LETTER Communicated by Zoubin Ghahramani

Factorial Hidden Markov Models and the Generalized Backfitting Algorithm

Robert A. Jacobs
robbie@bcs.rochester.edu
Department of Brain and Cognitive Sciences, University of Rochester, Rochester, NY 14627, U.S.A.

Wenxin Jiang
jiang@orie.cornell.edu
School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, U.S.A.

Martin A. Tanner
tanm@neyman.stats.nwu.edu
Department of Statistics, Northwestern University, Evanston, IL 60208, U.S.A.

Previous researchers developed new learning architectures for sequential data by extending conventional hidden Markov models through the use of distributed state representations. Although exact inference and parameter estimation in these architectures is computationally intractable, Ghahramani and Jordan (1997) showed that approximate inference and parameter estimation in one such architecture, factorial hidden Markov models (FHMMs), is feasible in certain circumstances. However, the learning algorithm proposed by these investigators, based on variational techniques, is difficult to understand and implement and is limited to the study of real-valued data sets. This chapter proposes an alternative method for approximate inference and parameter estimation in FHMMs based on the perspective that FHMMs are a generalization of a well-known class of statistical models known as generalized additive models (GAMs; Hastie & Tibshirani, 1990). Using existing statistical techniques for GAMs as a guide, we have developed the generalized backfitting algorithm. This algorithm computes customized error signals for each hidden Markov chain of an FHMM and then trains each chain one at a time using conventional techniques from the hidden Markov models literature. Relative to previous perspectives on FHMMs, we believe that the viewpoint taken here has a number of advantages. First, it places FHMMs on firm statistical foundations by relating them to a class of models that are well studied in the statistics community, yet it generalizes this class of models in an interesting way. Second, it leads to an understanding of how FHMMs can be applied to many different types of time-series data, including Bernoulli

and multinomial data, not just data that are real valued. Finally, it leads to an effective learning procedure for FHMMs that is easier to understand and easier to implement than existing learning procedures. Simulation results suggest that FHMMs trained with the generalized backfitting algorithm are a practical and powerful tool for analyzing sequential data.

1 Introduction

Researchers in the neural computation community often find it useful to distinguish between single-cause and multiple-cause generative models. In single-cause models, each data item is generated by an individual source. Mixture models are a good example of single-cause models because individual data items are generated by sampling from a single mixture component. In multiple-cause models, in contrast, each data item is generated by combining the influences of multiple sources. Factor models are an important example of multiple-cause models. Individual data items in this case are generated by linearly combining the values of a set of latent variables and then adding gaussian noise to this sum.

Recently the distinction between single-cause and multiple-cause models has become important when studying sequential data such as time-series data. A conventional method for summarizing sequential data is to use hidden Markov models (HMMs), which are a generalization of mixture models. In addition to several mixture components, they include a state variable that indicates the mixture component active at the current time step and a set of transition probabilities that indicates the likelihood that each component will become active given the previously active component. Consider the time-series data set $\mathcal{Y} = \{y^{(t)}\}_{t=1}^{T}$ where $y^{(t)}$ is a scalar and T is the duration of the series.[1] The assumed generative model is that at each time step, a new active component is selected according to the transition probabilities, and an output of the model is sampled from the selected component. This is illustrated in Figure 1(A). The vector $\mathbf{q}$ in this figure is the model's state variable, and y is its output variable. The elements of $\mathbf{q}$ are equal to zero except for the element corresponding to the active mixture component; this element is set to one. Initially (at $t = 1$), the active mixture component is sampled from a multinomial distribution, and the output $y^{(1)}$ is sampled from the chosen component. Subsequently (for $t > 1$), a new active component is sampled from a multinomial distribution whose parameters depend on the previous active component; then the output $y^{(t)}$ is generated by the new active mixture component. This process continues for the duration of the time series.

[1] Without loss of generality, we limit our discussion to one-dimensional data. The extension to multidimensional data is straightforward.

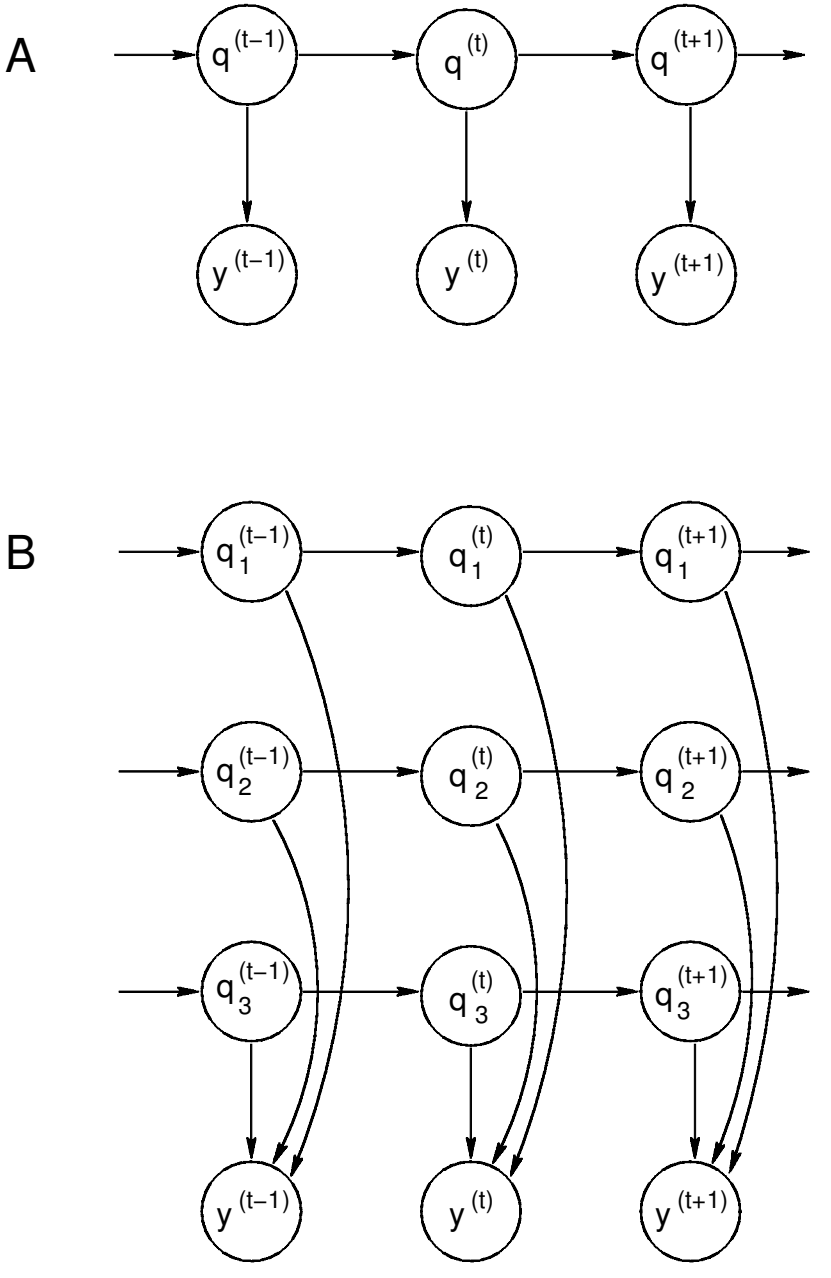

Figure 1: (A) A graphical representation of a hidden Markov model. (B) A graphical representation of a factorial hidden Markov model with three hidden Markov chains.

HMMs have several attractive features. They can model data that violate the stationarity assumptions characteristic of many other time-series models. In addition, there exists an expectation-maximization (EM) algorithm, known as the Baum-Welch algorithm, for finding maximum likelihood estimates of the parameter values of an HMM (Baum, Petrie, Soules, & Weiss, 1970; Rabiner & Juang, 1993). This algorithm is easy to understand and easy to implement. Unfortunately, HMMs have disadvantages too, and researchers continue to look for better models. An important disadvantage of HMMs is that they are instances of single-cause models and thus are inefficient from a representational viewpoint. The history of a time series up to time step t is represented by the active mixture component at time t indicated by the multinomial state variable $\mathbf{q}^{(t)}$. As Williams and Hinton (1991) pointed out, the multinomial assumption makes HMMs representationally inefficient. Due to this assumption, HMMs need 2^N possible values for the state variable (i.e., 2^N mixture components) to represent N bits of information about the history of a time series.

To remedy the representational inefficiency of HMMs, Williams and Hinton (1991) proposed generalizing HMMs through the use of distributed state representations. Ghahramani and Jordan (1997) studied a special case of this generalization, which they referred to as factorial hidden Markov models (FHMMs), and considered the use of FHMMs for summarizing real-valued time-series data. The most important feature of FHMMs is that they consist of multiple hidden Markov chains (see Figure 1B). Each of the chains contains a number of components along with a multinomial state variable that indicates the component that is active at the current time step and a set of transition probabilities that gives the likelihood that each component will become active given the previously active component. Chains are independent of each other; they do not interact when generating data. The output of the model at each time step is, however, dependent on the values of the state variables of all the chains. Hence, FHMMs are instances of multiple-cause generative models. Real-valued time-series data are generated as follows. At each time step, each chain determines its active component by sampling from a multinomial distribution whose parameters depend on the previous active component. Then an output contribution is made by each chain that is dependent on the chain's currently active component. The sum of the contributions of all the chains is the mean of a gaussian distribution, and a sample from this distribution is the output of the entire model. In contrast to conventional HMMs, FHMMs are efficient from a representational viewpoint. If each chain has two components, meaning that each chain's state variable is binary, then an FHMM needs N state variables (i.e., N hidden Markov chains with a total of $2N$ components) to represent N bits of information about the history of a time series.

Factorial hidden Markov models are a special case of dynamic Bayesian networks (DBNs). These networks are directed graphical models of stochas-

tic processes that generalize HMMs by representing the hidden state in terms of state variables with possibly complex interdependencies. Due to these interdependencies, exact maximum likelihood estimation is typically computationally intractable in DBNs, though the study of learning procedures that perform approximate estimation is an active area of research (Boyen & Koller, 1999; Dean & Kanazawa, 1988; Murphy & Weiss, 2001). FHMMs are a special case of DBNs in the sense that they use multiple state variables, though they do not permit the state variables to interact with each other.

Although FHMMs are appealing generative models, they are difficult to use for the purposes of statistical inference. In particular, the EM algorithm for finding maximum likelihood estimates of an FHMM's parameter values is computationally intractable (Ghahramani & Jordan, 1997). The problem occurs during the E-step of the algorithm when one is attempting to infer the expected value of each chain's state variable at each time step given the time-series data. Although the state variables of the hidden Markov chains are marginally independent, these variables are conditionally dependent given the time-series data. Intuitively, the computational intractability stems from the cooperative nature of the model; the settings of all the state variables cooperate in determining the mean of the time-series data at each time step. When inferring the expected value of one chain's state variable given the time-series data, it is therefore necessary to sum over all possible values of the other chains' state variables. Ghahramani and Jordan developed a procedure for performing inference in FHMMs through the use of the junction tree algorithm, a popular algorithm for performing inference in graphical models. The time complexity of the procedure is $O(TMK^{M+1})$, where T is the duration of the time series, M is the number of hidden Markov chains, and K is the number of components in each chain. This exponential time complexity makes exact inference and maximum likelihood parameter estimation intractable.

Because exact inference and parameter estimation in FHMMs is intractable, Ghahramani and Jordan (1997) studied approximate inference and parameter estimation. The primary focus of their article was on the use of variational techniques to approximate the posterior distribution of the state variables given the time series. In short, the idea is to approximate this distribution with another distribution that can be computed efficiently and can be shown using Jensen's inequality to provide a basis for a lower bound on the log-likelihood of the time series data. The parameters of the approximating distribution are found by minimizing the Kullback-Leibler distance between this distribution and the true posterior distribution (Jordan, Ghahramani, Jaakkola, and Saul, 1998). The resulting approximating distribution can be used to obtain an efficient parameter estimation algorithm. Ghahramani and Jordan showed that an FHMM trained using the variational approximation captured statistical structure in a time-series data set that a conventional HMM did not.

In this article, we propose an alternative method for approximate inference and parameter estimation in FHMMs that we call the generalized backfitting algorithm. This method is based on the perspective that FHMMs are a generalization of a well-known class of statistical models known as generalized additive models (GAMs; see Hastie & Tibshirani, 1990). GAMs are a method for mapping covariate or input variables to response or output variables and can be used in any setting in which generalized linear models are applicable, such as linear, logistic, or log-linear regression, the analysis of multinomial data, or the analysis of censored survival data. Consider a data set in which covariate variables x_1, x_2, and x_3 are mapped to response variable y: $\{x_1^{(n)}, x_2^{(n)}, x_3^{(n)} \rightarrow y^{(n)}\}_{n=1}^{N}$ where n indexes individual data items. When regarded as a generative model, GAMs assume that each data item is generated by first summing contributions made on the basis of individual covariate variables and then mapping this sum to a conditional mean of the response using a function h:

$$E[y^{(n)}|x_1^{(n)}, x_2^{(n)}, x_3^{(n)}] = h[f_1(x_1^{(n)}) + f_2(x_2^{(n)}) + f_3(x_3^{(n)})], \tag{1.1}$$

where $f_1(x_1^{(n)})$, $f_2(x_2^{(n)})$, and $f_3(x_3^{(n)})$ are the contributions made on the basis of the individual covariates, and h is an appropriately selected monotonic and differentiable function. The response $y^{(n)}$ is then sampled from an appropriate distribution with the given conditional mean.

When used for statistical inference, GAMs are appealing because they are simple. Rather than consider the relationship between all covariate variables and a response variable, GAMs attempt to circumvent the "curse of dimensionality" by iteratively considering each individual covariate variable one at a time. This training procedure for GAMs is known as the backfitting algorithm. When the parameters of a GAM are estimated, a mapping is learned from each covariate variable to a residual error, where the error is based on the value of the response variable and the sum of the contributions based on the remaining covariate variables. Consider the case where the response is real valued, and we are interested in updating the parameters of f_1, the function giving the contribution based on covariate x_1. Then the residual error $e_1^{(n)}$ on data item n that corresponds to $x_1^{(n)}$ is

$$e_1^{(n)} = y^{(n)} - [f_2(x_2^{(n)}) + f_3(x_3^{(n)})]. \tag{1.2}$$

The parameters of f_1 are updated so that $f_1(x_1^{(n)})$ more closely approximates $e_1^{(n)}$ for each data item. The backfitting algorithm cycles through updates of the functions f_1, f_2, and f_3 until convergence.

The parameter estimation method for FHMMs proposed in this article takes advantage of the fact that FHMMs are additive models; at each time step, each hidden Markov chain makes an output contribution based on the value of its state variable, and the sum of these contributions is used to

determine the conditional mean of the time-series data. As a result, lessons learned from the study of GAMs can be applied to FHMMs. The method treats learning in FHMMs and GAMs as similar in the sense that it uses an iterative procedure that considers each chain's state variable one at a time. More precisely, it considers the mapping at each time step between each chain's state variable and a residual error where the error is based on the current value of the time-series data and the sum of the contributions based on the remaining chains' state variables. However, learning in FHMMs is also different from learning in GAMs. Whereas each stage of learning in GAMs considers the known value of a covariate variable, each stage of learning in FHMMs considers the unknown value of a hidden state variable. Fortunately, however, because each chain's state variable is considered separately, the expected value of this variable can be computed efficiently using standard hidden Markov model techniques. The end result is a learning algorithm that closely resembles the backfitting algorithm used to train GAMs. Like the conventional backfitting algorithm, the proposed procedure is simple and effective. Unlike the conventional backfitting algorithm, however, the proposed procedure does not produce maximum likelihood parameter estimates. The need to estimate the expected values of the chains' state variables means that the procedure is an approximate method.

The article is organized as follows. Section 2 provides an overview of GAMs, describes how FHMMs can be viewed as generalizations of them, and describes the generalized backfitting algorithm. Relative to previous perspectives on FHMMs, we believe that the viewpoint taken here has a number of advantages. First, it places FHMMs on a firm statistical foundation by relating FHMMs to a class of models that is well studied in the statistics community, yet it generalizes this class of models in an interesting way. Second, it leads to an understanding of how FHMMs can be applied to many different types of time-series data, including Bernoulli and multinomial data, not just data that are real valued. Finally, it leads to an effective learning procedure for FHMMs that is easier to understand and easier to implement than existing learning procedures. Using this learning procedure, we believe that FHMMs are now a practical tool for analyzing sequential data. Section 3 provides some analytical results regarding the application of the generalized backfitting algorithm to FHMMs. This section clarifies several of the theoretical properties of the algorithm. Section 4 presents simulation results comparing the performances of FHMMs trained with the generalized backfitting algorithm to FHMMs trained with other learning procedures. The results suggest that FHMMs trained with any of the candidate algorithms have similar levels of performance when summarizing real-valued time series and that the new learning procedure proposed here outperforms alternative algorithms when summarizing multinomial time series. A summary and conclusions are presented in section 5.

2 FHMMs and GAMs

We begin this section by providing a brief overview of GAMs. A thorough treatment can be found in Hastie and Tibshirani (1990).

GAMs are closely related to generalized linear models (GLIMs) and can be understood by comparing them with GLIMs (McCullagh & Nelder, 1989). From a generative perspective, GLIMs generate data items using a three-stage process. First, a linear combination of the covariate variables, denoted η, is formed,

$$\eta = \sum_{m=1}^{M} x_m \beta_m, \tag{2.1}$$

where $x_1, \ldots, x_M$ is a set of covariates and $\beta_1, \ldots, \beta_M$ is a set of coefficients. Next, a monotonic and differentiable function, denoted h, maps the linear combination η to the expected value of the response variable y given the values of the covariates:

$$E(y \mid x_1, \ldots, x_M) = h(\eta). \tag{2.2}$$

Finally, the value of the response y is sampled from an output distribution. The choices of the distribution and the function h depend on the nature of the response variable. If, for example, the response y is real valued, then it is common to assume that y has a gaussian distribution and that the function h is the identity function. As a second example, if the response y is binary, then it is assumed that y has a Bernoulli distribution and that the function h is the logistic function. In general, the distribution of the response is a member of the exponential family of distributions. If h is chosen so that η equals the natural parameter of the distribution, then h^{-1} is known as the canonical link function for the distribution.

For the purposes of statistical inference, estimates of the values of the coefficients $\beta_1, \ldots, \beta_M$ are typically obtained using the iteratively reweighted least-squares (IRLS) algorithm. This algorithm is an instance of a Newton-Raphson algorithm, and if h is chosen so that its inverse is a canonical link function, then the algorithm is also a special case of the Fisher scoring procedure. In short, the linearized response $z^{(n)}$ for the nth data item is computed as the first-order Taylor's series approximation to $h^{-1}(y^{(n)})$ about the current estimate $E(y^{(n)} \mid x_1^{(n)}, \ldots, x_M^{(n)})$. New estimates of the coefficients are formed by weighted linear regression of the linearized responses for all data items onto the covariate variables with weights that are proportional to the variances of the linearized responses (see McCullagh & Nelder, 1989, for details). This process is repeated until the parameter estimates converge. The IRLS algorithm is attractive because no special optimization software is required, just a function that computes weighted least-squares estimates.

Consider the case when the response variable y is real valued with a gaussian distribution and the function h is the identity function. Then the linear combination $\eta = \sum_m x_m \beta_m$ is the conditional mean of a gaussian distribution given values for the covariates. When the IRLS algorithm is used to estimate the values of the coefficients, the linearized response z is equal to the actual response y and the weights are equal to one. That is, the IRLS algorithm reduces to conventional linear regression. As a second example, consider the case when the response y is binary with a Bernoulli distribution and the function h is the logistic function. In this case, the linearized response (i.e., the first-order Taylor's series approximation described above) is

$$z = \eta + \frac{y - h(\eta)}{h(\eta)(1 - h(\eta))}, \tag{2.3}$$

and the weights (i.e., the variances of the linearized responses) are

$$w = h(\eta)(1 - h(\eta)). \tag{2.4}$$

Intuitively, a weight is small when the variance is small because the response y is close to its expected value $h = E[y]$ in this case and, thus, the residual $y - h$ is small. In contrast, a weight is large when the variance is large. The IRLS algorithm performs a weighted least-squares regression of the linearized responses onto the covariates. This regression must be iterated because new estimates of the coefficients lead to new linearized responses and weights.

GAMs differ from generalized linear models in that an additive combination replaces the linear combination. From a generative perspective, GAMs generate data items using a three-stage process. First, an additive combination based on the covariates, denoted η, is formed:

$$\eta = \sum_{m=1}^{M} f_m(x_m), \tag{2.5}$$

where the f_ms are arbitrary univariate functions, one for each covariate. Next, a monotonic and differentiable function h maps the additive combination η to the expected value of the response variable y given the values of the covariates:

$$E(y \mid x_1, \ldots, x_M) = h(\eta). \tag{2.6}$$

Finally, the value of the response y is sampled from an output distribution. As in the case of GLIMs, the choices of the distribution and the function h depend on the nature of the response variable.

When statistical inference is performed, the parameters of the f_ms need to be estimated. The IRLS algorithm for GAMs is essentially unchanged from

its form for GLIMs with the exception that a weighted additive regression is used instead of a weighted linear regression. First, linearized responses z and weights w are computed for all data items. Then the parameters of the f_ms are updated by performing a weighted additive regression of the linearized responses onto the covariates. The backfitting algorithm is used to perform this weighted additive regression.

When applied to GAMs, the IRLS algorithm defines the linearized response z and the weight w in the same way as when the algorithm is applied to GLIMs. For example, if the response variable is real valued with a gaussian distribution and the function h is the identity function, then the linearized response z equals the actual response y and the weights equal one. If the response variable is binary with a Bernoulli distribution and the function h is the logistic function, then the linearized response and weight are given by equations 2.3 and 2.4, respectively. The backfitting algorithm iteratively considers each individual covariate variable one at a time in order to perform the weighted additive regression. It first forms a residual error equal to the difference between the value of the linearized response and the sum of the contributions based on the remaining covariate variables. The residual error corresponding to the mth covariate on the nth data item is

$$e_m^{(n)} = z^{(n)} - \sum_{i \neq m} f_i(x_i^{(n)}). \tag{2.7}$$

Next, the parameters of f_m are updated so that the contribution based on the mth covariate approximates the error $e_m^{(n)}$ for each data item in proportion to the weight for that data item. This process is repeated for all of the covariates, possibly multiple times.

An important point of this article is that factorial hidden Markov models can be regarded as a generalization of GAMs. From a generative perspective, the three-stage process by which FHMMs generate data is similar to the process by which GAMs generate data. First, at each time step, each hidden Markov chain makes an output contribution based on the value of its hidden state variable, and an additive combination of these contributions is formed. The contribution of chain m, denoted $f_m(\mathbf{q}_m)$, is given by $f_m(\mathbf{q}_m) = \boldsymbol{\mu}_m' \mathbf{q}_m$, where $\boldsymbol{\mu}_m$ is a K-dimensional vector of parameters ($\boldsymbol{\mu}_m'$ is the transpose of $\boldsymbol{\mu}_m$). The additive combination of the output contributions from all M chains, denoted η, is

$$\eta = \sum_{m=1}^{M} f_m(\mathbf{q}_m). \tag{2.8}$$

Next, a monotonic and differentiable function h maps the additive combination η to the expected value of the response variable y given the values of the state variables:

$$E(y \mid \mathbf{q}_1, \ldots, \mathbf{q}_M) = h(\eta). \tag{2.9}$$

Finally, the value of y is sampled from an output distribution. As in the case of GLIMs and GAMs, the choices of the distribution and the function h depend on the nature of the response variable. Note that both GAMs and FHMMs map a set of variables—covariate variables $x_1, \ldots, x_M$ in the case of GAMs and hidden state variables $\mathbf{q}_1, \ldots, \mathbf{q}_M$ in the case of FHMMs—to a response variable. Both of these classes of models also perform this mapping using an additive combination, not a linear combination, and then map this combination to a response using the inverse of a canonical link function.

When performing statistical inference using FHMMs, it is important to keep in mind that GAMs and FHMMs have both similarities and differences. The most significant difference is that GAMs consider the known values of covariate variables, whereas FHMMs consider the unknown values of hidden state variables. The similarities between the two classes of models mean that the parameter estimation algorithm for GAMs can be a useful guide to parameter estimation in FHMMs. The differences mean that the estimation algorithm for GAMs cannot be applied to FHMMs in its exact form, but some modifications will be required. Because the values of the hidden state variables are unknown, the algorithm for parameter estimation in FHMMs will require estimates of the expected values of these state variables. Fortunately, estimates of these expected values can be computed efficiently using standard techniques from the hidden Markov models literature.

The values of the free parameters of FHMMs need to be estimated during statistical inference. The IRLS algorithm for FHMMs is similar to its form for GAMs. In particular, a modified version of the backfitting algorithm is used to perform weighted additive regressions. We refer to this modified version as the generalized backfitting algorithm. The algorithm first defines the linearized responses and the weights. Once again, if the response variable is real valued with a gaussian distribution and the function h is the identity function, then the linearized response z equals the actual response y, and the weights equal one. If the response variable is binary with a Bernoulli distribution and the function h is the logistic function, then the linearized response and weight are given by equations 2.3 and 2.4. The algorithm then iteratively considers each individual state variable one at a time (or, equivalently, each individual hidden Markov chain one at at time). For each state variable, it forms a residual error equal to the difference between the value of the linearized response and the sum of the contributions based on the expected values of the remaining state variables. The residual error corresponding to the mth Markov chain at time t is

$$e_m^{(t)} = z^{(t)} - \sum_{i \neq m} f_i(E[\mathbf{q}_i^{(t)}]), \tag{2.10}$$

where $E[\mathbf{q}_i^{(t)}]$ is the expected value of the hidden state variable for chain i and $f_i(E[\mathbf{q}_i^{(t)}])$ is the expected contribution of chain i. Next, the parameters of the mth Markov chain are updated so that the contribution based on the

mth state variable approximates the error $e_m^{(t)}$ for all time steps in proportion to the weight for that time step. This process is repeated for all of the Markov chains, possibly multiple times.

In order to understand learning in FHMMs better, it is worth pointing out an important feature of the IRLS algorithm that leads to an interesting viewpoint regarding FHMMs and multicomponent or modular architectures. This viewpoint follows from the fact that the set of residual errors $\{e_m^{(t)}\}_{t=1}^T$ is itself a time-series data set. Because this is a real-valued time series, it can be modeled by a standard hidden Markov model with gaussian responses. In other words, FHMMs can be regarded as instances of multicomponent or modular architectures where the modules are the hidden Markov chains. Our analysis indicates that these chains should be implemented as HMMs with gaussian responses. From a generative perspective, the output of an FHMM as a whole at time t is formed by first summing the expected outputs of each of the individual HMM modules, mapping this sum to an expected value of an output distribution using an inverse canonical link function (i.e., the function h), and then sampling from this distribution in order to generate the target time-series data item $y^{(t)}$.

From the perspective of statistical inference, the fact that each HMM module's set of residual errors is itself a time-series data set is good news. The individual HMM modules can be quickly and effectively trained using conventional parameter estimation algorithms from the literature on hidden Markov models, such as the Baum-Welch algorithm. As a result, FHMMs as a whole can be efficiently trained. This viewpoint highlights the strengths of the backfitting approach to training additive models. When considering learning in any modular architecture, it is important to consider how error signals can be calculated that are customized for each of the architecture's individual modules. In the case of FHMMs, the generalized backfitting equation solves this problem (see equation 2.10). The time-series data $\{e_m^{(t)}\}_{t=1}^T$ are residual errors that are specifically tailored for HMM module m. HMM module m is then trained using the Baum-Welch algorithm to summarize the time series $\{e_m^{(t)}\}_{t=1}^T$, and this process is repeated for all modules.

The pseudocode in Figure 2 outlines the IRLS procedure applied to FHMMs. When implementing this procedure, we have found that it is relatively easy to go from a computer program for simulating conventional HMM models to a program for simulating FHMM models. The majority of the code for FHMMs is a subroutine that implements the Baum-Welch algorithm in individual HMM modules.

To complete the specification of the IRLS procedure, two more issues need to be dealt with. First, training data for HMMs typically do not include weights $\{w^{(t)}\}_{t=1}^T$ on the responses, and so we need to describe how the Baum-Welch algorithm can be modified so that its updates of an HMM's parameters take into account these weights. To do this, we introduce some

repeat until convergence do
  for hidden Markov chain $m = 1$ *to* $m = M$ *do*
    compute linearized responses $\{z^{(t)}\}_{t=1}^{T}$ *and weights* $\{w^{(t)}\}_{t=1}^{T}$
    compute time series $\{e_m^{(t)}\}_{t=1}^{T}$
    Baum-Welch algorithm: train chain m to summarize
      time series $\{e_m^{(t)}\}_{t=1}^{T}$ *using weights* $\{w^{(t)}\}_{t=1}^{T}$
  end
end

Figure 2: Pseudocode outlining the IRLS procedure applied to FHMMs

notation. Let $q_{mi}^{(t)}$ denote the ith element of state vector $\mathbf{q}_m^{(t)}$. Let $\gamma_{mi}^{(t)}$ denote the probability that component i of the mth HMM module is active at time t given the residual error time series: $\gamma_{mi}^{(t)} = p(q_{mi}^{(t)} = 1 | \{e_m^{(t)}\}_{t=1}^{T})$. Finally, let $\xi_{mij}^{(t)}$ denote the probability that component i of HMM module m is active at time t and that component j is active at time $t+1$ given the residual error time series: $\xi_{mij}^{(t)} = p(q_{mi}^{(t)} = 1, q_{mj}^{(t+1)} = 1 | \{e_m^{(t)}\}_{t=1}^{T})$. At each iteration of the Baum-Welch algorithm, the transition probability that component j will become active at time $t+1$ given that component i is active at time t is given by

$$p(q_{mj}^{(t+1)} = 1 \mid q_{mi}^{(t)} = 1) = \frac{\sum_{t=1}^{T-1} \xi_{mij}^{(t)} \, w^{(t)}}{\sum_{t=1}^{T-1} \gamma_{mi}^{(t)} \, w^{(t)}}. \tag{2.11}$$

The update equations for the mean and variance associated with the ith component of the mth HMM module, denoted μ_{mi} and σ_{mi}^2, respectively, are:

$$\mu_{mi} = \frac{\sum_{t=1}^{T} \gamma_{mi}^{(t)} \, w^{(t)} \, e_m^{(t)}}{\sum_{t=1}^{T} \gamma_{mi}^{(t)} \, w^{(t)}} \tag{2.12}$$

$$\sigma_{mi}^2 = \frac{\sum_{t=1}^{T} \gamma_{mi}^{(t)} \, w^{(t)} \, (e_m^{(t)} - \mu_{mi})^2}{\sum_{t=1}^{T} \gamma_{mi}^{(t)} \, w^{(t)}}. \tag{2.13}$$

If $w^{(t)} = 1$ for all time steps, then equations 2.11 through 2.13 are identical to the standard Baum-Welch update equations.

We also need to describe how to compute the expected values of each hidden Markov chain's state variable at each time step (i.e., the set $\{E[\mathbf{q}_m^{(t)}]\}_{t=1}^T$ for chains $m = 1, \ldots, M$; these values are used in equation 2.10). We have studied two different methods for computing these values. Empirical results suggest that the methods lead to similar levels of performance. The first method is to set the ith element of the vector $E[\mathbf{q}_m^{(t)}]$ equal to $\gamma_{mi}^{(t)}$, the probability that the ith component of chain m is active at time t given the residual error time series:

$$E[q_{mi}^{(t)}] = \gamma_{mi}^{(t)}. \tag{2.14}$$

We refer to the generalized backfitting algorithm where the expected values of the state variables are computed in this manner as the GBF-γ algorithm. The second method uses the Viterbi algorithm, another common technique in the HMM literature, to compute the single "best" state sequence given the time-series data for each hidden Markov chain. For chain m, this state sequence is defined as the sequence $\{\mathbf{q}_m^{(t)}\}_{t=1}^T$ that maximizes the probability of the sequence given the residual error time series. The expected values $\{E[\mathbf{q}_m^{(t)}]\}_{t=1}^T$ are simply set to this sequence:

$$\{E[\mathbf{q}_m^{(t)}]\}_{t=1}^T = \arg \max_{\{\mathbf{q}_m^{(t)}\}_{t=1}^T} p(\{\mathbf{q}_m^{(t)}\}_{t=1}^T \mid \{e_m^{(t)}\}_{t=1}^T). \tag{2.15}$$

The generalized backfitting algorithm where the expected values of the state variables are computed using the Viterbi algorithm is referred to as the GBF-V algorithm.

3 Some Analytical Results

This section considers some theoretical properties of FHMMs trained with the generalized backfitting algorithm GBF-γ. For simplicity, we consider the case where the observed time series $\mathcal{Y} = \{y^{(t)}\}_{t=1}^T$ is real valued, and thus its expected value is a linear function of the contributions of the individual hidden Markov chains. In the case of a nonlinear model, a linearization transform from y to z (e.g., see equation 2.3) leads to approximately the same conclusions.

Suppose that the observed data are generated by an FHMM with M independent Markov chains determining the data as described above. Let $y_m^{(t)} = f_m(\mathbf{q}_m^{(t)}) = \boldsymbol{\mu}_m' \mathbf{q}_m^{(t)}$ denote the output contribution of chain m at time t. The output of the FHMM as a whole is

$$y^{(t)} \mid \{\mathbf{q}_m^{(t)}\}_{m=1}^M \sim N\left(\sum_{m=1}^M f_m(\mathbf{q}_m^{(t)}), \sigma^2\right). \tag{3.1}$$

When statistical inference is performed, the parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M$ need to be estimated where $\boldsymbol{\theta}_m$ is the set of parameters associated with chain m, including its mean and variance parameters, its component transition probabilities, and possibly its initial component probabilities. For chain m, the GBF-γ algorithm uses the residual error,

$$e_m^{(t),new} = y^{(t)} - \sum_{i \neq m} E[\boldsymbol{\mu}_i' \mathbf{q}_i^{(t)} \mid \{e_i^{(t),old}\}_{t=1}^T]_{\boldsymbol{\theta}=\boldsymbol{\theta}^{old}}, \tag{3.2}$$

and then fits the time series $\{e_m^{(t),new}\}_{t=1}^T$ using a single-chain HMM and the Baum-Welch procedure.

Let $\{y_m^{(t)}\}_{t=1}^T$ be any time series generated by a single-chain HMM with parameters $\boldsymbol{\theta}_m$. What will be the estimated mean and variance of this time series? Is the residual error time series $\{e_m^{(t)}\}_{t=1}^T$ modeled correctly in mean and variance? In other words, is $E(e_m^{(t)}) = E(y_m^{(t)})$? Is $\text{var}(e_m^{(t)}) = \text{var}(y_m^{(t)})$? We demonstrate that the means of the residuals are modeled correctly.

Proposition 1. $E(e_m^{(t)}) = E(y_m^{(t)})$.

Proof.

$$E(e_m^{(t)}) = E(y^{(t)}) - \sum_{i \neq m} E\left(E\left[\boldsymbol{\mu}_i' \mathbf{q}_i^{(t)} \mid \left\{e_i^{(t),old}\right\}_{t=1}^T\right]_{\boldsymbol{\theta}=\boldsymbol{\theta}^{old}}\right) \tag{3.3}$$

$$= E(y^{(t)}) - \sum_{i \neq m} E\left(\boldsymbol{\mu}_i' \mathbf{q}_i^{(t)}\right) \tag{3.4}$$

$$= \sum_{i=1}^{M} E(\boldsymbol{\mu}_i' \mathbf{q}_i^{(t)}) - \sum_{i \neq m} E(\boldsymbol{\mu}_i' \mathbf{q}_i^{(t)}) \tag{3.5}$$

$$= E(\boldsymbol{\mu}_m' \mathbf{q}_m^{(t)}) \tag{3.6}$$

$$= E(y_m^{(t)}). \tag{3.7}$$

This shows that the means are equal.

Proposition 1 does not imply that the distribution of $e_m^{(t)}$ is modeled correctly. To the contrary, we believe that in general, $\text{var}(e_m^{(t)}) \neq \text{var}(y_m^{(t)})$. However, we cannot write an analytic expression for $\text{var}(e_m^{(t)})$ (expressed in terms of $y^{(t)}$ and $\mathbf{q}_i^{(t)}$s), and so we cannot prove this conjecture.

We now consider a more careful specification of what the generalized backfitting algorithm is estimating. Let $P_{\{y_m^{(t)}\}}[\{\tilde{y}_m^{(t)}\}|\boldsymbol{\theta}_m]$ denote the likelihood

of observed data $\{\tilde{y}_m^{(t)}\}_{t=1}^T$ for a single-chain HMM time series:

$$P_{\{y_m^{(t)}\}}[\{\tilde{y}_m^{(t)}\}|\boldsymbol{\theta}_m] = \sum_{\{\mathbf{q}_m^{(t)}\}_{t=1}^T} \pi_{\mathbf{q}_m^{(1)}} P_{\mathbf{q}_m^{(1)},\mathbf{q}_m^{(2)}} \cdots P_{\mathbf{q}_m^{(T-1)},\mathbf{q}_m^{(T)}} (2\pi\sigma^2)^{-T/2} \times \exp\left[-\frac{1}{2\sigma^2}\sum_{t=1}^{T}\left(\tilde{y}_m^{(t)} - \boldsymbol{\mu}_m'\mathbf{q}_m^{(t)}\right)^2\right], \quad (3.8)$$

where $\sum_{\{\mathbf{q}_m^{(t)}\}_{t=1}^T}$ denotes a sum over all possible state sequences, $\pi_{\mathbf{q}_m^{(1)}}$ is the initial ($t = 1$) probability for state variable $\mathbf{q}_m^{(1)}$, and $P_{\mathbf{q}_m^{(t-1)},\mathbf{q}_m^{(t)}}$ is the transition probability of moving from state $\mathbf{q}_m^{(t-1)}$ at time $t-1$ to state $\mathbf{q}_m^{(t)}$ at time t. Although seemingly complicated, this equation has the standard form for an HMM likelihood function.

Proposition 2. *The limiting point of the GBF-γ algorithm satisfies the following fixed-point equation:* $\mathbf{0} = \nabla_{\boldsymbol{\theta}_m} P_{\{y_m^{(t)}\}}[\{e_m^{(t)}\} \mid \boldsymbol{\theta}_m]$, $m = 1, \ldots, M$.

Proof. Using the fact that the Baum-Welch procedure finds maximum likelihood parameter estimates, this is obvious since at each step of the algorithm, the new estimate $\boldsymbol{\theta}_m^{new}$ maximizes the single-chain HMM likelihood and therefore satisfies $\mathbf{0} = \nabla_{\boldsymbol{\theta}_m^{new}} P_{\{y_m^{(t)}\}}[\{e_m^{(t),new}\} \mid \boldsymbol{\theta}_m^{new}]$ where the residuals $\{e_m^{(t),new}\}$ are computed using parameter values and residual errors of other chains from the "old" step.

Several points are worth noting:

- It is easy to extend proposition 2 to the case where there is more than one independent time series since the overall likelihood can be formed by taking products of the likelihoods for the individual series.
- The set of fixed-point equations may not correspond to maximizing any scalar objective function (integrability conditions would need to be satisfied, but this is difficult to check analytically). This lack of an objective function is perhaps the most important drawback of the generalized backfitting algorithm relative to other approximate maximum likelihood algorithms such as the variational algorithm. Fortunately, however, we find that the generalized backfitting algorithm always converges in practice and that it tends to increase monotonically the likelihood of the data at each iteration.
- Using the fact $E(e_m^{(t)}) = E(y_m^{(t)})$ from proposition 1, we conjecture that if the limit point of the algorithm is used to form an estimator of the mean of the time-series data at each time step, denoted $\hat{\mu}^{(t)}$, then this estimator will be consistent (i.e., $\hat{\mu}^{(t)} \xrightarrow{P} E(y^{(t)})$). Let $\boldsymbol{\theta}_N^\infty$ be a limit

point of the algorithm when N independent time series are used. Then $\hat{\mu}^{(t)} = E[\sum_{m=1}^{M} \boldsymbol{\mu}'_m \mathbf{q}_m^{(t)} \mid \boldsymbol{\theta} = \boldsymbol{\theta}_N^{\infty}]$. In other words, although $\boldsymbol{\theta}_N^{\infty}$ is not a maximum likelihood estimator (it may not maximize any objective function, and it is obtained by modeling the variance of $e_m^{(t)}$ incorrectly), it will produce a consistent estimator of the mean function for the observed data. Unfortunately, we cannot prove this conjecture for the general case, though it can be proved for the case of $K = 2$ (each hidden Markov chain has two components) and $T = 1$ (time series with length 1), in which case iteratively fitting HMM models for each chain is equivalent to fitting mixtures of gaussian distributions iteratively.

Finally, and as an aside, it may be worth noting that there could be important similarities between the generalized backfitting algorithm and the variational approximation proposed by Ghahramani and Jordan (1997). Interestingly, both learning procedures make use of the same set of error residuals $\{e_m^{(t)}\}_{t=1}^{T}$, $m = 1, \ldots, M$ (see equations 9b and 12b of Ghahramani & Jordan, 1997). In addition, both procedures ignore correlations among state variables (similar to mean-field methods). It is difficult, however, to make concrete statements about the similarities and differences of the procedures because the fixed-point equations for the variational approximation involve auxiliary parameters of an approximating distribution that are not part of the original factorial hidden Markov model.

4 Simulation Results

This section reports simulation results using three artificial data sets; the observations in the first data set are real valued, and the observations in the remaining datasets are multinomial. When attempting to predict how well the generalized backfitting algorithm will work on these data sets, it is important to keep in mind that although the algorithm is new, its constituent parts are not. Methods such as the IRLS procedure, the backfitting procedure, and the Baum-Welch procedure have all been successfully used in scores of applications over a period of many years. Consequently, we predict that FHMMs trained with the generalized backfitting algorithm should show good performance.

4.1 Real-Valued Data Sets. When studying real-valued data sets, several algorithms exist for estimating the parameter values of an FHMM. Ghahramani and Jordan (1997) found that an EM algorithm worked best and that an approximate EM algorithm in which Gibbs sampling is used during the E-step and an approximate algorithm based on variational techniques each worked nearly as well.

We conducted a set of simulations based on a related set of simulations conducted by Ghahramani and Jordan (1997). Training and test data were real valued and generated by an FHMM. We compared the training and test

set log-likelihoods of five models:

- *HMM*: An HMM trained using the Baum-Welch algorithm
- *Exact*: An FHMM trained using an exact EM algorithm
- *SVA*: An FHMM trained using the structured variational approximation of Ghahramani and Jordan
- *GBF-γ* : An FHMM trained using the GBF-γ algorithm
- *GBF-V*: An FHMM trained using the GBF-V algorithm

The FHMM that generated the data had M hidden Markov chains, each containing K components. All of its parameter values, except for the covariance matrices associated with the hidden Markov chains and the output covariance matrix, were sampled from a uniform $[0, 1]$ distribution and, if needed, appropriately normalized to satisfy the sum-to-one constraint of probability distributions. The covariance matrices were set to a multiple of the identity matrix, $C = 0.01I$. The training and test sets consisted of 20 sequences of length 20, where the observable vector at each time step was a four-dimensional vector. For each randomly sampled set of parameters, a training and test set were generated, and each model was run once. Fifteen sets of parameters, and thus 15 training and test sets, were generated for each of four problem sizes: $\{M = 3, K = 2\}$, $\{M = 3, K = 3\}$, $\{M = 5, K = 2\}$, and $\{M = 5, K = 3\}$.

The FHMMs trained to summarize the data consisted of M hidden Markov chains with K states each, whereas the HMMs had K^M states. Their parameter values were randomly initialized, except for the covariance matrices, which were set and fixed to $0.01I$. Learning algorithms were run for 100 iterations (in the case of FHMMs trained with either of the generalized backfitting algorithms, a cycle is defined as five iterations of the Baum-Welch algorithm for each hidden Markov chain, and the algorithm was run for 20 cycles). At the end of training, the log-likelihoods on the training and test sets were computed for all models using the exact algorithm.

The mean ($\pm$ standard error of the mean) log-likelihood of each model on each of the four problem sizes is shown in Table 1. Also included in this table is the log-likelihood of the training and test sets under the true model that generated the data. Taken as a whole, the simulation results are highly compatible with the results reported by Ghahramani and Jordan.

Model *HMM* tended to suffer from overfitting; its test set log-likelihood is significantly worse than its training set log-likelihood. This is evident for the smallest problem size and is extreme for the largest problem size. Among the models using FHMMs, model *Exact* tended to perform best. This result is unsurprising because it performs exact inference and maximum likelihood parameter estimation. Among the approximate models, models *SVA*, *GBF-γ* , and *GBF-V* showed similar levels of performance.

Table 1: Mean (± Standard Error of the Mean) Log-Likelihood of Each Model on the Four Problem Sizes for the Real-Valued Data Set.

M	K	Model	Training Set	Test Set
3	2	*True*	806.9 ± 22.6	797.2 ± 24.1
		HMM	544.0 ± 107.2	286.4 ± 151.3
		Exact	630.2 ± 103.0	592.8 ± 103.9
		SVA	604.3 ± 94.0	574.2 ± 93.5
		GBF-γ	644.1 ± 90.5	596.5 ± 91.5
		GBF-V	625.1 ± 90.3	582.8 ± 91.0
3	3	*True*	318.9 ± 14.0	319.0 ± 20.0
		HMM	356.4 ± 34.3	−1170.8 ± 110.7
		Exact	−73.5 ± 83.4	−175.0 ± 97.1
		SVA	−421.0 ± 78.5	−525.6 ± 92.1
		GBF-γ	−373.0 ± 57.4	−518.5 ± 69.9
		GBF-V	−424.9 ± 56.4	−555.5 ± 75.9
5	2	*True*	332.7 ± 30.8	336.8 ± 31.7
		HMM	172.0 ± 120.7	−1861.9 ± 233.8
		Exact	218.3 ± 73.4	148.0 ± 91.0
		SVA	−447.5 ± 166.1	−519.1 ± 174.5
		GBF-γ	−666.6 ± 206.1	−756.8 ± 208.2
		GBF-V	−722.6 ± 180.2	−812.1 ± 176.9
5	3	*True*	−335.9 ± 24.8	−327.7 ± 28.2
		HMM	205.9 ± 104.7	−9262.4 ± 606.4
		Exact	−973.5 ± 95.3	−1267.0 ± 120.9
		SVA	−1425.7 ± 115.8	−1656.0 ± 138.5
		GBF-γ	−1358.4 ± 122.5	−1554.9 ± 128.9
		GBF-V	−1407.2 ± 125.4	−1595.1 ± 135.0

4.2 Multinomial Data Sets. The variational techniques of Ghahramani and Jordan (1997) are specifically designed for real-valued time series and do not extend in an obvious way to other types of data. These investigators offered some speculative suggestions as to how their techniques might be extended, but ultimately left this as an open research question. In contrast, the generalized backfitting algorithm can be regarded as general purpose, because it is easily applied to any sequential data set in which the distribution of the response variable is a member of the exponential family of distributions. We demonstrate this by considering multinomial time series.

Training and test data were generated by an FHMM as described above. We compared the training and test set log-likelihoods of two models: *HMM*, an HMM trained using the Baum-Welch algorithm, and *GBF-γ* , an FHMM trained using the GBF-γ algorithm. The FHMM that generated the data had M hidden Markov chains, each containing K components. All of its parameter values were sampled from a uniform [0, 1] distribution and, if needed, appropriately normalized to satisfy the sum-to-one constraint of probability distributions. The training and test sets consisted of 20 sequences of length 20. For the first multinomial data set, the response variable could

take one of four possible values at each time step; that is, the response variable was a four-dimensional vector with one component equal to one and the remaining three components set to zero. The response variable could take one of eight possible values at each time step in the second multinomial data set. For each randomly sampled set of parameters, a training and test set were generated, and each model was run once. Fifteen sets of parameters, and thus 15 training and test sets, were generated for each of four problem sizes: $\{M = 3, K = 2\}$, $\{M = 3, K = 3\}$, $\{M = 5, K = 2\}$, and $\{M = 5, K = 3\}$.

Model *HMM* contained K^M states, whereas model *GBF-*γ was an FHMM with M hidden Markov chains with K states each. Their parameter values were randomly initialized, except for the covariance matrices of model *GBF-*γ, which were set and fixed to the identity matrix. In the case of model *HMM*, the Baum-Welch algorithm was run for 100 iterations. Model *GBF-*γ was run for 100 cycles, where each cycle consisted of one iteration of the Baum-Welch algorithm for each hidden Markov chain. At the end of training, the exact log-likelihoods on the training and test sets were computed for all models.

The mean ($\pm$ standard error of the mean) log-likelihood of each model on each of the four problem sizes is shown in Table 2 for the data set with an alphabet size of four and in Table 3 for the data set with an alphabet size of eight. Also included in these tables are the log-likelihoods of the training and test sets under the true model that generated the data. On the data set with an alphabet size of four, models *HMM* and *GBF-*γ showed similar levels of performance on the test sets. In contrast, on the data set with an alphabet size of eight, model *HMM* tended to overfit the training data, and thus model *GBF-*γ achieved a significantly better level of performance on the test sets. This result suggests that FHMMs scale better than HMMs as the alphabet size of the data set increases.

Table 2: Mean ($\pm$ Standard Error of the Mean) Log-Likelihood of Each Model on the Four Problem Sizes for the Multinomial Data Set whose Alphabet Size Was Four.

M	K	Model	Training Set	Test Set
3	2	*True*	-531.4 ± 4.6	-530.1 ± 4.1
		HMM	-510.9 ± 5.1	-548.2 ± 3.3
		GBF-γ	-547.2 ± 8.4	-554.8 ± 11.1
3	3	*True*	-543.4 ± 2.7	-544.6 ± 2.4
		HMM	-526.5 ± 3.6	-558.3 ± 3.8
		GBF-γ	-548.9 ± 2.4	-561.9 ± 3.4
5	2	*True*	-528.7 ± 6.0	-529.0 ± 5.7
		HMM	-497.3 ± 10.6	-566.2 ± 13.7
		GBF-γ	-534.1 ± 6.7	-543.3 ± 6.5
5	3	*True*	-534.9 ± 6.3	-531.1 ± 7.5
		HMM	-514.9 ± 11.5	-554.9 ± 15.5
		GBF-γ	-542.8 ± 4.5	-549.1 ± 6.6

Table 3: Mean (± Standard Error of the Mean) Log-Likelihood of Each Model on the Four Problem Sizes for the Multinomial Data Set Whose Alphabet Size Was Eight.

M	K	Model	Training Set	Test Set
3	2	*True*	−805.9 ± 5.2	−809.0 ± 3.2
		HMM	−740.8 ± 5.1	−925.0 ± 9.7
		GBF-γ	−880.0 ± 10.9	−880.6 ± 8.6
3	3	*True*	−817.5 ± 2.0	−818.3 ± 2.1
		HMM	−612.6 ± 2.3	−1655.2 ± 43.9
		GBF-γ	−842.8 ± 7.5	−867.4 ± 9.8
5	2	*True*	−796.4 ± 5.4	−799.0 ± 4.9
		HMM	−566.8 ± 5.2	−1803.7 ± 40.9
		GBF-γ	−845.0 ± 16.4	−865.7 ± 21.2
5	3	*True*	−799.5 ± 4.2	−796.9 ± 4.6
		HMM	−310.4 ± 26.9	−2972.0 ± 174.2
		GBF-γ	−805.7 ± 3.3	−821.1 ± 5.7

Considering all simulations results as a whole, we conclude, as do Ghahramani and Jordan (1997), that FHMMs represent a potentially important advance in the modeling of sequential data. Due to their use of distributed representations, FHMMs with a relatively small number of parameters can have a large representational capacity. Consequently, FHMMs can often learn structure in data sets that conventional HMMs cannot. In addition, we conclude that the generalized backfitting algorithm is an effective method for training FHMMs that is applicable to a wide variety of data types. As emphasized above, the constituent parts of the algorithm are familiar techniques in the neural computation, machine learning, and statistics communities, and so the efficacy of the algorithm is unsurprising.

5 Summary and Conclusion

Previous researchers developed new learning architectures for sequential data by extending conventional hidden Markov models through the use of distributed state representations (Williams & Hinton, 1991; Ghahramani & Jordan, 1997). An advantage of these architectures is that they are efficient from a representational viewpoint. A disadvantage is that exact inference and parameter estimation in these architectures is computationally intractable. Ghahramani and Jordan (1997) showed, however, that approximate inference and parameter estimation in one such architecture, FHMMs, is feasible in certain circumstances. Unfortunately, the learning algorithm proposed by these investigators, based on variational techniques, is difficult to understand and implement and is limited to the study of real-valued data sets.

This article has proposed an alternative method for approximate inference and parameter estimation in FHMMs based on the perspective that FHMMs are a generalization of a well-known class of statistical models known as GAMs (Hastie & Tibshirani, 1990). Using existing statistical techniques for GAMs as a guide, we have developed the generalized backfitting algorithm. This algorithm computes customized error signals for each hidden Markov chain of an FHMM and then trains each chain one at a time using conventional techniques from the hidden Markov models literature. Relative to previous perspectives on FHMMs, we believe that the viewpoint taken here has a number of advantages. First, it places FHMMs on firm statistical foundations by relating FHMMs to a class of models well studied in the statistics community, yet it generalizes this class of models in an interesting way. Second, it leads to an understanding of how FHMMs can be applied to many different types of time-series data, including Bernoulli and multinomial data, not just real-valued data. Finally, it leads to an effective learning procedure for FHMMs that is easier to understand and easier to implement than existing learning procedures. Simulation results suggest that FHMMs trained with the generalized backfitting algorithm are a practical and powerful tool for analyzing sequential data.

Although the study of sequential data has always been important, it has recently received renewed attention due to interest from relatively new scientific fields such as computational finance and bioinformatics. These fields make extensive use of conventional hidden Markov models due to their ease of use and attractive computational properties. We conjecture, however, that many problems that are currently approached through the use of HMMs may be better studied through the use of FHMMs trained with the generalized backfitting algorithm. The application of FHMMs and the generalized backfitting algorithm to real-world problems in the area of bioinformatics is our current focus of research.

Acknowledgments

We are grateful to Z. Ghahramani for making software available on his web site that was useful in this project and to the anonymous reviewers for their helpful comments on this manuscript. This work was supported by NIH research grant R01-EY13149 and NSF grant DMS-0102636.

References

Baum, L., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, *41*, 164–171.

Boyen, X., & Koller, D. (1999). Approximate learning of dynamic models. In M. S. Kearns, S. A. Solla, & D. A. Cohn (Eds.), *Advances in neural information processing systems, 11*. Cambridge, MA: MIT Press.

Dean, T., & Kanazawa, K. (1988). Probabilistic temporal reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*. St. Paul, MN: AAAI Press.

Ghahramani, Z., & Jordan, M. I. (1997). Factorial hidden Markov models. *Machine Learning, 29*, 245–273.

Hastie, T. J., & Tibshirani, R. J. (1990). *Generalized additive models*. London: Chapman and Hall.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. (1998). An introduction to variational methods for graphical models. In M. I. Jordan (Ed.), *Learning in graphical models*. Cambridge, MA: MIT Press.

McCullagh, P., & Nelder, J. (1989). *Generalized linear models*. London: Chapman and Hall.

Murphy, K., & Weiss, Y. (2001). The factored frontier algorithm for approximate inference in DBNs. In *Proceedings of the Sevententh Conference on Uncertainty in Artificial Intelligence*. San Francisco: Morgan Kaufmann.

Rabiner, L., & Juang, B.-H. (1993). *Fundamentals of speech recognition*. Englewood Cliffs, NJ: Prentice Hall.

Williams, C. K. I., & Hinton, G. E. (1991). Mean field networks that learn to discriminate temporally distorted strings. In D. Touretzky, J. Elman, & G. Hinton (Eds.), *Connectionist models: Proceedings of the 1990 Summer School*. San Francisco: Morgan Kaufmann.

Received October 2, 2001; accepted April 17, 2002.