioned rather than singular, and their contributions are likely to result from stochastic (sample) variation rather than reflecting the true probabilities in $\overline{P}$. How we choose this threshold θ is necessarily somewhat arbitrary. We use a second heuristic that attempts to draw the threshold just above the largest distance resulting from the variance of the estimates $\hat{P}_{w,s}$.

We build our history wordlist W by collecting the words that induce the rows we keep. This step dictates the number $n = |W|$ of presentation states in the presentation we will reconstruct. Finally, we apply the QR algorithm (related to QL by transposes) to this edited matrix and select the columns with the n largest distances. The result is that we are left with a square matrix G and a wordlist S. The construction of G guarantees that it is invertible and well-conditioned.

From here, we construct the matrices C^k such that $C^k_{ij} = \pi(ks_j|w_i)$ and let $B^k = C^k G^{-1}$ as in section 5.1. And if we let $p_i = \pi(s_i|\lambda)$ for each word s_i in the future wordlist S, we can set $\gamma = (p_1, \ldots, p_n)G^{-1}$. The same calculations we used in the proof

of theorem 5.1.8 show that $\left(W, \mathcal{X}, \left\{B^k\right\}, \gamma\right)$ satisfies all the necessary conditions and is in fact a Proto-GHMM. This finishes the construction of the presentation, which we summarize as follows.

Algorithm 5.2.1. If we are given one or more sample sequences of output from a process $\mathcal{P} = \left(\mathcal{X}^{\mathbb{Z}}, \mathbb{X}, \mathbf{P}\right)$ for which $\mathbf{P}$ is unknown, we may construct a Proto-GHMM $W, \mathcal{X}\left\{B^k\right\}, \lambda$ which approximately represents $\mathcal{P}$ by the following steps.

1. Construct a matrix of estimated word probabilities. This may be done as follows.
   a. Fix a number K and choose the largest l such that each word of length $2l+1$ occurs an average of K times.
   b. Let Y be the set of all words of length l or less.
   c. For each $w, s \in Y$, compute
   $$\hat{P}_{w,s} = \frac{\text{the number of times } ws \text{ occurs}}{\text{the number of times } w \text{ occurs}}. \tag{5.30}$$
2. Find a nonsingular, well-conditioned minor G of P. One way to do this is as follows.
   a. Perform a QL decomposition of $\hat{P}$ to compute a significance for each row.
   b. Fix a threshold θ and discard all rows of $\hat{P}$ with significance less than θ. Let n be the number of rows remaining.
   c. Perform a QR decomposition of the remainder of $\hat{P}$ and select the n columns with the highest significance. These subcolumns of $\hat{P}$ comprise the matrix G.
3. Let the history wordlist W be the set of words associated with the rows of G, and let the future wordlist S be the set of words associated with the columns of G.
4. For each $k \in \mathcal{X}$, construct the matrix C^k with entries $C^k_{ij} = \pi(ks_j|w_i) = \hat{P}_{w_i, ks_j}$
5. For each $k \in \mathcal{X}$, compute $B^k = C^k G^{-1}$.
6. For each $s_i \in S$, compute an estimate $\pi(s_i)$ of $\mathbf{P}(s_i)$.
7. Compute $\gamma = (\pi(s_1), \ldots, \pi(s_n))G^{-1}$.

This computes all the parts of a Proto-GHMM $\left(W, \mathcal{X}, \left\{B^k\right\}, \gamma\right)$. If this Proto-GHMM is valid, it is a GHMM presentation for a process which has word probabilities close to those of $\mathcal{P}$.

This construction is not completely satisfactory, however. We cannot assert that $\left(W, \mathcal{X}, \left\{B^k\right\}, \gamma\right)$ is a GHMM because we have not shown that it is valid. In fact it

is possible to construct a sample from which a reconstruction yields an invalid Proto-GHMM. As mentioned in section 4.1, determining whether or not a given presentation is valid is quite difficult. We will address this further in section 6.2.

Statistical Considerations Algorithm 5.1.1 produces a Proto-GHMM as a function of a sample. If the sample is a random sample of the output from a process $\mathcal{P}$, the sample is a random variable, and so the Proto-GHMM is also a random variable. Thus the reconstruction algorithm, together with $\mathcal{P}$ and the length of the sample, induce a distribution on the space of Proto-GHMMs.

We would like to be able to describe this distribution, but doing so is quite complicated. One of the difficulties is describing the distribution of $\hat{P}$. Each entry $\pi(s|w)$ in $\hat{P}$ has the form $\frac{X}{Y}$, where X appears to have a binomial distribution, with parameter $\theta = \mathbf{P}(s|w)$ and Y trials, and Y is the random variable describing the number of occurrences of the word w. However, X may not be binomial, because occurrences of s may not be independent of previous occurrences. Similarly, the distribution of Y cannot be described easily. Moreover, entries of $\hat{P}$ are not independent. Although it may appear that the joint distributions of some groups of entries may be a function of a multinomial distribution, this is not the case. Our "trials" are not independent, as they come from examining pieces of the sample sequences, and these pieces may overlap. For example, if the alphabet is $\mathcal{X} = \{0, 1\}$ and the sample consists of a single sequence, then the numbers of occurrences of 01 and 10 may differ by at most one. This complicates any description of the distribution of $\hat{P}$. In addition, even the size of $\hat{P}$ depends on the sample. Although it may be possible to characterize the distribution of $\hat{P}$ in a tractable way, doing so is beyond the scope of this dissertation.

If we were able to express this distribution reasonably, we would manipulate it further in order to derive the distribution it induces on the set of all Proto-GHMMs. To do this, we would look at the output of the QL algorithm as a random variable and examine its distribution. Continuing in this way, we would derive distributions for G, the C^ks, and eventually for γ and the B^ks. However, we cannot proceed with this program because we cannot describe the distribution of $\hat{P}$. Most of the steps would be laborious, and probably not very interesting.

The distribution of the output of QL raises two interesting questions in mathematical statistics. First, given a random matrix $\hat{A}$ which is the sum of an unknown singular matrix A and a random matrix (with zero mean and some assumptions about its variance), how can we best estimate the rank of A? Second, how ill-conditioned can A be — or how small must the variance of the random matrix be — so that we can reliably estimate the rank of A?

Computational Issues The computational issues for the reconstruction algorithm are the numbers of operations and the storage requirements of this algorithm. As is common practice, we will be concerned only with how these quantities scale — their *order* — and not with precise estimation.

First, we need notation for the parameters of the problem. We have used L as the length of the sample, and $m = |\mathcal{X}|$ as the size of the alphabet. Let Y be the large wordlist used to construct $\hat{P}$, and let N be the number of words in Y — this means that $\hat{P}$ is $N \times N$. Let l be the length of the longest word in Y, and let n be the number of words in the minimal wordlists we derive. These parameters are not all independent; we must have $n < N$ and may reasonably expect, assuming the entropy rate of the process is close to the maximum possible, that $l \approx \log_m N$.

To construct the estimates, we need to count the number of occurrences of each word of the form ws for w and s in Y. To do so, we step through the sample. At each step, and for each length up to $2l$, we identify the word of that length which starts at our present location in the sample, and add one to that word's count. (This technique requires minor adjustments involving the ends of sample sequences to produce the estimates exactly as we defined them in terms of cutpoints on page 95.) The number of locations is essentially L, and the number of words at each location is $2l$. The work to be done for each word may be organized so that it has constant time. Thus, the number of operations required to do the estimation has order $\mathcal{O}(lL)$.

The storage requirements of estimation are simple. To estimate $\hat{P}$, we must store one counter for each of N^2 words. In addition, the matrices C^k depend on the counts for words of the form wks, with w and s in Y and $k \in \mathcal{X}$. It is convenient to do the necessary counts for words of length up to $2l+1$ at the same time. This does not change the order of the number of operations, and it increases the storage to $\mathcal{O}\big(N^2 m\big)$.

Constructing minimal wordlists, and a basis for the mixed states, is the next stage of reconstruction. The QL algorithm we use to do this requires $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ storage. In practice, this is the most time-consuming part of the algorithm. Computing G^{-1} takes $\mathcal{O}(n^3)$ operations, which is dominated by N^3 and so may be ignored.

Finally, the algorithm constructs the matrices C^k and performs the multiplications $C^k G^{-1}$. There are m of these multiplications, each a product of $n \times n$ matrices, so this stage requires $\mathcal{O}(n^3 m)$ operations and $\mathcal{O}(n^2 m)$ memory. This last quantity is dominated by the storage requirements of the estimation stage, and we will ignore it. Thus, the entire reconstruction algorithm requires

$$\mathcal{O}(lL + N^3 + n^3 m) \tag{5.31}$$

operations and $\mathcal{O}(N^2 m)$ memory.

Of the parameters in expression 5.31, only L and m are known in advance. The number of states n is difficult to estimate — indeed, a good part of the work of the reconstruction algorithm is in estimating it. However, m is usually small enough and n is usually sufficiently smaller than N that $n^3 m$ is small compared to N^3. Thus, the number of operations does not scale strongly with the eventual number of presentation states.

With the heuristic we used to choose Y, the remaining two parameters are related by $l \approx \log N$. This will necessarily be true of any method for choosing Y, since the number of possible words of length l or less is $(m^{l+1} - 1)/(m - 1) \approx m^l$. Thus, l changes slowly compared to N. Because of this and because expression 5.31 depends linearly on l but on the cube of N, we see that N is far more important to the scaling of the operation count.

The heuristic for Y (page 96) gives us a way to estimate l and N. It chooses l so that the words of length $2l+1$ occur an average of K times. There are about L opportunities for such words to occur, so there must be about L/K such words. Assuming all m^{2l+1} possible words of length $2l + 1$ appear in the sample, we have $m^{2l+1} \approx L/K$, or

$$m^l \approx \left(\frac{L}{K}\right)^{\frac{l}{2l+1}} \approx \sqrt{\frac{L}{K}}. \tag{5.32}$$

Taking the base m of both sides gives us the approximation $l \approx \log_m \sqrt{L/K}$. Assuming all possible words of length l occur in the sample, the left-hand side of equation 5.32 is

an estimate of the N produced by the heuristic, so we have $N \approx \sqrt{L/K}$. With these substitutions, the expression 5.31 becomes

$$\mathcal{O}\left(L \log_m \frac{L}{K} + \left(\frac{L}{K}\right)^{\frac{3}{2}} + n^3 m \right) \tag{5.33}$$

and we see that the reconstruction algorithm takes order $L^{\frac{3}{2}}$ operations. Similarly, it requires memory of $\mathcal{O}(mL/K)$.

We conclude this computational analysis a few final comments. First, recall that there is more than one possible choice of the heuristic for selecting Y. The conclusion we have just reached, that the reconstruction algorithm takes time of order $L^{\frac{3}{2}}$, depends explicitly on the choice. Other choices may give higher or lower exponents, and may make the algorithm as a whole better or worse. We have no reason to believe that the heuristic we have used is a particularly good one.

Second, it is worth mentioning the time and memory required by the forward-backward algorithm because the reconstruction algorithm will inevitably be compared to it. The forward-backward algorithm operates by repeatedly making small adjustments to a presentation. It needs $\mathcal{O}(n^2 L)$ operations per iteration and uses $\mathcal{O}(nL + n^2 m)$ storage overall. There are variants that take fewer operations, but these do not appear to be widely used [30,31]. The number of states initially selected is likely to be larger than the n estimated by the reconstruction algorithm because this number must be chosen *a priori* and because it is usually necessary to use more states than are mathematically needed in order to get an HMM which describes the data well. Additionally, the forward-backward algorithm requires an unspecified number of iterations to converge. The author knows no way to estimate how this number of iterations scales with the size of the data set or the complexity of the process.

Finally, the reconstruction algorithm, as presented here, should be thought of as a mathematical draft, rather than a finished program. At this time it has been implemented, but the implementation cannot be considered an optimal coding. There has not been a systematic search either for a good heuristic for Y or for the heuristic used to choose the threshold θ. And there has not been any systematic comparison of the results of this algorithm to those of the forward-backward algorithm.

When the existing implementation is given actual probabilities (that is, we apply QL to $\overline{P}$ instead of $\hat{P}$) for a process defined by a GHMM, it reliably produces a minimal

GHMM which is, to machine precision, equivalent to the original. When it is given a sufficiently large sample of the output from a GHMM, it typically produces a GHMM that represents a process with word probabilities similar to those of the original GHMM. It is prone to two types of failures, however. In one, it produces a GHMM with an excessive number of states because the threshold θ is too low. In the other, it produces an invalid Proto-GHMM. The reconstruction algorithm needs more work on the implementation details in order to make it widely usable, but the overall framework has a solid theoretical grounding and may, in time, replace the forward-backward algorithm.

6 Conclusions and Further Directions

In the preceding chapters, we have introduced process states, described the process states of an HMM, and shown that the span of an HMM's process states is a finite-dimensional vector space. We have introduced GHMMs, shown how to tell when pairs of GHMMs represent the same process, and shown how to construct a minimal GHMM equivalent to a given one. Finally, we have given procedures for constructing a GHMM presentation for a process either from word probabilities or from a sample of output. We will conclude this dissertation by making some observations and by suggesting some directions for further investigation. We will discuss the viewpoint which led to this work, the problem of GHMM validity, and the question of how to find an HMM that is equivalent to a given GHMM. We will end by discussing the implications of this dissertation for the broader field of modeling complex systems.

6.1 Process states and presentations

The results of this dissertation have followed from a few ideas. The first of these is that the process is more fundamental than the presentation which represents it. This idea is present in the dynamical systems literature; see [32,33,11]. The second is that a process has states which are inherent to it, and distinct from the states of any presentation. The third is a question: What are the process states for a process in terms of a presentation that defines it? And the fourth is the observation that if a process has an HMM presentation, then its process states lie in a finite-dimensional vector space. Together, these ideas lead to a viewpoint that is useful for the study of the processes generated by HMMs.

Some previous work on HMMs has defined the processes represented by HMMs, usually in discussing the problem of HMM equivalence [2]. However, these works have kept the focus on the HMMs' presentations themselves, and have worked with processes very little. In this dissertation, on the other hand, the focus is on the processes, and HMMs are of interest primarily as convenient representations of processes. We justify this shift in focus, and the accompanying shift in viewpoint, with the assertion that processes are the more fundamental objects. The process, not the presentation, is almost always the object of the real focus. (There may be a few applications in which a

phenomenon being studied has a known structure, and the presentation states may be chosen so as to have some inherent reality of their own. But in any other use of HMMs, the presentation states themselves have no meaning and the HMM is being used solely as a representation of a process.)

A similar contrast between this dissertation and previous work on HMMs can be made for process states and presentation states. In previous work on HMMs, the word *state* is used exclusively to refer to presentation states. Much of this work uses the vectors we have named *mixed states* [1,3] without referring to them as states of any sort. (The terms *mixed state*, *process state*, and *presentation state* were coined by the author for this work, so they could not have been used in previous works. Mixed states have not been named; process states have been called *causal states* [11], and presentation states have simply been called *states*.) In some works — for example, [1] — it is clear that the authors were aware that mixed states render the future conditionally independent of the past.

Beginning with the process, we are led to define the process state, because process states are inherent to the process, and presentation states are not. And later, when we introduce HMMs, it is natural to ask what their process states look like. With this background, the mixed states become objects with meaning — conditional future distributions — instead of merely being intermediate results in a computation. Thus HMMs, which we use as a convenient way to represent processes, have given us a convenient way to represent process states. From this we see that an HMM's process states lie in a finite-dimensional vector space, and we see the presentation states in their true role as basis vectors for this space.

This viewpoint can lead us in other directions as well. Process states are useful for calculating various statistics. The entropy of a process, for instance, may readily be computed from an HMM presentation by use of mixed states, but not directly from the presentation states. This was done in [1,34,35]

The author has work in progress concerning the computation of other statistics — notably statistical complexity and excess entropy [9,11] — and a classification of processes with HMM presentations. It appears that this perspective will be a useful one for any research involving HMMs.

6.2 Generalized Hidden Markov Models

Many of the results in this dissertation are stated for Generalized Hidden Markov Models instead of Hidden Markov Models. For several reasons, however, the reader may prefer to work with HMMs. Certainly HMMs are more familiar, and the reader is likely to be more comfortable with them than with GHMMs because of their negative entries. There is no question of validity with HMMs — every HMM is valid. Also, HMMs can be interpreted by examining the transition matrices, though such interpretation may be suspect unless equivalent HMMs receive similar interpretations. And finally, there are questions which make sense for HMMs but not for GHMMs, such as, What presentation state is the HMM most likely to be in at time t? (The meaningfulness of the answer is questionable, as discussed above, unless individual states have intrinsic meanings.)

We ask the reader to work with GHMMs for the following reasons. First, when the presentation states are correctly understood as the basis elements for the space containing the process states, the entries in the transition matrices are understood to be coordinates and not probabilities. With this in mind, restricting vectors to the positive cone of the space, in which all coordinates are positive, is unnatural and arbitrary. Second, the results of chapters 4 and 5 use the tools of linear algebra. Use of these tools becomes much more difficult if it is necessary to stay entirely in the positive cone. Furthermore, it is possible — though no examples are known to the author — that there are processes which may be represented with fewer states as GHMMs than as HMMs, or that processes exist that can be represented as GHMMs but not as HMMs.

Determining whether or not a Proto-GHMM is valid — given $\left(V, \mathcal{X}, \left\{T^k\right\}, \pi\right)$ such that π and $\sum_k T^k$ are unit-sum, is $\pi T^x \vec{1} \geq 0$ for all words x? — seems to be a hard problem. If we simply generate random output, negative "probabilities" of symbols usually show up quickly or not at all. But the absence of negative "probabilities" up to any finite time does not prove that the Proto-GHMM is valid.

We might try the following naive algorithm for testing validity. If we order the set of all finite words, we can compute the probability of each word in turn. When we reach a negative probability, we conclude the Proto-GHMM is invalid and halt. But if it is valid, the process never halts.

The question of validity can be phrased in the following way. Does the set of

reachable process states intersect the set of vectors v for which $vT^k\vec{1} < 0$ for any symbol k? The latter of these sets is a finite union of half-spaces, so the answer does not change if we replace the former set with its convex hull. This suggests the following approach: we take a queue of mixed states, initially containing only the stationary vector π and a convex set S, initially empty. For each vector v that we remove from the queue and for each symbol k, we compute the "probability" $vT^k\vec{1}$ and halt if it is negative. Then we generate the mixed state $u = N\left(vT^k\right)$ which is the result if the process is in the process state corresponding to v and then emits k. If u is not in S, we replace S with the convex hull of $S \cup \{u\}$ and add u to the queue. We continue in this way until the queue is empty.

In essence, this algorithm constructs a subset of the set of all words such that if none of the words in the subset has negative "probability," then the Proto-GHMM is valid. However, this algorithm has the same problem as the naive algorithm. For some GHMMs it will never stop because this subset is still infinite — the queue may never be empty. For the simple nondeterministic source we saw in section 3.5, for example, there is an infinite sequence of words all of which lie outside the convex hull of all the preceding words.

No finite method for generating the convex hull of the set of reachable process states is known, though it may be possible to develop such a method based on linear programming. The work of Heller, which gives results for functions of finite Markov Chains in terms of convex polygonal cones, may contain part of a solution [21]. Indeed, it is not known whether or not this set can always be finitely described. Thus, we do not know of a practical method for testing whether or not a Proto-GHMM is valid.

6.3 Converting GHMMs to HMMs

There is another aspect of our understanding of GHMMs which is unsatisfactory. If we have a GHMM, is there an HMM which is equivalent to it? If so, is there an equivalent HMM that has the same number of states as the GHMM? These questions are open, but in light of the present understanding, we offer the following conjectures.

Conjecture 6.3.1. Given a GHMM, there is an HMM that is equivalent to it.

Conjecture 6.3.2. Given a GHMM, there is an HMM that is equivalent to it with the same number of presentation states.

For a slightly different formulation of GHMMs — which output from states instead of from transitions — Balasubramanian [8] has asserted that conjecture 6.3.1 holds but conjecture 6.3.2 does not. A similar assertion is implied in [7], but neither paper gives any proof or any counterexamples that apply to GHMMs as defined here. Darmadhikari and Heller state results for "regular" functions of a Markov Chain that the author has not applied to GHMMs [10,21]. One or both of these results may show that conjecture 6.3.2 is false. The author of this dissertation expects that both conjectures hold, but has not found a proof.

Whether or not conjectures 6.3.1 and 6.3.2 hold, there are GHMMs for which equivalent HMMs exist. We can construct such GHMMs by conjugating HMMs, or by reconstructing from probabilities. When an equivalent HMM exists, how can we find it? Because this is a matter of converting from a generalized HMM to an ordinary HMM, we will call this the *degeneralization problem*.

There is reason to believe that the degeneralization problem is nontrivial. Suppose, for instance, we have an algorithm that will degeneralize any GHMM for which an equivalent HMM exists. If we apply this algorithm to an invalid Proto-GHMM, it must fail. If we apply it to an arbitrary Proto-GHMM, and it succeeds, we have shown that the Proto-GHMM is valid. Thus, if we have a degeneralization algorithm we have a way to establish the validity of a substantial collection of Proto-GHMMs. Furthermore, if conjecture 6.3.1 is true, the only way this degeneralization algorithm can fail is if the Proto-GHMM is invalid. In this case, our degeneralization algorithm serves as a general test which determines the validity of Proto-GHMMs.

We can give a necessary and sufficient condition for a GHMM to have a HMM equivalent to it, but this condition is so close to restating the definition that it has little value. Given a GHMM, there is an HMM equivalent to it if and only if there exists a convex set C in the space containing the mixed states with a finite number of vertices such that

1. the initial distribution π is in C, and

2. for all vertices v and for all symbols k the vector vT^k lies in the convex hull of $C \cup \left\{\vec{0}\right\}$.

If C exists, then the vertices of C are the presentation states of an HMM equivalent to the given GHMM. As with validity, the problem is finding the convex set.

No degeneralization algorithm exists at present, but one possible approach is known. If we conjugate by a carefully chosen matrix, we can make any particular entry in any given transition matrix nonnegative, with the possible cost of making some other entry negative. It may be possible to choose a matrix that makes all entries simultaneously nonnegative. The task of finding such a matrix may be approached as a maximization task: maximize the sum of the negative entries of the matrices AT^kA^{-1} over the space of $n \times n$ stochastic invertible matrices A.

General multidimensional maximization techniques fail to find suitable matrices even when they are known to exist. This probably occurs because these techniques search for local maxima and the space of invertible matrices is disconnected. However, it may be possible to develop a global technique specialized to the form of this specific problem.

Further, and of particular interest if conjecture 6.3.1 holds but conjecture 6.3.2 fails, it is possible to add states to a GHMM by an operation similar to conjugation. If v is a stochastic row vector of length n, and h is an arbitrary column vector of length n, consider the $n \times (n+1)$ matrix

$$A = (I - hv \quad h) \tag{6.1}$$

and the $(n+1) \times n$ matrix

$$B = \begin{pmatrix} I \\ v \end{pmatrix}. \tag{6.2}$$

Note that both A and B are stochastic and their product is $AB = I$. If $\left(V, \mathcal{X}, \left\{T^k\right\}, \pi\right)$ is a GHMM and we let $\tau = \pi A$, and for all k we let $U^k = BT^kA$, then for any suitable V', $\left(V', \mathcal{X}, \left\{U^k\right\}, \tau\right)$ is an $n+1$–state GHMM which is equivalent to $\left(V, \mathcal{X}, \left\{T^k\right\}, \pi\right)$. Thus, if we cannot find an n–state HMM equivalent to our original GHMM, we can search for one with more than n states. A solution to the degeneralization problem may well emerge from these techniques.

6.4 Reconstruction

The reconstruction algorithm is the crowning achievement of this dissertation. The algorithm is novel and it is not a variation on an older algorithm. It operates directly, without requiring an initial configuration and without making iterative adjustments to the model. The connection between the source data and the resulting Proto-GHMM through the probability estimates is natural and clear.

The practical implementation of the reconstruction algorithm (that builds GHMMs from samples) shows considerable promise, but needs further development in order to be widely useful. As discussed at the end of section 5.2, this work includes looking for better heuristics, considering other possible estimators, a proper statistical study, and general fine-tuning.

6.5 Last remarks

The problem of HMM (or GHMM) reconstruction may be thought of as a "complex estimation problem" or a problem of "model inversion with uncertainty." That is, we want to take a random sample and build a model from it. But what we have is a class of models and a method of generating random samples from a model in this class. However, the method of generating samples, while not complicated, is involved enough that there is not a practical way to "invert" it. There are many model classes to which this description applies, including a number which have applications: Hidden Markov Models, neural networks, and a number of less well known model classes used in pattern recognition.

In recent years, there has been a proliferation of work attempting to solve these problems by iterated improvement. Forward-backward and back propagation, for HMMs and neural nets, respectively, are probably the oldest of these. Many of these approaches use versions of the expectation-maximization (EM) algorithm, which is a general algorithm that may be specialized to address many situations. Others use stochastic optimization algorithms, such as simulated annealing and genetic algorithms. Most of these approaches have similar failings: they get stuck in local optima, they may depend strongly on the initial (random) model, and they often require larger models than is appropriate to get a decent answer. Nonetheless, these approaches are being used because these algorithms

provide a way to get some sort of answer to questions that previously would have been intractable. It is not inappropriate to describe these methods as crude tools by which one can bring a computer to bear on modelling problem.

This dissertation has taken a different approach. We began by attempting to better understand HMMs on a theoretical level. This led to an attempt to characterize the class of processes which could be represented by HMMs. With the better theoretical understanding we had gained, a new approach to the reconstruction problem became accessible. Although it remains to be seen how the reconstruction algorithm will serve in practice, there is reason to believe that in time it may replace forward-backward as the main method by which HMMs are constructed from the sample data.

It is the author's contention that this experience is applicable to other problems of model inversion with uncertainty. It is undoubtedly easier to implement an iterative improvement algorithm than to do this sort of theoretical study, so iterative improvement schemes may remain useful in the study of new model classes. After a model class has proved its utility, however, it is valuable to gain the theoretical understanding necessary to develop more direct algorithms.