Mixed memory Markov models

Lawrence K. Saul
lsaul@research.att.com

Speech and Image Processing Services
AT&T Labs - Research
600 Mountain Avenue, 2D-349
Murray Hill, NJ 07974

Michael I. Jordan
jordan@psyche.mit.edu

Center for Biological and Computational Learning
Massachusetts Institute of Technology
79 Amherst Street, E10-034D
Cambridge, MA 02139

Abstract

We consider how to parameterize Markov models with prohibitively large state spaces. This is done by representing the transition matrix as a convex combination—or mixture—of simpler dynamical models. The parameters in these models admit a simple probabilistic interpretation and can be fitted iteratively by an Expectation-Maximization (EM) procedure. We give examples where these models may be a faithful and/or useful representation of the underlying dynamics. We also derive a set of generalized Baum-Welch updates for hidden Markov models (HMMs) that make use of this parameterization. Because these models decompose the hidden state as the Cartesian product of two or more random variables, they are well suited to the modeling of coupled time series.

1 Introduction

The modeling of time series is a fundamental problem in machine learning, with widespread applications. These include speech recognition[17], natural language processing[14], traffic surveillance[11], protein modeling[13], musical analysis/synthesis[10], and numerous others.

Probabilistic models of time series typically start from some form of Markov assumption—namely, that the future is independent of the past given the present. For the purpose of statistical estimation, a problem arises if either: (i) the system possesses a large number of degrees of freedom, or (ii) the window of "present" knowledge necessary to predict the future extends over several time steps. In these cases, the number of parameters to specify the Markov model can overwhelm the amount of available data. In particular, for a system with n possible states and memory length k, the number of free parameters scales exponentially as n^{k+1}.

The situation is even worse for latent variable models in which the Markov assumption applies to the hidden state space. In this case, the parameter estimation may not only require exponential amounts of data to avoid overfitting, but also exponential amounts of computation to fill in missing values for the hidden states. Most applications of hidden Markov models (HMMs)[2, 17] have therefore been limited to models with first-order dynamics and small numbers of hidden states. In general, inference and learning are not computationally tractable for models with (combinatorially) large numbers of hidden states. In practice, such models must be trained by some deterministic or sampling-based approximation to the Baum-Welch algorithm.

In this paper we propose a principled way to parameterize Markov models with large state spaces. This is done by representing the transition matrix as a convex combination—or mixture—of simpler dynamical models. We refer to the resulting models as *mixed memory Markov models.* We also show how this idea generalizes to the transition and emission matrices of HMMs.

Our methodology is especially geared to *factorial* models—that is, models in which large state spaces are represented via the Cartesian product of smaller ones. Applied to time series, these models are known as dynamic probabilistic networks[19] or factorial HMMs[12]. Graphically, they are represented as sets of Markov chains that are connected (via directed links) to a common set of observable nodes. These models arise naturally in the study of coupled time series, where the observations have an *a priori* decomposition as the Cartesian product of two or more random variables. Factorial HMMs aim to combine the power of distributed representations with the richness of probabilistic semantics. For this reason, they have been studied by researchers interested in all aspects of artificial intelligence[8, 12, 19, 21].

We believe that any useful parameterization of factorial models should satisfy two basic criteria. First, it should give rise to efficient algorithms for statistical estimation. Second, it should provide a compact—but flexible—representation of the underlying probabilistic dependencies.

Mixed memory models satisfy both these criteria. Parameter estimation in these models is handled by the well-known Expectation-Maximization (EM) procedure[9]. The tractability of EM is a special property of mixed memory models; other parameterizations (e.g., sigmoid[15], noisy-OR[16]) must typically rely on some form of gradient descent[1, 19]. In practice, EM algorithms have a number of advantages over gradient-based methods, including monotone convergence in likelihood, lack of step size parameters, and naturalness at handling probabilistic constraints.

In terms of representational power, mixed memory models can express a rich set of probabilistic dependencies. Though compact and easy to interpret, they do not make strong assumptions of conditional independence, nor are they restricted (in the modeling of equal-time correlations) to Gaussian random variables. As such, they represent a significant extension of previous work[12] on EM algorithms for factorial HMMs.

The main significance of this work lies in its application to factorial HMMs and the modeling of coupled time series. In principle, though, our methodology can be applied whenever large state spaces arise as the Cartesian product of two or more random variables. We will take advantage of this generality to present mixed memory models in a number of different settings. In doing this, our goal is to build up—in a gradual way—the somewhat involved notation needed to describe factorial HMMs.

The organization of this paper is therefore as follows. In order of increasing complexity, we consider: (1) higher-order Markov models, where large state spaces arise as the Cartesian product of several time slices; (2) factorial Markov models, where the dynamics are first order but the observations have a componential structure; and (3) factorial HMMs, where the Markov dynamics apply to hidden states, as opposed to the observations themselves. For factorial HMMs, we also describe an efficient approximation for computing the statistics of the hidden states. We conclude that mixed memory models provide a valuable tool for understanding complex dynamical systems.

2 Higher order Markov models

Let $i_t \in \{1, 2, \ldots, n\}$ denote a discrete random variable that can take on n possible values. A kth order Markov model is specified by the transition matrix $P(i_t|i_{t-1}, i_{t-2}, \ldots, i_{t-k})$. To avoid having to specify the $O(n^{k+1})$ elements[1] of this matrix, we consider parameterizing the model by the convex combination:

$$P(i_t|i_{t-1}, i_{t-2}, \ldots, i_{t-k}) = \sum_{\mu=1}^{k} \psi(\mu)\ a^{\mu}(i_t|i_{t-\mu}), \quad (1)$$

where $\psi(\mu) \geq 0$, $\sum_\mu \psi(\mu) = 1$, and $a^\mu(i'|i)$ are k elementary $n \times n$ transition matrices. The model in eq. (1) is therefore specified by $O(kn^2)$ parameters, as opposed to $O(n^{k+1})$ for the full memory model. Note how $\psi(\mu)$ is used to weight the influence of past observations on the distribution over i_t. This type of weighted sum is the defining characteristic of mixed memory models.

For the purpose of parameter estimation, it is convenient to interpret the index μ in eq. (1) as the value of a latent variable. We denote this latent variable (at each time step) by x_t and consider the joint probability distribution:

$$P(i_t, x_t = \mu|i_{t-1}, \ldots, i_{t-k}) = \psi(\mu)\ a^\mu(i_t|i_{t-\mu}). \quad (2)$$

Note that marginalizing out x_t (i.e., summing over μ) recovers the previous model for the transition matrix, eq. (1). Thus we have expressed the dynamics as a mixture model, in which the parameters $\psi(\mu)$ are the marginal probabilities, $P(x_t = \mu)$. Likewise, we can view the parameters $a^\mu(i'|i)$ as the conditional probabilities, $P(i_t = i'|i_{t-1}, i_{t-2}, \ldots, i_{t-k}, x_t = \mu)$.

Let $I = \{i_1, i_2, \ldots, i_L\}$ denote an observed time series of length L. The sufficient statistics for a full memory Markov model are the transition frequencies. To fit the mixed memory Markov model we avail ourselves of the EM procedure [9]. In general terms the EM algorithm calculates *expected* sufficient statistics and sets them equal to the *observed* sufficent statistics. The procedure iterates and is guaranteed to increase the likelihood at each step.

For the model in eq. (2), the EM updates are:

$$\psi(\mu) \leftarrow \frac{\sum_t P(x_t = \mu|I)}{\sum_{t\nu} P(x_t = \nu|I)}, \quad (3)$$

$$a^\mu(i'|i) \leftarrow \frac{\sum_t P(x_t = \mu|I)\ \delta(i_{t-\mu}, i)\ \delta(i_t, i')}{\sum_t P(x_t = \mu|I)\ \delta(i_{t-\mu}, i)}, \quad (4)$$

where we have used $\delta(i, i')$ to denote the Kronecker delta function

$$\delta(i, i') = \begin{cases} 1 & \text{if } i = i' \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

We have written out the denominator in eq. (3) to make the normalization of $\psi(\mu)$ explicit; for a sequence of length L, this term of course simplifies to $L-1$. The EM updates for this model are easy to understand; at each

[1] See [18] for an alternative approach—the *variable memory Markov model.*

English	Italian	Finnish
aback	abaca	runo
abacus	abache	mieleni
abalone	abachi	minun
abandon	abacista	tekevi
abase	abaciste	aivoni
abash	abacisti	ajattelevi
abate	abaco	laulamahan
abater	abaliena	saaani
abbas	abalienando	sanelemahan
abbe	abalienano	suoltamahan
abbey	abalienare	laulamahan
abbot	abalienarono	sanat

Table 1: Some of the words used to fit Markov models of English, Italian, and Finnish spelling.

iteration, the model parameters are adjusted so that the statistics of the joint distribution match the statistics of the posterior distribution. The expectations in eqs. (3) and (4) may be straightforwardly computed from Bayes rule:

$$P(x_t = \mu | I) = \frac{\psi(\mu)\ a^{\mu}(i_t | i_{t-\mu})}{\sum_{\nu} \psi(\nu)\ a^{\nu}(i_t | i_{t-\nu})}. \qquad (6)$$

Note that this algorithm requires no fine-tuning of step sizes, as does gradient descent.

In terms of representational power, the model of eq. (1) lies somewhere in between a first order Markov model and a kth order Markov model. To demonstrate this point, we fitted various Markov models to word spellings in English, Italian, and Finnish. The state space for these models was the alphabet (e.g., 'a' to 'z' for English), and the training data came from very long lists of words with four or more letters (see table 1). The lists contained tens of thousands of words, making them sufficiently long to obtain accurate estimates of $P(i_t | i_{t-1}, i_{t-2})$.

In table 2, we give the results measured in entropy per character for a number of simple models. The results show that the mixed memory model does noticeably better than the first-order Markov model. Of course, it cannot capture all the structure of the full second-order model, which has more than ten times as many parameters. The mixture model should accordingly be viewed as an intermediate step between first and higher-order Markov models.

We envision two situations in which the model of eq. (1) may be gainfully applied. The first is when the dynamics of the process generating the data are faithfully described by a mixture model. In this case, one would expect the mixture model to perform as well as the (full) higher-order model while requiring substantially less data for its parameter estimation. The second situation in which this model may be appropriate is when the amount of training data is extremely sparse relative to the size of the state space. In this case, the parameterization in eq. (1), though a poor approximation to the true model, may be desirable to avoid overfitting.

order	memory	English	Italian	Finnish
0th	none	0.900	0.844	0.840
1st	full	0.776	0.696	0.707
2nd	mixed	0.754	0.678	0.679
2nd	full	0.689	0.622	0.607

Table 2: Entropy per character, computed from Markov models of English, Italian, and Finnish spelling.

A real-world example of the first situation might be the modeling of web sites visited during a session on the World Wide Web. The modeling of these sequences has applications to web page prefetching and resource management[3, 7] on the Internet. Typically, the choice of the next web site is conditioned on a previous site, but not necessarily the last one that has been visited. (Recall how often it is necessary to retrace one's steps, using the BACK option.) Mixed memory models capture this type of conditioning explicitly. It is also worth noting that the size of the state space (i.e., number of web sites) is extremely large and growing at a rapid pace. Compactly parameterized models are therefore appropriate in this domain.

A real-world example of the second situation is language modeling. The ability to discern likely sequences of words from unlikely sequences is an important component of automated speech recognition. For large vocabularies—in the tens of thousands of words—there is never sufficient data to estimate (robustly) the statistics of second or higher order Markov models. In practice, therefore, these models are "smoothed" or interpolated[6] with lower order models. The interpolation with lower order models is forced on practitioners by the enormous size of the state space (e.g., 10^4 words) and the small (in relative terms) amount of training data (e.g., 10^8 words).

We have shown that the model in eq. (1) is intermediate between first and higher order Markov models. In particular, it is intermediate both in terms of representational power and the amount of data required for parameter estimation. Models of this form are therefore a good candidate for language modeling, where the goal of model accuracy must be carefully balanced against the problem of overfitting. One of us (LS) is currently experimenting with these models, applied to a sixty-thousand word vocabulary and a eighty-million word corpus.

3 Factorial Markov models

In the last section, we saw how large state spaces arose as the result of higher order dynamics. In this section, we consider another source of large state spaces—namely, factorial representations. Many time series have a natural componential structure. Consider for example the four voices—soprano (S), alto (A), tenor (T), and bass (B)—of a Bach fugue[10]. We can model each voice by a separate Markov model, but this will not capture the correlations due to harmony. The most straightforward way to model the coupling between voices is to write down a Markov model whose dynamical state is the Cartesian product of the four voices. But the combinatorial structure of this state space leads to an explosion in the number of free parameters; thus it is imperative to provide a compact representation of the transition matrix.

Mixed memory models are especially geared to these sorts of situations. Let I_t denote the tth element of a vector time series, and i_t^μ the μth component of I_t. If each vector has k components, and each component can take on n values, then the overall state space has size n^k. To model the coupling between these components in a compact way, we represent the transition matrix by the convex combination:

$$P(I_t|I_{t-1}) = \prod_\nu \sum_\mu \psi^\nu(\mu)\; a^{\nu\mu}(i_t^\nu|i_{t-1}^\mu). \qquad (7)$$

Here again, the parameters $a^{\nu\mu}(i'|i)$ are k^2 elementary $n \times n$ transition matrices, and the parameters $\psi^\nu(\mu)$ are positive numbers that satisfy $\sum_\mu \psi^\nu(\mu) = 1$. The number of free parameters in eq. (7) is therefore $O(k^2n^2)$, as opposed to $O(n^{2k})$ for the full memory model.

The parameters $\psi^\nu(\mu)$ measure the amount of correlation between the different components of the time series. In particular, if there is no correlation, then $\psi^\nu(\mu) = \delta(\mu,\nu)$, and the νth component at time t is independent of the other components at time $t-1$. On the other hand, if $\psi^\nu(\nu) < 1$, then from eq. (7) we see that the other components at time $t-1$ influence the probability distribution over i_t^ν. The matrices $a^{\nu\mu}(i'|i)$ provide a compact way to parameterize these influences.

As in the previous section, it is convenient to introduce latent variables x_t^ν and view eq. (7) as a mixture model. Thus we may write:

$$\begin{aligned} P(i_t^\nu, x_t^\nu = \mu|I_{t-1}) &= \psi^\nu(\mu)\; a^{\nu\mu}(i_t^\nu|i_{t-1}^\mu), \qquad (8)\\ P(I_t, X_t|I_{t-1}) &= \prod_\nu P(i_t^\nu, x_t^\nu|I_{t-1}). \qquad (9)\end{aligned}$$

Here, the role of x_t^ν is to select which component of I_{t-1} determines the transition matrix for i_t^ν. As before, we can derive an EM algorithm to fit the parameters of this

soprano	alto	tenor	bass
61	54	49	46
61	54	49	44
61	54	49	44
66	54	49	46
66	54	49	46
66	54	49	46
66	54	49	46
66	54	51	46
66	54	51	46
66	54	52	46
66	54	52	46
66	56	51	48
66	56	51	48
66	56	51	48
66	56	51	48
66	56	51	48
66	56	51	48
66	56	51	48
66	56	51	48
64	56	49	49
64	56	49	49
64	56	49	49
64	56	49	49
64	52	49	49
64	52	49	49

Table 3: Portion of the four-component time series generated by Bach's last fugue.

model. In this case, the EM updates are:

$$\begin{aligned} \psi^\nu(\mu) &\leftarrow \frac{\sum_t P(x_t^\nu = \mu|I)}{\sum_{t\mu'} P(x_t^\nu = \mu'|I)} \qquad (10)\\ a^{\nu\mu}(i'|i) &\leftarrow \frac{\sum_t P(x_t^\nu = \mu|I)\; \delta(i, i_{t-1}^\mu)\; \delta(i', i_t^\nu)}{\sum_t P(x_t^\nu = \mu|I)\; \delta(i, i_{t-1}^\mu)} \qquad (11)\end{aligned}$$

where I stands for the observed time series. Naturally, the structure of these updates is quite similar to the model of the previous section.

To test this algorithm, we learned a model of the four-component time series generated by Bach's last fugue. Table 3 shows a piece of this time series; here, each element represents a sixteenth note, while the number codes the pitch. This fugue has a rich history, the details of which are given in [10]. The time series (3284 beats long) was made public following the Santa Fe competition on time series prediction.

By examining the parameters of the fitted model, we can see to what extent each voice enables one to make predictions about the others. In general, we observed that the mixture coefficients $\psi^\nu(\mu)$ were very close to zero or one; for example, $\psi^S(S,A,T,B) = (0.96, 0.02, 0.01, 0.01)$. The reason for this is that the

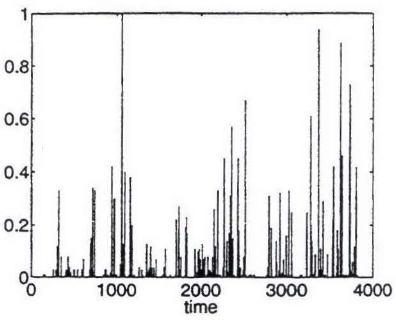

Figure 1: Plot of soprano-tenor correlations versus times, obtained from a mixed memory Markov model of Bach's last fugue.

voices do not typically change pitch with every sixteenth note. Hence, for each voice the note at the current beat is a very good predictor of the note at the next one.

When the voices do make a transition (i.e., move up or down in pitch), however, the coupling between voices becomes evident. To see this, we can look at the *posterior* probabilities of the latent variables, x_t^μ, which reveal the extent to which the voices interact at specific moments in time. Figure 1, shows a plot of the posterior probabilities $P(x_t^S = T|I)$ versus time, calculated from the fitted model. Within the framework of the mixture model, these probabilities measure the relative degree to which the soprano's note at time t can be predicted from the tenor's note at time $t-1$. The moments at which this probability acquires a non-zero value indicate times when the tenor and soprano are tightly coupled. Not surprisingly, these pulses of coupling (when viewed as a time series) have a discernable rhythm and regularity of their own.

4 Factorial HMMs

Building on the results of the last section, we now consider the generalization to factorial hidden Markov models (HMMs). These are HMMs whose states and observations have an internal, combinatorial structure. How might such structure arise? Suppose we are trying to model the processes that give rise to a speech signal. A number of unobserved variables interact to generate the signal that we ultimately observe. In an articulatory model of speech production, these variables might encode the positions of various organs, such as the lip, tongue, and jaw. In a recognizer, these variables might encode the current phonemic context, the speaker accent and gender, and the presence of background noise. In either case, the hidden state for these models is naturally decomposed as the Cartesian product of several random variables.

Another motivation for factorial representations is that in many applications, the observations have an *a priori* componential structure. This is the case, for example, in audiovisual speech recognition[5], where information from different modalities is being combined and presented to the recognizer. It is also the case in frequency subband-based speech recognition[4], where different recognizers are trained on sub-bands of the speech signal and then combined to make a global decision. Simple ways to integrate these different components are: (a) collapsing the data into a single time series or (b) reweighting and combining the likelihood scores of independent HMMs. One might hope for a more sophisticated integration, however, by building a joint model that looks for correlations on the actual time scale of the observations.

Whatever the manner in which they arise, factorial HMMs pose two concrete problems. The first is representation. In most applications, there is not sufficient data to estimate the elements of the full transition and emission matrices formed by taking the Cartesian product of the individual factors. How should one parameterize these matrices without making restrictive or inelegant assumptions? Ideally, the representation should not make unjustified assumptions of conditional independence, nor should it force us to give up desirable properties of the EM algorithm, such as monotone convergence in log-likelihood.

The second problem in factorial HMMs is one of computational complexity. The Baum-Welch algorithm for parameter estimation scales as $O(N^2)$, where N is the number of hidden states. If the hidden state is a Cartesian product of k random variables, each of degree n, then the effective number of hidden states is $N = n^k$. Even for small k, this may be prohibitively large to calculate the statistics in the E-step of the EM algorithm. Hence, one is naturally led to consider approximations to these statistics. Though Markov chain Monte Carlo methods have been proposed to estimate these statistics, we generally consider such methods to be too slow for the inner loop of a learning algorithm.

Let us now return to our development of factorial HMMs with these issues in mind. We will see that mixture models provide a good compromise to the problem of representation, and that efficient deterministic approximations are available for the problem of parameter estimation.

For concreteness, let us suppose that we have trained k simple HMMs on separate time series of length L:

$$J^1 = \{J_1^1, J_2^1, \ldots, J_L^1\} \tag{12}$$
$$J^2 = \{J_1^2, J_2^2, \ldots, J_L^2\} \tag{13}$$
$$\vdots$$
$$J^k = \{J_1^k, J_2^k, \ldots, J_L^k\} \tag{14}$$

Now we wish to combine these HMMs into a single model in order to capture (what may be) useful correlations between the different time series. For simplicity, we assume that each individual HMM had n hidden states and m types of observations. Thus the hidden state space of the combined model, represented at each time step by the Cartesian product

$$I_t = i_t^1 \otimes i_t^2 \otimes \cdots \otimes i_t^k, \tag{15}$$

has size n^k. Likewise, the number of possible observations for the combined model is m^k.

Our first task is to extend the notation of the previous section to factorial HMMs. In an HMM, it is the hidden states (as opposed to the observations) that have a Markov dynamics. Accordingly, in this setting, we use I_t to denote the vector of hidden states at time t, i_t^ν to denote the νth component of this vector, and eq. (7) to model the associated transition matrix. Similarly, we use J_t to denote the observed vector of time series elements at time t and j_t^μ to denote the components of this vector. By analogy to eq. (7), we parameterize the emission probabilities by:

$$P(J_t|I_t) = \prod_\nu \sum_\mu \phi^\nu(\mu) b^{\nu\mu}(j_t^\nu | i_t^\mu), \tag{16}$$

where $b^{\nu\mu}(j|i)$ are k^2 elementary $n \times m$ transition matrices. Note that this model can capture correlations between the hidden states of the μth Markov chain and the observations in the νth time series.

For the purposes of parameter estimation, it is again convenient to introduce latent variables that encode the mixture components in eq. (16). By analogy to eqs. (8) and (9), we have:

$$\begin{aligned} P(j_t^\nu, y_t^\nu = \mu | I_t) &= \phi^\nu(\mu) b^{\mu\nu}(i_t^\mu, j_t^\nu), & (17) \\ P(J_t, Y_t | I_t) &= \prod_\nu P(j_t^\nu, y_t^\nu | I_t). & (18) \end{aligned}$$

Having encoded the mixture components as hidden variables, we can now apply an EM algorithm to estimate the model parameters. In this case, the updates have the form:

$$\begin{aligned} \psi^\nu(\mu) &\leftarrow \frac{\sum_t P(x_t^\nu = \mu | J)}{\sum_{t\mu'} P(x_t^\nu = \mu' | J)}, & (19) \\ a^{\nu\mu}(i'|i) &\leftarrow \frac{\sum_t P(x_t^\nu = \mu | J)\ \delta(i, i_{t-1}^\mu)\ \delta(i', i_t^\nu)}{\sum_t P(x_t^\nu = \mu | J)\ \delta(i, i_{t-1}^\mu)} & (20) \\ \phi^\nu(\mu) &\leftarrow \frac{\sum_t P(y_t^\nu = \mu | J)}{\sum_{t\mu'} P(y_t^\nu = \mu' | J)}, & (21) \\ b^{\nu\mu}(j|i) &\leftarrow \frac{\sum_t P(y_t^\nu = \mu | J)\ \delta(i, i_t^\mu)\ \delta(j, j_t^\nu)}{\sum_t P(y_t^\nu = \mu | J)\ \delta(i, i_t^\mu)} & (22) \end{aligned}$$

where J denotes the observed time series. A Viterbi approximation is obtained by considering only the most probable sequence of hidden states I^*, rather than summing over all possible sequences.

Note that computing the posterior probabilities in these updates requires $O(Ln^{2k})$ operations; the same is true for computing the Viterbi path. To reduce this computational burden, we have used an approximation for estimating the statistics in factorial HMMs, first outlined in [20]. The basic idea behind our approach is simple: the structure of the factorial HMM, though intractable as a whole, gives rise to efficient approximations that exploit the tractability of its underlying components. In this paper we discuss how these approximations can be used to estimate the Viterbi path. In general, the ideas may be extended to approximate the full statistics of the posterior distribution $P(I, X, Y|J)$.

The Viterbi path is the most probable sequence of hidden states given the data, or:

$$I^* = \arg\max_I \prod_t P(I_t|I_{t-1}) P(J_t|I_t), \tag{23}$$

where the observations J_t are held fixed on the right hand side. Rather than compute this path exactly in $O(Ln^{2k})$ steps, we consider an iterative procedure that returns a (possibly sub-optimal) path in $O(Lk^3n^2)$ steps.

Our iteration is based on a subroutine that finds the optimal path of hidden states through the μth chain *given fixed values for the hidden states of the others.* Note that when we instantiate the hidden variables in all but one of the chains, the effective size of the hidden state space collapses from n^k to n, and we can perform the optimization with respect to the remaining hidden states in $O(Ln^2)$ steps. A factor of k^2 is picked up when converting the right hand side of eq. (23) into a form for which the standard Viterbi algorithm can be applied; thus this elementary *chainwise Viterbi* operation requires $O(Lk^2n^2)$ steps.

The algorithm for approximately computing the full Viterbi path of the factorial HMM is obtained by piecing these subroutines together in the obvious way. First, an initial guess is made for the Viterbi path of each component HMM. (Typically, this is done by ignoring the intercomponent correlations and computing a separate Viterbi path for each chain.) Then, the chainwise Viterbi algorithm is applied, in turn, to each of the component HMMs. After the Viterbi algorithm has been applied k times, or once to each chain, the cycle repeats; each iteration of this process therefore involves $O(Lk^3n^2)$ steps.

Note that each iteration results in a sequence of hidden states that is more probable than the preceding one; hence, this process is guaranteed to converge to a final (though possibly suboptimal) path. In practice, we have

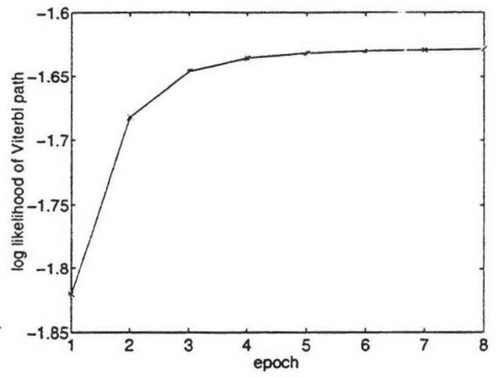

Figure 2: Plot of the log-likelihood of the Viterbi path versus the number of training iterations.

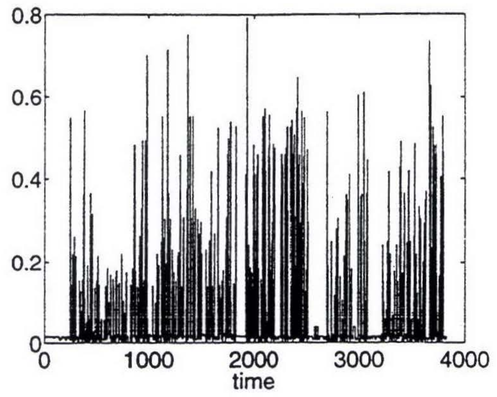

Figure 3: Plot of soprano-tenor correlations versus time, obtained from a mixed memory HMM of Bach's last fugue.

found that this process typically converges to a stable path in three or four iterations.

The chainwise Viterbi algorithm is not guaranteed to find the truly optimal sequence of hidden states for the factorial HMM. The success of the algorithm depends on the quality of the initial guess and, as always, the good judgement of the modeler. The approximation is premised on the assumption that the model describes a set of weakly coupled time series—in particular, that the auto-correlations within each time series are stronger than the cross-correlations between them. We view the approximation as a computationally cheap way of integrating HMMs that have been trained on parallel data streams. Its main virtue is that it exploits the modeler's prior knowledge that these separate HMMs should be weakly coupled. When this assumption holds, the approximation is quite accurate.

To test these ideas, we fitted a mixed memory HMM to the Bach fugue from section 3. Each voice had a component HMM with six hidden states; thus, in our previous notation, $n = 6$ and $k = 4$. We employed a Viterbi approximation to the full EM algorithm, meaning that the posterior probabilities in eqs. (19)–(22) were conditioned not only on the observations J, but also on the Viterbi path, I^*. The most probable sequence of hidden states I^* was estimated by the iterative procedure described above. Figure 2 shows (for a typical run) how the log-likelihood of this path increased with each iteration of the EM procedure. Again it was interesting to see how this model discovered correlations between the different voices of the fugue. Figure 3 shows a plot of the posterior probabilities $P(x_t^S = T|J)$ versus time, calculated from the factorial HMM (after training). The frequent pulses of non-zero probability indicate (within the framework of this model) moments of strong coupling between the soprano and tenor themes of the fugue.

5 Discussion

Many parameterizations have been proposed for probabilistic models of time series. The mixed memory models in this paper have three distinguishing features. First, they can express a rich set of probabilistic dependencies, including coupled dynamics in factorial models. Second, they can be fitted by EM algorithms, thus avoiding the drawbacks of gradient descent. Third, they are compact and easy to interpret; notably, as in ordinary Markov models, every parameter defines a simple conditional probability. All these features should enable researchers to build more sophisticated models of dynamical systems.

Acknowledgements

We thank Marney Smyth for retrieving the word lists, Tommi Jaakkola for helping us with Finnish, and Fernando Pereira for pointing out the application to web page prefetching. We also acknowledge useful discussions with Zoubin Ghahramani and Yoram Singer. This work was initiated while LS was a member of the Center for Biological and Computational Learning at MIT. During that time, it was supported by NSF grant CDA-9404932 and ONR grant N00014-94-1-0777.

References

[1] P. Baldi and Y. Chauvin. Hybrid modeling, HMM/NN architectures, and protein applications. *Neural Computation* 8:1541–1565 (1996).

[2] L. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process. In O. Shisha (ed.), *Inequalities* 3:1–8. Academic Press, New York, NY (1972).

[3] A. Bestavros and C. Cunha. A prefetching protocol using client speculation for the WWW. *Boston University Department of Computer Science Technical Report TR-95-011* (1995).

[4] H. Bourlard and S. Dupont. A new ASR approach based on independent processing and recombination of partial frequency bands. In H. Bunnell and W. Idsardi (eds.), *Proceedings of the 4th International Conference on Speech and Language Processing*, 1:426–429 (1996).

[5] C. Bregler and S. Omohundro. Nonlinear manifold learning for visual speech recognition. In E. Grimson (ed.), *Proceedings of the 5th International Conference on Computer Vision*, 494–499. IEEE Computer Society Press, Los Alamitos, CA (1995).

[6] S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics* 310–318 (1996).

[7] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of WWW client-based traces. *Boston University Department of Computer Science Technical Report TR-95-010* (1995).

[8] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3): 142–150 (1989).

[9] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* B39, 1–38 (1977).

[10] M. Dirst and A. Weigend. Baroque forecasting: on completing J. S. Bach's last fugue. In A. Weigend and N. Gershenfeld (eds.), *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, Reading, MA (1993).

[11] J. Forbes, T. Huang, K. Kanazawa, and S. Russell. The BATmobile: towards a Bayesian automated taxi. *Proceedings of the 14th International Joint Conference on Artificial Intelligence.* Montreal, Canada (1995).

[12] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. In D. Touretzky, M. Mozer, and M. Hasselmo (eds.), *Advances in Neural Information Processing Systems* 8:472–478 (1996).

[13] D. Haussler, A. Krogh, I. Mian, and K. Sjolander. Protein modeling using hidden Markov models: analysis of globins. *Proceedings of the Hawaii International Conference on System Sciences* 1:792–802. IEEE Computer Society Press, Los Alamitos, CA (1993).

[14] A. Nadas. Estimation of probabilities in the language model of the IBM speech recognition system. *IEEE Transactions on ASSP* 32(4): 859–861 (1984).

[15] R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113 (1992).

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kauffman, San Mateo, CA (1988).

[17] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286 (1989).

[18] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: learning probabilistic automata with variable memory length. To appear in *Machine Learning.*

[19] S. Russell, J. Binder, D. Koller, and K. Kanazawa. Local learning in probabilistic networks with hidden variables. *Proceedings of the 14th International Joint Conference on Artificial Intelligence.* Montreal, Canada (1995).

[20] L. Saul and M. Jordan. Exploiting tractable substructures in intractable networks. In D. Touretzky, M. Mozer, and M. Hasselmo (eds.), *Advances in Neural Information Processing Systems* 8:486–492 (1996).

[21] C. Williams and G. Hinton. Mean field networks that learn to discriminate temporally distorted strings. *Proceedings of the Connectionist Models Summer School*, 18–22 (1990).