Machine Learning, ??, 1–11 (1998)

Mixed memory Markov models: decomposing complex stochastic processes as mixtures of simpler ones

LAWRENCE K. SAUL lsaul@research.att.com
AT&T Labs, Florham Park, NJ 07932

MICHAEL I. JORDAN jordan@cs.berkeley.edu
University of California, Berkeley, CA 94720

Editor: Padhraic Smyth

Abstract. We study Markov models whose state spaces arise from the Cartesian product of two or more discrete random variables. We show how to parameterize the transition matrices of these models 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 derive a set of generalized Baum-Welch updates for factorial hidden Markov models that make use of this parameterization. We also describe a simple iterative procedure for approximately computing the statistics of the hidden states. Throughout, we give examples where mixed memory models provide a useful representation of complex stochastic processes.

Keywords: Markov models, mixture models, discrete time series

1. Introduction

The modeling of time series is a fundamental problem in machine learning, with widespread applications. These include speech recognition (Rabiner, 1989), natural language processing (Nadas, 1984), traffic surveillance (Forbes et al., 1995), protein modeling (Haussler et al., 1993), musical analysis/synthesis (Dirst & Weigend, 1993), visual gesture recognition (Yamato et al., 1992), and numerous others.

Probabilistic models of discrete 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, problems arise if either: (i) the system possesses a large number of degrees of freedom, or (ii) the window of present knowledge required 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 difficulties are compounded for latent variable models in which the Markov assumption applies to the hidden state space. In this case, it may be computationally intractable to infer values for the hidden states. For example, in first-order hidden Markov models (HMMs), computing the likelihood of a sequence of observations scales as n^2, where n is the number of hidden states (Rabiner, 1989). In practice, exact probabilistic inference is therefore limited to HMMs with relatively small (or tightly constrained) state spaces.

In this technical note, we propose a principled way to investigate 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. While the use of mixture distributions to parameterize higher-order Markov models is well known (Raftery, 1985; Ney et al, 1994; MacDonald & Zucchini, 1997), here we apply this methodology more generally to *factorial* models—models in which large state spaces are represented via the Cartesian product of smaller ones.

Our note builds on earlier work describing factorial HMMs (Ghahramani & Jordan, 1997) and dynamic probabilistic networks (Binder et al., 1997). These papers show that complex stochastic processes can be graphically represented by sets of Markov chains connected (via directed links) to a common set of observable nodes. Such 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 latent, distributed representations with the richness of probabilistic semantics (Williams & Hinton, 1990). Capturing this type of probabilistic reasoning is a fundamental problem in artificial intelligence (Dean & Kanazawa, 1989).

We believe that mixed memory Markov models have several advantages for representing complex stochastic processes and learning from examples. The parameters in these models admit a simple probabilistic interpretation and can be fitted iteratively by an Expectation-Maximization (EM) procedure (Dempster et al., 1977). The EM algorithm has several desirable properties, including monotone convergence in log-likelihood, lack of step size parameters, and naturalness at handling probabilistic constraints. In many situations, it provides a compelling alternative to gradient-based learning methods (Baldi & Chauvin, 1996; Binder et al., 1997).

Mixed memory models can also express a rich set of probabilistic dependencies, making them appropriate for modeling complex stochastic processes. Applied to factorial HMMs, they generalize the work by Ghahramani & Jordan (1997) in two important directions: by introducing coupled dynamics, and by considering non-Gaussian observations. They also give rise to a simple iterative procedure for making inferences about the hidden states. We describe this procedure not only for its practical value, but also because it very cleanly illustrates the idea of exploiting tractable substructures in intractable probabilistic networks (Saul & Jordan, 1997).

The main significance of this work lies in its application to factorial HMMs and the modeling of coupled time series. In principle, though, mixed memory Markov models can be applied wherever 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 note 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.

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 of this matrix, we consider parameterizing the model by the convex combination (Raftery, 1985; Ney et al., 1994):

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

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 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.

The mixture model in eq. (1) is to be distinguished from models that approximate higher-order Markov models by "n-gram smoothing"; that is, by employing a linear combination of nth order transition matrices (Chen & Goodman, 1996). Our model is not an n-gram smoother; rather it approximates a higher-order Markov model by taking a linear combination of non-adjacent bigram models. The model in eq. (1) also differs from mixture-of-experts models as applied to continuous time series (Zeevi et al, 1996), in which the predictions of different nth order regressors are combined by the weights of a softmax gating function.

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:

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

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 prior probabilities, $P(x_t = \mu)$. Likewise, we can view the parameters $a^{\mu}(i'|i)$ as the conditional probabilities, $P(i_t = i'|i_{t-1}, \ldots, i_{t-k}, x_t = \mu)$.

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 (Dempster et al., 1977). In general terms the EM algorithm calculates *expected* sufficient statistics and sets them equal to the *observed* sufficient statistics. The procedure iterates and is guaranteed to increase the likelihood at each step. For the model in eq. (2), the EM updates are (Ney et al, 1994):

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

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

Table 1. Entropy per character, computed from various Markov models.

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

In the case where multiple time series are available as training data, the sums over t should be interpreted as sums over series as well. The EM updates for this model are easy to understand; at each 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:

$$P(x_t = \mu | I) = \frac{\psi(\mu)\ a^{\mu}(i_t | i_{t-\mu})}{\sum_{\nu} \psi(\nu)\ a^{\nu}(i_t | i_{t-\nu})}. \tag{5}$$

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. The matrices $a^{\nu}(i'|i)$ were initialized by count-based bigram models predicting each letter by the μth preceding one. (This type of initialization, in which the component sub-models are first trained independently of one another, is useful to avoid poor local maxima in the learning procedure.) In table 1, we give the results measured in entropy per character. The results show that the mixed memory model does noticeably better than the first-order model. Of course, it cannot capture all the structure of the full second-order model, which has over ten times as many parameters. The mixture model should accordingly be viewed as an intermediate step between first and higher-order 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. A real-world example 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 on the Internet (Bestavros & Cunha, 1995; Cunha et al., 1995). Typically, the choice of the next web page 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.) The model in eq. (1) captures this type of conditioning explicitly. Here, the states of the Markov model would correspond to web pages; the matrices $a^{\nu}(i'|i)$, to links from web page i to web page j; and the index μ, to the number of backward or retraced steps taken before activating a new link.

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. Ney et al (1994) have investigated models of this form for large vocabulary 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 (Chen & Goodman, 1996) 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). Recently, one of us applied a more sophisticated version of eq. (1) to large vocabulary language modeling (Saul & Pereira, 1997). In only a few CPU hours, it was possible to fit over ten million parameters to the statistics of an eighty million word corpus. Moreover, the smoothed combination of mixed memory and higher order Markov models led to significantly lower entropies on out-of-sample predictions.

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 (Dirst & Weigend, 1993). 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 make two simplifying assumptions: (i) that the components i_t^ν at time t are conditionally independent given the vector I_{t-1}, or

$$P(I_t|I_{t-1}) = \prod_{\nu=1}^{k} P(i_t^\nu|I_{t-1}); \qquad (6)$$

and (ii) that the conditional probabilities $P(i_t^\nu|I_{t-1})$ can be expressed as a weighted sum of "cross-transition" matrices:

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

Here again, the parameters $a^{\nu\mu}(i'|i)$ are k^2 elementary $n \times n$ transition matrices, while 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 $O(k^2 n^2)$, as opposed to $O(n^{2k})$ for the full memory

Table 2. Portion of the four-component time series generated by Bach's last fugue.

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

model. (By allowing non-square transition matrices, this model can also be generalized to the case where the different components take on different numbers of values.)

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 $\psi^\nu(\mu)$ is the identity matrix, and the νth component is independent of all the rest. On the other hand, for non-zero $\psi^\nu(\mu)$, all the components at one time step influence the νth component at the next. 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:

$$P(i_t^\nu, x_t^\nu = \mu | I_{t-1}) = \psi^\nu(\mu)\, a^{\nu\mu}(i_t^\nu | i_{t-1}^\mu), \tag{8}$$

$$P(I_t, X_t | I_{t-1}) = \prod_\nu P(i_t^\nu, x_t^\nu | I_{t-1}). \tag{9}$$

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 model. In this case, the EM updates are:

$$\psi^\nu(\mu) \leftarrow \frac{\sum_t P(x_t^\nu = \mu | I)}{\sum_{t,\mu'} P(x_t^\nu = \mu' | I)} \tag{10}$$

$$a^{\nu\mu}(i'|i) \leftarrow \frac{\sum_t P(x_t^\nu = \mu, i_{t-1}^\mu = i, i_t^\nu = i' | I)}{\sum_t P(x_t^\nu = \mu, i^m u_{t-1} | I)} \tag{11}$$

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. This fugue has a rich history (Dirst & Weigend, 1993). The time series (3284 beats long) was made public following the Santa Fe competition on time series prediction. Table 2 shows a portion of this time series: here, each element represents a sixteenth note, while the numerical value codes the pitch. To help avoid poor local maxima in the learning procedure, the transition matrices $a^{\nu\mu}(i'|i)$ were initialized by count-based bigram models predicting the νth voice at time t from the μth voice at the previous time step.

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. The reason for this is that the voices do not typically change pitch with every sixteenth note. Hence, for each voice the note at the previous beat is a very good predictor of the note at the current 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*

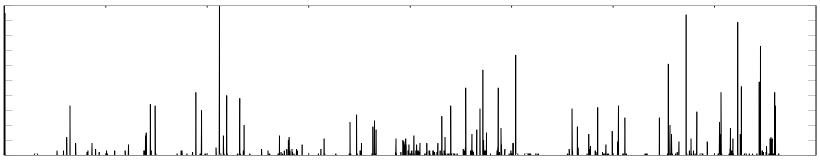

Figure 1. Plot of soprano-tenor correlations versus time, as measured by the posterior probabilities of a mixed memory Markov model.

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 the previous time step. 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 discernible local 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 (Ghahramani & Jordan, 1997; Binder et al., 1997). 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 (Bregler & Omohundro, 1995), where information from different modalities is being combined and presented to the recognizer. It is also the case in frequency subband-based speech recognition (Bourlard & Dupont, 1996), 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.

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 exist for the problem of parameter estimation.

For concreteness, suppose that we have trained k simple HMMs on separate time series of length L. 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. If each individual HMM had n hidden states and m types of observations, then the hidden state space of the combined model has size n^k; likewise, the observation space of the combined model has size m^k. At each time step, we denote these spaces by the Cartesian products:

$$I_t = i_t^1 \otimes i_t^2 \otimes \cdots \otimes i_t^k \qquad \text{(hidden)}, \tag{12}$$

$$J_t = j_t^1 \otimes j_t^2 \otimes \cdots \otimes j_t^k \qquad \text{(observed)}. \tag{13}$$

In an HMM, it is the hidden states (as opposed to the observations) that have a Markov dynamics. Accordingly, in this setting, we use eq. (6–7) to model the hidden state transition matrix. By analogy to eqs. (6–7), we parameterize the emission probabilities by:

$$P(J_t|I_t) = \prod_\nu \sum_\mu \phi^\nu(\mu) b^{\nu\mu}(j_t^\nu|i_t^\mu), \tag{14}$$

where $b^{\nu\mu}(j|i)$ are k^2 elementary $n \times m$ emission 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. (14). By analogy to eqs. (8) and (9), we have:

$$P(j_t^\nu, y_t^\nu = \mu|I_t) = \phi^\nu(\mu) b^{\mu\nu}(i_t^\mu, j_t^\nu), \tag{15}$$

$$P(J_t, Y_t|I_t) = \prod_\nu P(j_t^\nu, y_t^\nu|I_t). \tag{16}$$

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:

$$\psi^\nu(\mu) \leftarrow \frac{\sum_t P(x_t^\nu = \mu|J)}{\sum_{t,\mu'} P(x_t^\nu = \mu'|J)}, \tag{17}$$

$$a^{\nu\mu}(i'|i) \leftarrow \frac{\sum_t P(x_t^\nu = \mu, i_{t-1}^\mu = i, i_t^\nu = i'|J)}{\sum_t P(x_t^\nu = \mu, i_{t-1}^\mu = i|J)}, \tag{18}$$

$$\phi^\nu(\mu) \leftarrow \frac{\sum_t P(y_t^\nu = \mu|J)}{\sum_{t,\mu'} P(y_t^\nu = \mu'|J)}, \tag{19}$$

$$b^{\nu\mu}(j|i) \leftarrow \frac{\sum_t P(y_t^\nu = \mu, i_t^\mu = i, j_t^\nu = j|J)}{\sum_t P(y_t^\nu = \mu, i_t^\mu = i|J)} \tag{20}$$

where J denotes the observed time series. A Viterbi approximation is obtained by conditioning not only on J, but also on the most probable sequence of hidden states, I^*, where

$$I^* = \arg\max_I \prod_t P(I_t|I_{t-1})P(J_t|I_t). \tag{21}$$

Note that computing the posterior probabilities in these updates requires $O(Ln^{2k})$ operations; the same is true for computing the Viterbi path. To avoid this computational burden, we have used an approximation for estimating the statistics in factorial HMMs, first outlined in Saul & Jordan (1996). 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 note, we discuss how these approximations can be used to estimate the Viterbi path. In general, these ideas may be extended to approximate the full statistics of the posterior distribution, as for example in Ghahramani & Jordan (1997).

In the factorial HMM, dynamic programming procedures to compute the Viterbi path algorithm require $O(Ln^{2k})$ steps. As a practical alternative, we consider an iterative procedure that returns a (possibly sub-optimal) path in polynomial time. 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. (21) 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 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 judgment 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 as strong or stronger

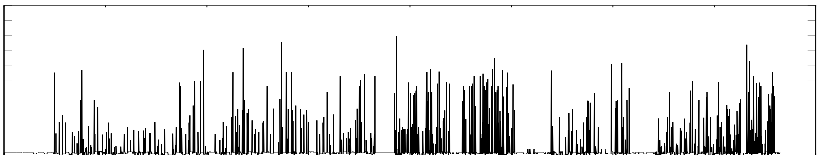

Figure 2. Plot of soprano-tenor correlations versus time, as measured by the posterior probabilities of a mixed memory HMM.

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. One hopes in this model that the hidden states will reflect musical structure over longer time scales than a single note. In our experiments, 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. (17)–(20) 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. Again it was interesting to see how this model discovered correlations between the different voices of the fugue. Figure 2 shows a plot of the posterior probabilities $P(x_t^S = T|I^*, J)$ versus time, calculated from the factorial HMM (after training). The frequent pulses 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 note 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 potential 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.

Acknowledgments

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. J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning* **29**:213–244 (1997).
5. 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).
6. 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).
7. 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).
8. 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).
9. T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3): 142–150 (1989).
10. 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).
11. 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).
12. 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).
13. Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine Learning* **29**:245–273 (1997).
14. 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).
15. I. MacDonald and W. Zucchini. *Hidden Markov and Other Models for Discrete-Valued Time Series*. Chapman and Hall (1997).
16. A. Nadas. Estimation of probabilities in the language model of the IBM speech recognition system. *IEEE Transactions on ASSP* 32(4): 859–861 (1984).
17. H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language* **8**:1–38 (1994).
18. L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286 (1989).
19. A. Raftery. A model for high-order Markov chains. *Journal of the Royal Statistical Society* **B47**:528–539 (1985).
20. D. Ron, Y. Singer, and N. Tishby. The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning* **25**:117–150 (1996).
21. 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).
22. L. Saul and F. Pereira. Aggregate and mixed-order Markov models for statistical language processing. *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*, 81–89 (1997).
23. 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).
24. J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden Markov models. *Proceedings of the 1992 International Conference on Computer Vision*, 379–385 (1992).
25. A. Zeevi, R. Meir, and R. Adler. Time series prediction using mixtures of experts. In M. Mozer, M. Jordan, and T. Petsche (eds.), *Advances in Neural Information Processing Systems* 9:309–315 (1997).