Scientific Programming 10 (2002) 201–219
IOS Press

A Bayesian approach for structural learning with hidden Markov models

Cen Li[a] and Gautam Biswas[b]
[a]*Department of Computer Science, Middle Tennessee State University, Box 48, Murfreesboro, TN 37132, Tel.: +1 615 904 8168; Fax: +1 615 898 5567; E-mail: cli@mtsu.edu*
[b]*Department of Electrical and Computer Engineering, Vanderbilt University, Box 1679 Station B, Nashville, TN 37235, USA*
Tel.: +1 615 343 6204; Fax: +1 615 343 5459; E-mail: biswas@vuse.vanderbilt.edu

Abstract. Hidden Markov Models(HMM) have proved to be a successful modeling paradigm for dynamic and spatial processes in many domains, such as speech recognition, genomics, and general sequence alignment. Typically, in these applications, the model structures are predefined by domain experts. Therefore, the HMM learning problem focuses on the learning of the parameter values of the model to fit the given data sequences. However, when one considers other domains, such as, economics and physiology, model structure capturing the system dynamic behavior is not available. In order to successfully apply the HMM methodology in these domains, it is important that a mechanism is available for automatically deriving the model structure from the data. This paper presents a HMM learning procedure that simultaneously learns the model structure and the maximum likelihood parameter values of a HMM from data. The HMM model structures are derived based on the Bayesian model selection methodology. In addition, we introduce a new initialization procedure for HMM parameter value estimation based on the K-means clustering method. Experimental results with artificially generated data show the effectiveness of the approach.

1. Introduction

The Hidden Markov Model (HMM) methodology has been applied extensively in many domains for efficient modeling of multi-dimensional sequence data. Examples of successful applications include the speech recognition domain for temporal sequence modeling [1], and the genomics domain for spatial sequence modeling [2]. The HMM methodology is a probabilistic state based approach, where the set of system states that govern the dynamic behavior or spatial characteristics may not be directly observable, thus the term "hidden", the hidden states manifest indirectly as multi-dimensional output sequences that are directly observable. The HMM learning task starts with sequence data, and attempts to fit a probabilistic state transition model that best describes the data.

HMM methodology can be applied to modeling tasks for any real world system that is dynamic in nature, for example, systems in economics, social science, and medical domains. With the proliferation of computing technology, there is a lot of interest in analyzing these dynamic systems in terms of better understanding the system behaviors and/or performing prediction or forecasting tasks. Typically, data describing the systems is recorded over different periods of time, and is in the form of multiple sequences of data, each describing one particular aspect (or feature) of the system. To illustrate the model learning task, let us consider hypothetical data generated from a dynamic process. We measure two time-varying features of the system, f_1 and f_2, over a fixed time interval (100 time units in our example), and collected values for f_1 and f_2 at fixed increments, say every 0.5 sec. We repeat the data collection process a number of times, therefore, our data is made up of a set of f_1 and f_2 feature value sequences and each sequence is 100 time units long. Figure 1 shows data recorded at two different time periods from the same, hypothetical dynamic system.

Our goal is to construct a model describing the dynamic behavior of the underlying system by constructing HMMs that best describe the data. Typically, in

ISSN 1058-9244/02/$8.00

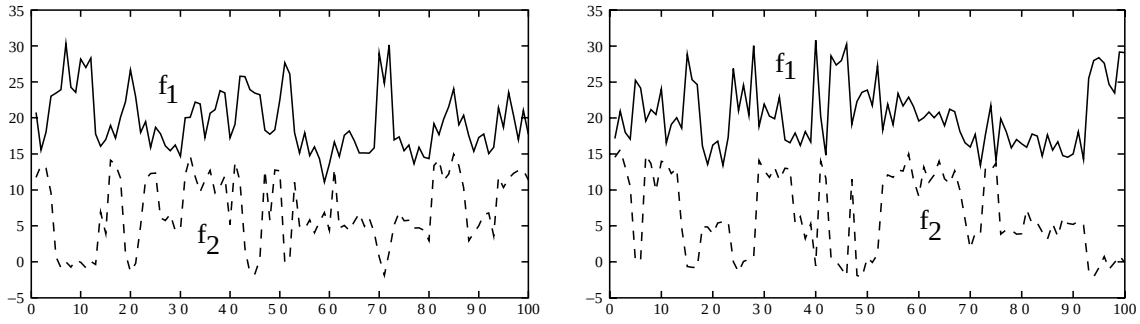

Fig. 1. Data recorded at two different time periods from the same, hypothetical dynamic system.

applications such as speech recognition and genomics research, the structures of the HMMs, i.e., the number of states in the model and the links between states, are predefined by the domain experts. Therefore, the HMM learning problem focuses on the learning of the parameter values of the HMM from data [1,3]. But, for many other domains, not as much is known about the domain structure, therefore, it becomes more difficult to define HMM structure beforehand. This significantly limits the application of the current HMM learning methodologies.

In our work, we extend existing methodologies, by proposing an extended HMM learning procedure that simultaneously derives both the structure and the parameters of the HMM from sequence data of the type shown in Fig. 1. To elaborate, we first discuss the HMM structure.

More formally, a HMM is a non-deterministic stochastic Finite State Automata (FSA) that satisfies the first order Markov property. For a stochastic process, X, let $X(t)$, denotes the state of a system at time t. X is said to satisfy the Markov property if, for any sequence $X(t_0), X(t_1), \ldots, X(t_n)$, such that $t_0 < t_1 < \ldots t_{n-1} < t_n$, we have

$$\begin{aligned} \mathrm{P}\{X(t_n) = x | X(t_0) = x_0, \ldots, X(t_{n-1}) = x_{n-1}\} \\ = \mathrm{P}\{X(t_n) = x | X(t_{n-1}) = x_{n-1}\}. \end{aligned}$$

The state in which the system finds itself at time t_n depends only on its state at time t_{n-1}, $X(t_{n-1})$, which contains all the relevant information concerning the history of the process behavior [4]. Most discrete-event models of physical systems are assumed to satisfy this Markov property, i.e., their state description at the current time t, is a function of their last state description at time $t-1$, and the input to the system at time t.

A first order HMM, λ, with M states, $S = (S_1, S_2, \ldots, S_M)$, that models data described by K temporal features can be characterized by three sets of probabilities: $\vec{\pi}, A$, and $B(\vec{\mu}, \Sigma)$, where

- the *initial state probabilities*, $\vec{\pi}$, a vector of size M, $(0 \leqslant \pi_i \leqslant 1)$, defines the probability of any of the states, S_i, being the initial state of a given temporal sequence;
- the *transition matrix*, A of size $M \times M$, $(0 \leqslant a_{ij} \leqslant 1, 1 \leqslant i, j \leqslant M)$, defines the probability of transition from state i at time t, to state j at the next time step; and
- the *emission probability matrix*, B of size $M \times K$, defines the probability of generating feature values at any given state [3]. When the system behavior can be represented as a sequence of discrete states, a discrete density HMM (DDHMM) may be employed, where the probability density function (*pdf*) associated with the individual states follow multinomial distributions. When the system behavior is sampled from continuous real-valued distributions, a continuous density HMM (CDHMM) is employed, where the *pdf* associated with the individual states follow multivariate normal distributions. For CDHMM, the multivariate Gaussian distributions associated with individual states are characterized by their mean vector, $B_{\vec{\mu}}$, and co-variance matrix, B_{Σ}. In our work, temporal data are assumed to be continuous valued and CDHMMs are used for modeling these data. In addition, we assume that all temporal features are independent of each other. Therefore, the covariance matrix at each state is diagonal, and can be represented as a variance vector, $B_{\vec{\sigma}}$.

Looking back at our example presented in Fig. 1, the multi-dimensional data sequence has to be broken down into segments, where each segment corresponds to a hidden state of the system. It is assumed that the

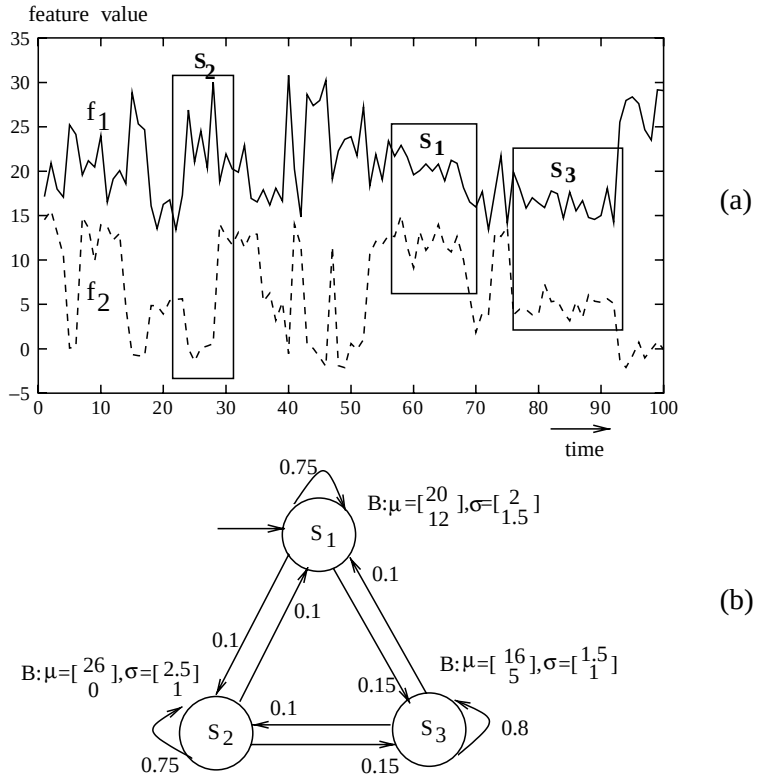

Fig. 2. The mapping between the states in the derived HMM and the data collected from the dynamic system.

overall dynamic behavior of the system is governed by a finite set of states: $S_1, S_2, \ldots, S_K$. The behavior of the system in each state, i.e., the output produced, is a function of the emission probability matrix B, and the overall behavior of the system is a combination of the state transition probability matrix, A, and the emission matrix, B. Let us say that for our example data shown in Fig. 1, our learning algorithm correctly determines that the underlying process is made up of three states, and derives the transition probabilities among the states, as well as the emission probabilities for each state. The resulting model can be depicted as a stochastic automata shown in Fig. 2(b). Rectangles in the sequence data in Fig. 2(a) illustrate the temporal data output from each of the three hidden states.

In this paper, we cast the HMM learning problem in the Bayesian model selection framework [5]. Bayesian model selection enforces the Occam's razor principle. When constructing a HMM for data, Bayesian model selection employs a criterion function that trades off the fit between the model and the data, and the complexity of the model itself. The reasoning behind this is that if one allows the complexity of the model to increase arbitrarily, one can always derive models that almost perfectly fit the data. Such a model typically overfits the data and has no predictive value in practical applications. Therefore, it makes sense to tradeoff complexity and accuracy. It introduces a penalty term that is directly proportional to the model complexity function. The trade off between this term and the accuracy of fit results in the derivation of HMM models that are good fit for the data, but is not too complex in structure.

The rest of this paper discusses the Bayesian HMM learning methodology, and presents experimental results to support its effectiveness. Section 2 reviews the HMM modeling methodology. Section 3 introduces the Bayesian model selection methodology and presents the Bayesian HMM structural learning procedure. Section 4 discusses the design of the experiments, and the experimental results obtained. Section 5 summaries the paper and discusses directions for future work.

2. The HMM modeling methodology

2.1. Learning HMM from data

The HMM learning problem can be viewed at two different levels: (i) the parameter learning level, and (ii)

the structural learning level. At the parameter learning level, the model structure, i.e., the number of states in a HMM, is given, the HMM learning process estimates the model parameter values, (i.e., the π vector, and the A and B matrices), from data using iterative optimization methods. The degree of fit of the derived models is dependent on the initial values of the parameters and the parameter estimation methods used. In addition, it is a function of the model size. When the given model size differs significantly from the true model size, no matter how good the parameter estimation procedure is, the derived models are not likely to provide a good characterization for the data. At the structural learning level, the HMM learning procedure learns the model structure and the parameter values of the model from the data simultaneously. Next, we discuss existing methods developed for the two HMM learning levels.

2.2. Learning HMM parameters from data

Given the structure, i.e., the number of states in the HMM, parameter estimation methods try to derive the model parameters, $\vec{\pi}$, A, and B, such that the parameter configuration optimizes a predefined objective criterion. Criteria that have been used for HMM parameter estimation include data likelihood, mutual information measured from multiple HMMs [6], entropy based distance functions [7], and maximum between HMM model distance [8]. We employ the well known *Baum-Welch* (BW) [9] parameter estimation method, which uses the data likelihood maximization criterion to measure the fitness of a model parameter configuration and to guide the search process.

The *Baum-Welch* parameter estimation procedure is a variation of the more general EM algorithm [10]. It iterates between an expectation step (E-step) and a maximization step (M-step). The E-step assumes the current parameter configuration of the model and computes the expected values of the necessary statistics. The M-step uses these statistics to update the model parameters so as to maximize the expected likelihood of the parameters [11]. The iterative process continues until the parameter configuration converges.[1] The maximum likelihood HMM parameter estimation is implemented using the *Forward-Backward* dynamic programming procedure. Details of the *Forward-Backward* procedure can be found in [3].

[1] In experiments we have run, we consider a parameter configuration to converge if the log data likelihood of two consecutive model configurations differ by less than 10^{-6}.

Like other hill climbing approaches, a problem with the *Baum-Welch* procedure is that it is sensitive to the initial configuration of the model, and tends to converge to local maxima parameter configurations. The only way to solve this problem is exhaustive search, which is computationally infeasible in many applications. The *Viterbi* algorithm presents a compromise. It is designed to start with an initial HMM, λ, with a random parameter configuration. Given a sequence data $O = O_1, O_2, \ldots, O_L$, the algorithm finds the single best state sequence, called the *Viterbi* path Q, by segmenting the sequence values into states that maximize $P(O, Q|\lambda)$. The state membership, $j, 1 \leqslant j \leqslant M$, of the t^{th} value, $O_t, 1 \leqslant t \leqslant L$, is determined based on the likelihood of the partial observation sequence $O_0, O_1, \ldots O_{t-1}$, along the partial *Viterbi* path, δ_{t-1}, as well as the probability of the t^{th} observation value, given the current parameter configuration of the model. Once sequence values are segmented into states along the *Viterbi* path, model parameters can be updated based on the statistics collected using the segmented data.

Viterbi method is computationally efficient. The time complexity of the algorithm is $O(ML)$. The method suffers from two major problems: (i) the quality of the initialization still depends on an initial random parameter configuration of the model, and (ii) the data segmentation is done through a local search. As a result, the initialization algorithm does not always guarantee a good starting parameter configuration for the *Baum Welch* procedure for convergence to the globally maximum configuration.

2.3. Learning HMM model size/structure from data

An important issue for developing general HMM-based learning systems is to solve the model size derivation problem. When the same set of data is modeled using HMMs of different sizes, significantly different models may be generated. HMMs of different sizes model data at different levels of abstraction. HMMs with a larger number of states typically provide a more detailed description for data than HMMs with a lesser states. On the other hand, as the size of the HMM grows, the model becomes more and more complex, which often overfits data and loses prediction value. In addition, a model that is too complex makes the model interpretation task more difficult. The question is how to determine the most appropriate HMM model size for any given data set.

The HMM model selection methods can be categorized into two groups depending on whether a method is based on: (i) a model reduction strategy, or (ii) a model expansion strategy. Model reduction strategy starts with a HMM that has a large number of states and successively reduces the size of the model by merging and eliminating states. Solcke and Omohundro's state merging algorithm [12] and Casacuberta et al.'s state elimination method [13], and Brand's entropic prior approach [14] are examples of this strategy.

Solcke and Omohundro's [12] algorithm starts with an initial model of size $S_0 = NL$, where N is the size of the data and L is the length of the temporal sequences. The model contains one state for every value observed in every sequence. The initial model configuration corresponds to the maximum likelihood model for the data. The size of the model is gradually reduced by merging pairwise states of the model. The merging process terminates when further state reduction does not improve the posterior probability of the model given the data. The pairwise state merging process makes this method computationally expensive. At each merge step, all $S \cdot (S-1)/2$ (S is the size of the model, initially $S = S_0$) pairs of states are considered for merging. Each pair of states is *tentatively* merged, and the quality of the model, before and after the merge, is compared. The merge that results in the largest improvement in model posterior probability is selected to be carried out formally. When data is described with continuous valued temporal sequences of long length, the computation involved in each state reduction step can be very significant. Another problem with this method is that, once it is committed to a merge step, the method cannot "back off" from a decision which, in the light of new data, may turn out to be an overgeneralization.

Casacuberta et al. [13] proposed deriving the structure of HMMs through combined error correcting grammatical inference (ECGI) and state reduction techniques. First, ECGI is used to learn a HMM when sequence data are provided one at a time. For every new sequence which cannot be exactly recognized by the current model, the model is updated by adding states and transitions which are required for the new sequence to be accepted. Models constructed using ECGI are typically much smaller than the maximum likelihood models initially derived using Solcke and Omohundro's algorithm, but they can still be rather large. To overcome this problem, a model reduction procedure is applied to eliminate states that are less important. The importance of the states is determined by the number of times the state is used for the parsing of the sequence data across the set of states in the model. Less important states are eliminated from the model. This model reduction approach is more efficient than Stocle and Omohundro's state merging approach. But with this method, the state definitions of the model are not updated after each state elimination step. In comparison, this is not as good as the state merging approach where a few of the less significant states may be combined to form one important state and remain in the model.

In Brand [14]'s approach, HMM structure learning process is folded into the conventional EM HMM parameter estimation procedure. The M-step of the EM process is replaced with the entropic MAP estimator. Iterative MAP estimation drives weakly supported transition and state parameters toward extinction. A subsequent step trims the transition and states, with a guarantee that any such deletion will increase the posterior probability of the model.

Model expansion methods include Takami and Sagayama's [15] SSS algorithm and Ostendorf and Singer's [16] extended SSS algorithm. They start with a HMM of the minimum size (i.e., one state), and increase the size of the model in steps by successively splitting individual states based on a chosen criterion. Takami and Sagayama's [15] successive state splitting (SSS) algorithm constructs the Hidden Markov network (HMnet), a special case of a HMM, from data. Their method starts with a minimum size HMnet containing a single state. Then the model is gradually expanded by successively splitting states that have the largest variance in their state emission *pdf*. The actual split of the state can be *contextual*, i.e., concatenation of two states in parallel, or *temporal*, i.e., concatenation of two states in series. The emission *pdf*s of the two states are re-estimated. The parameter re-estimation is done through a local search procedure, i.e., only parameters that are directly connected to the splitting state will be updated. The state splitting procedure continues until a prescribed number of total states is reached. The parameter values of the HMnet are re-estimated once the model has expanded to its pre-determined size. Because the size of the final model is fixed, the SSS algorithm's focus is on the topology of the model, i.e., how states are connected in the model.

Ostendorf and Singer [16] further expanded the basic SSS algorithm by choosing the state and the candidate split at the same time based on the likelihood gains. Their state splitting method relies on a data likelihood measure to make decisions on state splitting choices. The recognition accuracy of the model on test data is

used as the stopping criteria for the splitting process. The main limitation of the two SSS algorithms is that they perform local parameter updating after each split operation. Local parameter updating methods are more efficient than the global parameter updating methods, for example, the *Baum Welch* procedure, but they are shown to produce less fit parameter values for data [16]. The model parameters become even lesser fit for data after a sequence of splitting operations.

The goal of our HMM learning methodology is to derive discrete event models from data that represent the behavior of dynamic systems. For model interpretation purposes, it is preferable that the models are compact, i.e., involve lesser states. Also, for most well behaved systems, the number of states go through by such systems is usually quite small. This means that the size of the HMM that captures the temporal behavior of a dynamic process is typically much smaller than the length of the data sequence. Furthermore, considering the nature of the data is multi-dimensional, continuous valued sequences, a model reduction approach such as the one proposed by Solcke and Omohundro is not computationally feasible in many applications. Therefore, our HMM learning method adopts the model expansion strategy that expands from a minimal size model to derive a relatively small size model.

In addition to the strategy for model construction, an objective criterion function is needed to determine when an expansion process terminates and the best model to be selected for the data. We employ the Bayesian model selection methodology for this task. Bayesian model selection criterion trades off the fitness of model to the complexity of the model. The best model selected using the Bayesian model selection criterion is the one that is compact in size and is adequately describes the data.

3. A Bayesian approach for learning HMM from data

Recently, Bayesian methods have received much attention in developing model learning techniques [17–19]. In this section, we first review the Bayesian model selection methodology. Then we discuss how it is adapted to solve the HMM learning problem.

3.1. Bayesian model selection

Given data X, and a model, M, derived from X, Bayes theorem defines the posterior probability of the model, $P(M|X)$, as:

$$P(M|X) = \frac{P(X|M)P(M)}{P(X)}, \tag{1}$$

where $P(X)$ and $P(M)$ are prior probabilities of the data and the model, respectively, and $P(X|M)$ is the marginal likelihood of the data. The prior probability of the data is unchanged across different models. Therefore, for model comparison purposes, we can express $P(M|X) \propto P(X|M)P(M)$. Assuming all models are equally likely a priori, we can reduce the posterior probability to depend on the marginal likelihood only, i.e., $P(M|X) \propto P(X|M)$. Using this formalization, we define the best model as the one that gives the highest marginal likelihood.

Computing the marginal likelihood for complex models has been an active research area [20–23]. Given the parameter configuration, θ, of a model, M, the marginal likelihood of the data is computed as

$$P(X|M) = \int_{\theta} P(X|\theta, M)P(\theta|M)d\theta. \tag{2}$$

A class of approximation techniques, Monte-Carlo methods, have received a great deal of attention in the statistics community [20,24]. Although, in theory, these techniques are known to converge to produce accurate results, they are computationally too expensive to be of practical use on non-trivial size data sets. An alternative class of approximation techniques is based on the large sample properties of probability distributions. They are computationally much more efficient than Monte-Carlo techniques, and provide accurate results when certain assumptions hold. In this section, we look at three marginal likelihood approximation methods: (i) the Laplace approximation [22], (ii) the Bayesian Information Criterion (BIC) [22,23,25], and (iii) the Cheeseman-Stutz approximation [26].

The Laplace approximation

The Laplace approximation is widely used by Bayesian statisticians. For large amounts of data, $P(\theta|X, M) \propto P(X|\theta, M) \cdot P(\theta|M)$ can often be approximated as a multivariate Gaussian distribution [22]. In particular, define

$$g(\theta) \equiv \log(P(X|\theta, M) \cdot P(\theta|M)). \tag{3}$$

Let $\tilde{\theta}$ to be the parameter configuration that maximizes $g(\theta)$.

This parameter configuration also maximizes $P(\theta|X, M)$, and is known as the *Maximum a Posterior*(*MAP*) configuration for θ. A second degree Taylor polynomial approximation for $g(\theta)$ at $\tilde{\theta}$ yields:

$$g(\theta) \approx g(\tilde{\theta}) - \frac{1}{2}(\theta - \tilde{\theta})A(\theta - \tilde{\theta})^t, \quad (4)$$

where $(\theta - \tilde{\theta})^t$ is the transpose of row vector $(\theta - \tilde{\theta})$, and A is the negative Hessian of $g(\theta)$ evaluated at $\tilde{\theta}$. Computing $e^{g(\theta)}$ and using equation 3, we obtain

$$\begin{aligned} & P(\theta|X, M) \\ & \propto P(X|\theta, M)P(\theta|M) \\ & \approx P(X|\tilde{\theta}, M)P(\tilde{\theta}|M)e^{\{-\frac{1}{2}(\theta-\tilde{\theta})A(\theta-\tilde{\theta})^t\}} \end{aligned} \quad (5)$$

Integrating both sides of Eq. 5 over θ, and taking the logarithm of the result, we obtain the Laplace approximation of the marginal likelihood:

$$\begin{aligned} & \log P(X|M) \\ & \approx \log P(X|\tilde{\theta}, M) + \log P(\tilde{\theta}|M) \\ & \quad + \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|A|, \end{aligned} \quad (6)$$

where d is the dimension of $g(\theta)$. It has been shown that under certain regularity conditions, the Laplace approximation is quite accurate, but its computation is expensive, in particular the computation of the negative Hessian [22].

Bayesian Information Criterion (*BIC*)

BIC [25], is derived from the Laplace approximation [22]. By retaining only the terms in equation 7 whose computational complexity increases with the size of data, N:

1. $\log P(X|\tilde{\theta}, M)$, which increases linearly with N, and
2. $log|A|$, which increases as $d \cdot logN$,

we obtain the very efficient but less accurate BIC approximation. Also, for large N, $\tilde{\theta}$ can be approximated by the ML parameter configuration, $\hat{\theta}$. Thus, we have:

$$\log P(M|X) = \log P(X|M, \hat{\theta}) - \frac{d}{2}\log N. \quad (7)$$

The first term in the BIC computation, $\log P(X|M, \hat{\theta})$, is the likelihood term which tends to favor larger and more detailed models of data. The second term, $-\frac{d}{2}logN$, is the model complexity penalty term. d represents the number of significant parameters in the model. The larger and more complex the model, the larger the d value, thus a higher model complexity penalty. This term bias the selection of simpler models with a smaller number of parameters. BIC selects the best model for the data by trading off these two terms. By using the model complexity term, the BIC measure automatically enforces the Occam's principle for favoring the simplest model which explains the observations. BIC is considered to be equivalent to Rissanen's Minimum Description Length (MDL) measure [27] derived from information theory. They both find models that are a good fit to data, and they both favor smaller models. BIC uses the model complexity penalty term to penalize large models that require a larger number of parameters to characterize the model. MDL selects the best model that can be encoded using the minimum number of bits in machine code. Larger models contain more information to be encoded, i.e., more parameters and more structure information [28]. Thus more bits are required to encode the models, and, therefore, are less favored.

Cheeseman-Stutz(*CS*) *Approximation*

Cheeseman and Stutz first proposed the CS approximation method for marginal likelihood computation in their Bayesian clustering system, AUTOCLASS [26] by defining:

$$P(X|M) = P(X'|M)\frac{P(X|M)}{P(X'|M)}, \quad (8)$$

where X' represents the data with known cluster labels (known data), and X represents data that is not labeled with cluster information (incomplete data). The first term is the marginal likelihood of the known data. An exact computation of this term involves integration through all possible parameter configurations of the model as in Eq. 2. The integration is approximated by a summation over a set of local maximum parameter configurations, θ_s [26,29]:

$$\begin{aligned} P(X'|M) &= \int P(\theta|M)P(X'|\theta, M)d\theta \\ &\approx \sum_{\theta\in\theta_s} P(\theta|M)P(X'|\theta, M). \end{aligned} \quad (9)$$

To reduce computation, in our work, we have further extended this approximation by using a single maximum likelihood configuration, $\hat{\theta}$, $\hat{\theta} \in \theta_s$, to approximate the summation, i.e., $P(X'|M) \approx P(\hat{\theta}|M)P(X'|\hat{\theta}, M)$.

The second term in the CS approximation is a gross adjustment term. Both its numerator and denominator are expanded using the BIC measure. Ignoring differences between the penalty terms in the numerator and the denominator, we obtain:

$$\begin{aligned} \log P(X|M) \approx & \log P(\hat{\theta}|M) \\ & + \log P(X|\hat{\theta}, M), \end{aligned} \quad (10)$$

where X is the incomplete data. $P(\hat{\theta}|M)$ is the prior probability of the maximum likelihood model parameters. It is computed as the product of the prior probabilities of the individual parameters in the model. The larger and more complex the model is, the more parameters are involved, thus the smaller the product. This, again, automatically enforces the Occam's razor principle for selecting the simpler models for the data.

3.2. Bayesian model selection for HMM model size selection

Our goal for HMM model size selection is to construct the HMM, λ, with the smallest number of states that best models the data, X. In the Bayesian model selection framework, the best model size is the one, which when coupled with its ML parameter configuration, gives the highest model posterior probability, or marginal likelihood. In this section, we describe how the BIC and the CS approximations for marginal likelihood computation are adapted for the HMM model size selection task.

3.2.1. Approximate marginal likelihood for HMM model size selection

Applying the BIC approximation, the marginal likelihood of the HMM λ for data X is computed as:

$$\begin{aligned} &\log P(X|\lambda) \\ &\approx \log P(X|\lambda,\hat{\theta}) - \tfrac{d}{2}\log N \\ &= \sum_{i=1}^{N} \log P(x_i|\lambda,\hat{\theta}) - \frac{d}{2}\log N, \end{aligned} \tag{11}$$

where N represents the number of objects in X, d is the number of significant parameters[2] and θ represents the ML parameters in λ. The likelihood of individual objects x_i given model λ, $\log P(x_i|\lambda,\hat{\theta})$ is computed using the *forward* procedure [3].

On the other hand, when applying the CS approximation, the marginal likelihood of HMM λ is computed as the sum of data likelihood and model prior probability. For the model prior probability computation, we make the following assumptions:

[2]Significant parameters include all the parameters for emission probability definitions and only the initial probabilities and transition probabilities that are greater than a predefined threshold value t. t is chosen to be 10^{-3} for all experiments conducted in this paper.

– For a HMM with M distinct states, the set of initial state probabilities, $\pi_i, i = 1,\ldots,M$, and the transition probabilities between pairwise states, a_{ij}, $1 \leqslant i,j \leqslant M$, follow the multinomial distribution, $\pi_i \geqslant 0, \sum_{i=1}^{M}\pi_i = 1$, and $a_{ij} \geqslant 0, \sum_{j=1}^{M} a_{ij} = 1$, for $1 \leqslant i \leqslant M$. Following Cheeseman and Stutz's work [26], the prior probability distribution of the transition parameter values are assumed to follow the Dirichlet distribution [30]:

$$\begin{aligned} &P(a_{i1},\ldots,a_{iM}|\lambda) \\ &= Dirichlet(a_{i1},\ldots,a_{iM}|\lambda,\alpha_1, \\ &\quad \ldots,\alpha_M) \\ &\approx \prod_{j=1}^{M} a_{ij}^{\alpha_j}, \quad (1 \leqslant i \leqslant M). \end{aligned} \tag{12}$$

$$P(\pi_1,\ldots,\pi_M|\lambda) \approx \prod_{j=1}^{M} \pi_j^{\alpha_j}, \tag{13}$$

The non-negative hyper-parameters, the α_j's, measure the confidence of the prior probability of the model parameters. We choose a uniform distribution over all α_j's (i.e., we assume all the α values are equal).

– For parameters that define the emission probabilities, we assume that the feature mean values in each state, μ_i, are uniformly distributed, and the standard deviation in each state, σ_i, follow Jeffery's prior distribution

$$P(\mu_f|\lambda) = \frac{1}{\mu_{f_{\max}} - \mu_{f_{\min}}}, \tag{14}$$

$$P(\sigma_f|\lambda) = \sigma_f^{-1}[log\frac{\sigma_{f_{\max}}}{\sigma_{f_{\min}}}]^{-1}, \tag{15}$$

where μ_f and σ_f represents the mean and standard deviation value for the fth temporal feature, and $\mu_{f_{\max}}/\mu_{f_{\min}}$ and $\sigma_{f_{\max}}/\sigma_{f_{\min}}$ are the maximum/minimum mean and standard deviation values for feature f.

Therefore, the prior probability of the ML parameters of a HMM is:

$$\begin{aligned} &\log P(\hat{\theta}|\lambda) \\ &\approx log[(\textstyle\prod_{i=1}^{M} P(a_{i1},\ldots,a_{iM}|\lambda))\cdot \\ &\quad P(\pi_1,\ldots,\pi_M|\lambda)\cdot \\ &\quad \textstyle\prod_{i=1}^{M}\prod_{f=1}^{F}(P(\mu_{if}|\lambda)\cdot P(\sigma_{if}|\lambda))] \\ &\propto log[(\textstyle\prod_{i=1}^{M}\prod_{j=1}^{M} a_{ij}^{\alpha_j})\cdot(\prod_{j=1}^{M}\pi_j^{\alpha_j})\cdot \\ &\quad \textstyle\prod_{i=1}^{M}\prod_{f=1}^{F}(\frac{1}{\mu_{f_{\max}}-\mu_{f_{\min}}}\cdot \\ &\quad {\sigma_f}^{-1}[log\frac{\sigma_{f_{\max}}}{\sigma_{f_{\min}}}]^{-1})], \end{aligned} \tag{16}$$

and the CS approximation of the marginal likelihood for the HMM is:

$$\begin{aligned}
&\log P(X|\lambda) \\
&\approx \log P(\hat{\theta}|\lambda) + \log P(X|\hat{\theta},\lambda) \\
&\propto \sum_{i=1}^{M}\sum_{j=1}^{M}\log(a_{ij}^{\alpha_j}) + \sum_{j=1}^{M}\log(\pi_j^{\alpha_j}) \\
&\quad + \sum_{i=1}^{M}\sum_{f=1}^{F} \\
&\quad \log\left(\frac{1}{\mu_{f_{\max}}-\mu_{f_{\min}}}\cdot {\sigma_{if}}^{-1}\cdot[\log\frac{\sigma_{f_{\max}}}{\sigma_{f_{\min}}}]^{-1}\right) \\
&\quad + \sum_{i=1}^{N}\log P(x_i|\hat{\theta},\lambda).
\end{aligned} \tag{17}$$

3.2.2. Characteristics of the BIC and CS measures

To show the characteristics of the BIC and the CS measures for HMM model size selection, we constructed a four-state HMM, λ, using the model construction procedure described in Fig. 7. 15 Data objects are generated from λ by assuming a probabilistic walk through the HMM. Each data object is described by two temporal features and the length of the temporal sequence is 100.

Figure 3 shows the data likelihood, the model complexity penalty/model prior probability, and the BIC and CS values, for the data set. The dashed lines show the likelihood values of data given the learned HMMs of different sizes. The dotted lines show the value of the penalty term (Fig. 3(a)) and the parameter prior probability (Fig. 3(b)) for each model. The solid lines show the BIC (Fig. 3(a)) and CS (Fig. 3(b)) measures as a combination of the above two terms. We observe that as the size of the model increases, the model likelihood also increases and the model complexity penalty and model prior probability decreases monotonically. Both BIC and CS have their highest value corresponding to the correct model size, 4. Given this characteristic of the BIC and the CS measures, when data provided is sufficient, the optimal HMM model size for data may be selected by choosing the model size that corresponds to the highest BIC and CS values.

A number of previous studies have established the dependency between the accuracy of the structure derived versus the amount of data available in model selection problems [28,31,32]. An empirical study of this dependency relation for the HMM learning problem using the BIC and the CS measures as the model selection criteria is presented in a forthcoming paper. In this paper, we assume that the data provided is sufficient for model learning purposes. Next, we discuss our heuristic search control structure for HMM model size selection using the BIC and CS measures. The approach is designed based on the assumption that data provided is sufficient for the learning of HMM model size and model parameters.

3.3. Heuristic search control for HMM model size selection

Under the assumption that the data provided is sufficient and complete, the characteristics of the BIC and CS measures allow us to employ a sequential model expansion strategy with an objective stopping criterion. The expansion process starts with the smallest HMM model, i.e., a one state HMM. The size of the model is increased in steps of 1. After each model expansion step, the parameters of the HMM model is estimated from data using the Baum-Welch procedure. Then, the model is evaluated using the BIC or the CS measure. If the score of the current model decreases from that of the previous model, we conclude that we have just passed the peak value, and accept the previous model as our best model. Otherwise, we continue with the model expansion process. Figure 4 illustrates this iterative search process to determine the HMM size.

Our methodology is similar to the successive state splitting (SSS) approach [16] in that both approaches start with the minimum size model, and increase the model size in the smallest increment until the best model is found. However, our method differs from the SSS approach in the way model parameters are estimated. Instead of performing a local adjustment of parameter values of "affected states" after a state is split, as in SSS, our method performs a global update of the parameters of the entire model after each model expansion step. "Local parameter adjustment" has been shown to generate sub-optimal parameter estimations. In addition, SSS relies on pre-defined model size for terminating the model expansion process, while our method employs an object criterion function for determining the termination condition for the model expansion process.

3.4. HMM model parameter initialization

The quality of the HMM model size selection process is closely linked to the accurate estimation of the HMM parameter values. Like other iterative maximum likelihood approaches, the accuracy of the parameter values estimated using the Baum Welch procedure depends on the quality of the initial parameter values of

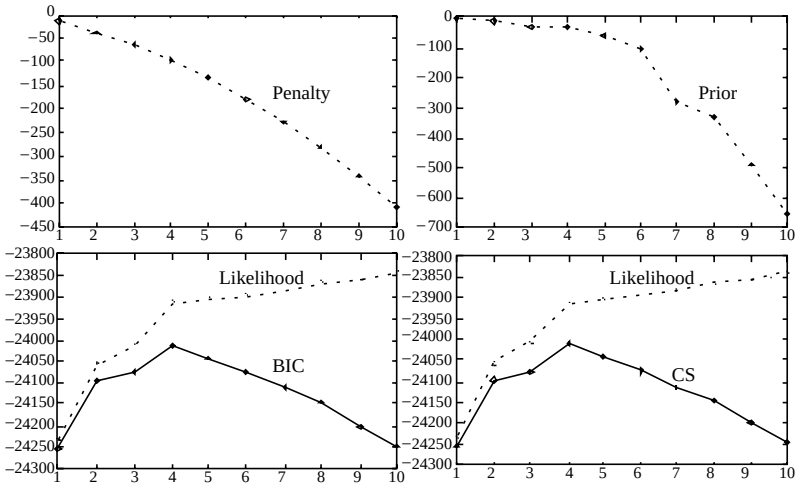

Fig. 3. The characteristics of the BIC and CS measures for HMM model size selection using sufficient data.

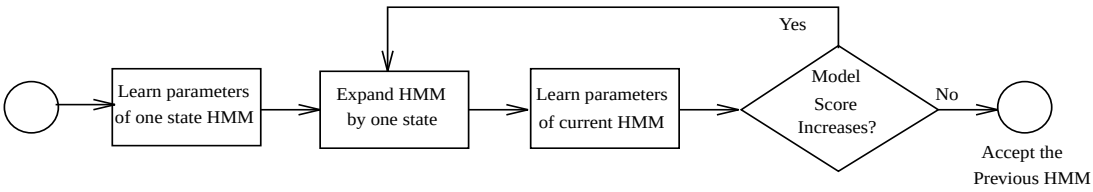

Fig. 4. The model expansion process for HMM model size selection.

the model. The *Viterbi* initialization process is one way of starting with initial parameter values that are better than random. However, it suffers from a number of limitations that we discussed earlier. To address these shortcomings, we have developed a new parameter initialization scheme based on the K-means static data clustering algorithm [33], that we call the *Clustering* initialization process. Our empirical results show that it generates better initial parameter values resulting in more accurate model determination.

The *Clustering* HMM initialization procedure can be described as a four step process, as illustrated in Fig. 5. Given a data set of N objects, described by F temporal features, and each feature has sequence length L, the four steps in the *Clustering* initialization procedure for a M state HMM can be described as:

Step 1: Convert the temporal data into data vector of size $N \times L$ by treating temporal values collected at individual time steps as a static value vector of dimension F.

Step 2: Apply the K-means clustering algorithm [33] to partition this data set into M groups. The structure derived is the one with the minimum overall mean squared error for this M-cluster configuration.

Step 3: Compute the centroids (mean, standard deviation) of these M groups. These vectors become the initial emission parameter values of the M state HMM.

Step 4: Map the individual value vectors in the static data and their group information back to their original positions in the temporal sequences. Accumulate the frequencies of occurrence of the value vectors in all the sequences. Then assign the accumulated frequency statistics to the corresponding transition parameters in the model.

The *Clustering* initialization method takes a more global approach to computing the initial state definition than the *Viterbi* initialization method. Furthermore, since this method depends entirely on data, and not on yet another initial model parameter configuration, the resulting configuration is more stable. Therefore, the state definitions derived for the clusters are better than those based on the *Viterbi* method. This, in turn,

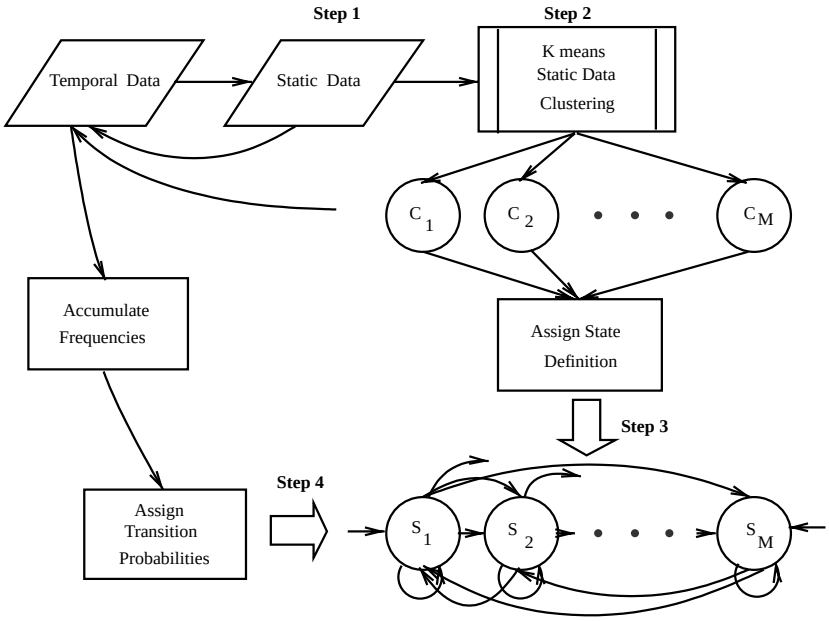

Fig. 5. The *Clustering* parameter initialization method for HMM.

generates transition probabilities that are closer to the true probabilities. Since the parameter configuration derived via the *Clustering* initialization is closer to the ML parameter configuration, it requires fewer *Baum Welch* iterations for the configuration to converge to the ML configuration. In the experimental section, we demonstrate that the *Clustering* initialization results in lesser BW iterations before convergence, and the HMM models derived based on the *Clustering* initialization are of a higher quality than those derived based on the *Viterbi* initialization.

4. Experimental evaluation

In this section, we experimentally evaluate the Bayesian HMM learning procedure. The following three experiments were conducted to evaluate:

- the effectiveness of the HMM model size selection using the BIC and CS measures,
- the effectiveness of the *Clustering* parameter initialization scheme in comparison to the *Viterbi* parameter initialization method for the Baum-Welch parameter estimation procedure, and
- the accuracy of the HMM learning algorithm when compared to the generating HMM.

All experiments are conducted using artificially generated data. We first describe the model and data generation process. Then, we discuss the criteria used for evaluating the experimental results. Finally, we discuss the experimental results and the interpretation of these results.

4.1. Model and data generation

4.1.1. HMM model generation

For experiments one and two, HMMs of different sizes were constructed using the following two step proces:

step 1: assign state definitions by defining the emission *pdf*s for individual features in each state. The mean and standard deviation values were randomly sampled from value ranges [0, 100] and [0,5], respectively.

step 2: assign the state initial and transition probabilities by random sampling from a value range [0, 1], and then normalize the probabilities to ensure $\sum_{i=1}^{M} \pi_i = 1$, and $\sum_{j=1}^{M} a_{ij} = 1, i = 1, \ldots, M$, where M is the number of states in a model.

The goal of the third experiment is to study the modeling capabilities of the HMM learning procedure. To facilitate this, the separability of the state definitions for the HMMs, from which the data was generated, was varied systematically. All the models had four states.

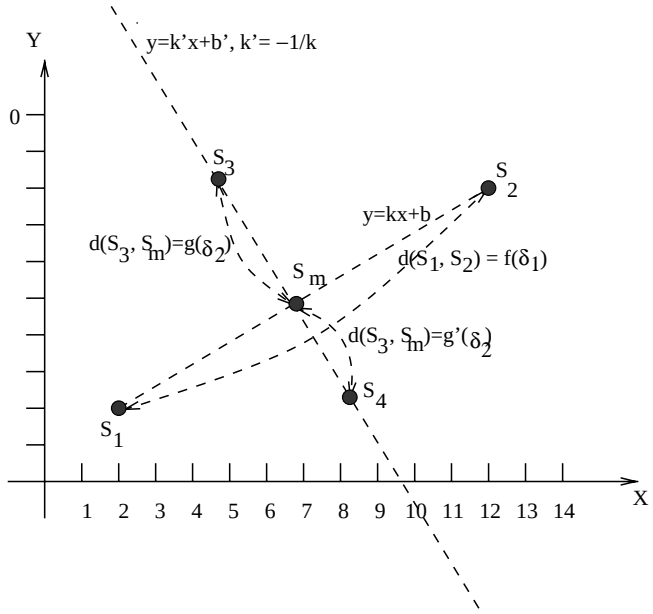

Fig. 6. The construction of 4-state HMMs of different state separability as controlled by δ_1 and δ_2 values.

We divided the four states of each model into two pairs: the *base pair* had states S_1 and S_2, and the *contrasting pair* had states S_3 and S_4. The data objects were described by two temporal features. Therefore, the HMM state definitions, i.e., the mean and standard derivation of the emission parameters were mapped onto the two-dimensional space. Two parameters, δ_1 and δ_2, are used to set the distance between the base pair of states and the distance between the contrasting pair of states to the base pair of states, respectively. The distance between the *base pair* of states, S_1 and S_2, is defined by:

$$D(S_1, S_2) = \delta_1 \cdot \sqrt{(\sigma_1^x + \sigma_2^x)^2 + (\sigma_1^y + \sigma_2^y)^2},$$

where (σ_1^x, σ_1^y) and (σ_2^x, σ_2^y) are standard deviations of the two features in state S_1 and S_2 respectively. The distance from the *contrasting pair* to the mid point, S_m, of the *base pair* of states is defined as:

$$D(S_3, S_m) = \delta_2 \cdot (\sqrt{2}\sigma_3^x + \sigma_d), \quad \text{and}$$
$$D(S_4, S_m) = \delta_2 \cdot (\sqrt{2}\sigma_4^x + \sigma_d),$$

where $(\sigma_3^x = \sigma_3^y)$ and $(\sigma_4^x = \sigma_4^y)$ are the standard deviations of the two features in states S_3 and S_4 respectively, and $\sigma_d = \frac{1}{2}[\max(\sigma_1^x, \sigma_1^y) + \max(\sigma_2^x, \sigma_2^y)]$. To create HMMs of varying state separability, the value of δ_1 was varied between 0.5 and 3.0. For every value of δ_1, i.e., every position of the two base states, the value of δ_2 is gradually varied between 0.1 and 3.0.

The higher the δ_1 value, the more separated are the pair of the *base pair* of states, and the higher the δ_2 values, the more separated are the *contrasting pair* of states to the base pair. The HMMs having $\delta_1 = 3.0, \delta_2 = 3.0$ is the easiest to be rediscovered from data. The HMMs having $\delta_1 = 0.5, \delta_2 = 3.0$ is the hardest to be rediscovered from data. A step by step description of the procedure for generating the model parameters is given in Fig. 7 and illustrated in Fig. 6.

4.1.2. HMM based temporal data generation

Temporal data is generated by assuming a probabilistic walk through the model. Successor states are a function of the current state and the transition probabilities from that state. The initial state is selected randomly with the probability of picking a particular state i being a direct function of π_i. The data generated within each state corresponds the *pdf* for that state. The HMM based temporal data generation procedure is summarized in Fig. 8.

Data sets containing varying number of data objects (between 10 and 60) were generated from different HMMs. Each data object is described by two temporal features, and the sequence length of each feature is set to 100.

For experiments one and two, we generated five different HMMs for each of the four model sizes: 5, 10,

1. Construct the base pair (S_1, S_2)
   1.1 Randomly select the mean values, $\vec{\mu_1} = \begin{bmatrix} \mu_1^x \\ \mu_1^y \end{bmatrix}$, and standard deviations, $\vec{\sigma_1} = \begin{bmatrix} \sigma_1^x \\ \sigma_1^y \end{bmatrix}$, for S_1,
   1.2 Randomly select the standard deviations for S_2 : $\vec{\sigma_2} = \begin{bmatrix} \sigma_2^x \\ \sigma_2^y \end{bmatrix}$,
   1.3 Compute the mean values for S_2 as
   $\vec{\mu_2} = \vec{\mu_1} - \delta_1 \cdot (\vec{\sigma_1} + \vec{\sigma_2})$,
2. Compute k, k', S_m, and σ_d:
   2.1 Compute the slope k of the line, $l_{(S_1,S_2)}$, connecting S_1 and S_2 :
   $k = \frac{\mu_1^y - \mu_2^y}{\mu_1^x - \mu_2^x}$,
   2.2 Compute the slope, k', of the perpendicular line to $l_{(S_1,S_2)}$
   $k' = \frac{1}{k}$,
   2.3 Compute the mid point S_m between S_1 and S_2 :
   $\vec{\mu_m} = \frac{1}{2}(\vec{\mu_1} + \vec{\mu_2})$,
   2.4 Let $d_A = \max(\sigma_1^x, \sigma_1^y)$, and $d_B = \max(\sigma_2^x, \sigma_2^y)$,
   Define $\sigma_d = \frac{1}{2}(d_A + d_B)$,
3. Add state S_3 onto the perpendicular line to $l_{(S_1,S_2)}$,
   3.1 Randomly select $\sigma_3^x = \sigma_3^y$,
   3.2 The distance between S_3 and S_m is determined by parameter δ_2, as :
   $\sqrt{(\mu_3^x - \mu_m^x)^2 + (\mu_3^y - \mu_m^y)^2} = \delta_2 \cdot (\sigma_d^x + \sqrt{2}\sigma_3^x)$,
   3.3 Compute the mean values of the two features in S_3 as:
   $$\begin{cases} \mu_3^y &= \mu_m^y + \frac{\delta_2 \cdot (\sigma_d^x + \sqrt{2}\sigma_3^x)}{\sqrt{k'^2+1}}, \\ \mu_3^x &= \mu_m^x + k' \cdot (\mu_3^y - \mu_m^y) \end{cases}$$
4. Add state S_4 onto the perpendicular line to $l_{(S_1,S_2)}$,
   which is on the opposite side of S_3 given $l_{(S_1,S_2)}$,
   4.1 Randomly select $\sigma_4^x = \sigma_4^y$, (σ_3^x may be different than σ_4^x),
   4.2 The distance between S_4 and S_m is determined by parameter δ_2, as :
   $\sqrt{(\mu_4^x - \mu_m^x)^2 + (\mu_4^y - \mu_m^y)^2} = \delta_2 \cdot (\sigma_d^x + \sqrt{2}\sigma_4^x)$,
   4.3 Compute the mean values of the two features in S_4 as:
   $$\begin{cases} \mu_4^y &= \mu_m^y - \frac{\delta_2 \cdot (\sigma_d^x + \sqrt{2}\sigma_4^x)}{\sqrt{k'^2+1}} \\ \mu_4^x &= \mu_m^x + k' \cdot (\mu_4^y - \mu_m^y) \end{cases}$$

Fig. 7. Model generation procedure for 4-state HMMs with different state separability, controlled by (δ_1, δ_2).

15, and 20 states. For experiment three, for each pair of (δ_1, δ_2) values, five different HMMs are generated. One data set is created from each of these generating HMMs.

4.2. Performance measures

We evaluate the quality of the derived HMMs using the following criteria: (i) the likelihood of the training data computed from the converged HMM, (ii) the size of the HMM derived from data, and (iii) the normalized difference in parameter values between the derive HMM and the true HMM. We compare the effectiveness of the two HMM initialization methods by comparing the number of iterations it took for the Baum Welch procedure to converge. Each of these criteria is described in more detail next.

– *Data likelihood* Data likelihood measures the log likelihood of data to a given HMM model. For a data set X, with N objects, and a HMM, (θ, λ), trained on X, the log likelihood of data is computed as:

$$\log P(X|\theta, \lambda) = \sum_{i=1}^{N} \log P(X_i|\theta, \lambda). \quad (18)$$

Given the same data and the same HMM model size, the higher the data likelihood, the better fit between the model and the data.

– *HMM model size* Starting with data generated from a M-state HMM, the closer the size of the derived HMM model is to the true model size, M, the better the model size selection method.

– *Normalized difference in model parameter values* To evaluate the quality of a derived HMM, we compare the parameter values of the derived model to those of the generating model if they are of the same size. This is done in two steps. First, the correspondence between the states in the derived and the generating models were established by matching states that are the closest in emission defini-

1. Locate the initial state based on $\pi_i, i = 1, \cdots, M$
2. **Repeat** for the desired length of the temporal sequence :
   2.1 Generate a feature value vector based on the *pdf* of the current state
   2.2 Generate a value between [0, 1] based on a uniform distribution
   2.3 Locate the next state depending on the value generated in current state and the transitions leaving the current state

Fig. 8. HMM based temporal data generation.

tions. Next, the normalized difference between the corresponding emission parameter values and the transition probability values is computed as:

$$\text{Normalized difference in emission parameters} = \frac{A_{ik} - \hat{A}_{ik}}{\hat{A}_{ik}},$$

where A_{ik} and $\hat{A}_{ik}$ are the emission parameters for state i, feature k, in the derived and the generating HMMs respectively, and

$$\text{Normalized difference in transition probabilities} = \frac{B_{ij} - \hat{B}_{ij}}{\hat{B}_{ij}},$$

where B_{ij} and $\hat{B}_{ij}$ are the transition probabilities from state i to state j in the derived and the generating model respectively.

– *BW iterations* We measure the total number of Baum Welch iterations required for the parameters of a HMM to converge after initialization.

4.3. Experimental results and interpretation

4.3.1. Experiment one: Effectiveness of the Clustering *parameter initialization method*

The first part of this experiment compares the *Clustering* and the *Viterbi* HMM parameter initialization methods in terms of (i) the number of *Baum Welch* iterations each takes for convergence after the initialization, and (ii) the quality of the derived HMMs, measured in terms of the log likelihood of the training data given the derived models.

Figure 9 compares the log data likelihood, and the number of *Baum Welch* iterations required for HMM convergence using the two initialization methods. In all cases, the HMMs derived from the *Clustering* initialization have an equal or higher data log likelihood than those derived from the *Viterbi* initialization. Furthermore, the *Baum Welch* procedure performed after the *Clustering* initialization requires many fewer iterations for convergence than after the *Viterbi* initialization.

Table 1
HMM sizes re-discovered based on the *Clustering* and the *Viterbi* parameter initialization methods. The mean and standard deviation values are computed from HMM sizes selected over five different data sets

True HMM size	HMM initialization methods	
	Clustering	*Viterbi*
5	5(0)	5.6(0.49)
10	10(0)	10.2(1.17)
15	13(2.2)	7.6(0.8)
20	19.6(1.35)	9(2.37)

The second part of this experiment compares the HMM sizes selected when the two initialization methods were applied. Table 1 illustrates the mean and standard deviation values of the HMM sizes rediscovered from the data. The results show that HMM size selection based on the *Clustering* initialization method is more effective in finding the optimal HMM size for the data than when the *Viterbi* initialization method is used. It is observed that when the *Viterbi* initialization method is applied, in most cases, the HMM sizes selected are sub-optimal, i.e., the sizes of the derived HMMs are smaller than those of the true HMMs. On the other hand, the HMM sizes derived using the *Clustering* initialization correspond to the true HMM sizes in most cases.

Results from these two experiments show that the *Clustering* initialization method is a better choice than the *Viterbi* initialization for HMM parameter initialization. Therefore, the *Clustering* HMM parameter initialization was used in all subsequent experiments.

4.3.2. Experiment two: Effectiveness of the BIC and CS measures in HMM size selection

This experiment compares the effectiveness of the BIC and CS measures in selecting the optimal HMM size for data. Both the BIC and CS criteria measure the fit between the model and data in terms of the data likelihood. The difference between the two measures lies in the model complexity penalty computation. [3]

[3] The CS measure computes the prior probability of the models. When minimum prior knowledge is assumed, i.e., in our case, we

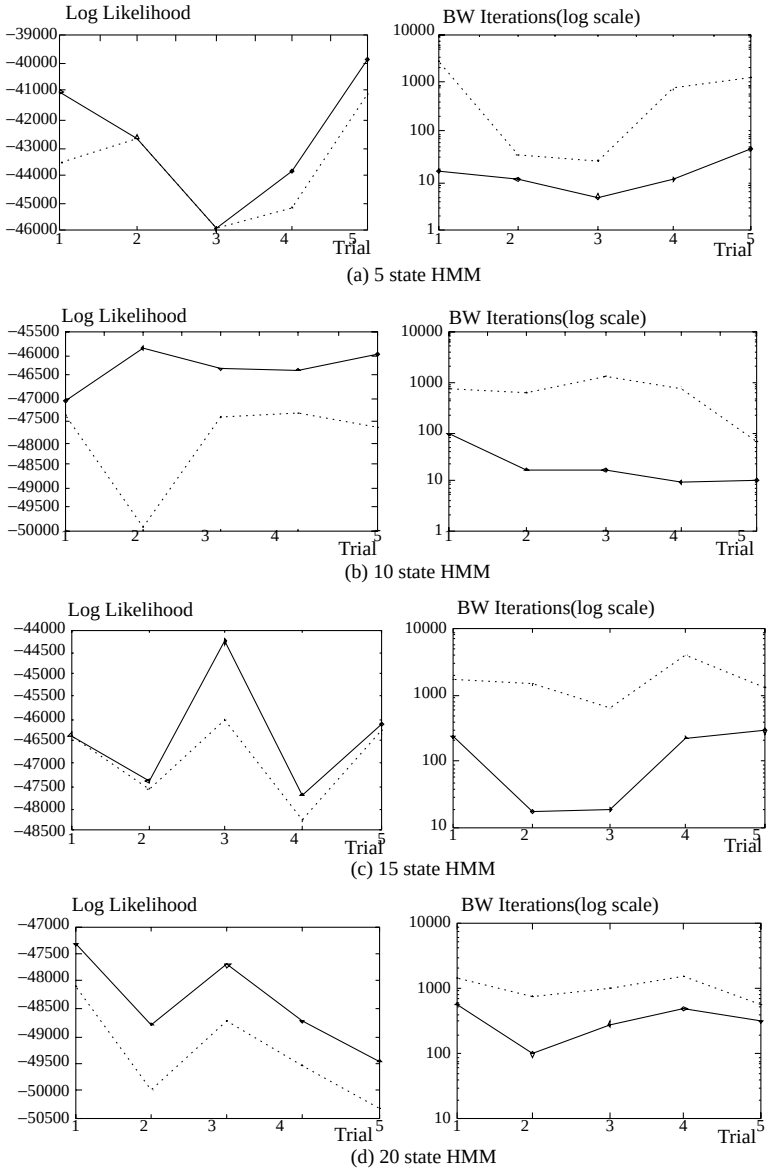

Fig. 9. Comparing the *Clustering* and the *Viterbi* HMM parameter initialization methods with HMMs of different sizes.

The model complexity penalty computation in BIC focuses on the size of the model in terms of the number of significant parameters in the model. The model complexity penalty is directly proportional to the number of the significant parameters in the model. The model complexity penalty computation in CS focuses not only on the size of the model but also the actual values of the model parameters. Model prior probability is computed as the sum of the log prior probabilities of the individual parameter values.

When the model contains many transition probabilities of very small values, for example, when the size of the model becomes large, the log prior probability generated based on these parameter values can be quite low. This leads to a rather small CS value for the model. This may lead to the case where the CS measure peaks at HMM sizes that are smaller than the true optimal model size for the data. We refer to this problem the *low transition value problem.*

assume uniform α values for the prior computation of all state initial probabilities and transition probabilities, model prior probability computation can be thought of as another way of computing the penalty for model complexity.

Table 2
Comparing the BIC and CS measures for HMM model size selection. Results shown here are the mean and standard deviation values (in parenthesis) of the HMM model sizes selected from five different HMMs

True HMM size	Criterion measures	
	BIC	CS
5	5(0)	5(0)
10	10(0)	10(0)
15	13(2.2)	13.2(2.2)
20	19.6(1.35)	17.8(1.6)

Table 2 shows the mean and the standard deviation values of the HMM sizes selected for data. It is observed that the HMM sizes derived using the two measures are comparable. In particular, they selected the same HMM sizes for generating HMMs of 5, 10, and 15 states. For 20-state generating HMMs, the HMM sizes derived using the BIC measure are slightly better than those selected using the CS measure. This can be attributed to the *low transition probability problem* discussed earlier.

It can be concluded that:

- the BIC measure performs equally well for generating HMMs across different sizes;
- The BIC and CS measures are comparable for HMM size selection when the generating HMMs are relatively small; and
- The CS measure is not reliable when the generating HMMs are large

For the rest of the experiments, the BIC measure was used for HMM size selection.

4.3.3. Experiment three: Effectiveness of the Bayesian HMM learning method

In this experiment, we perform a comprehensive study of the effectiveness of our Bayesian HMM learning method. We study the robustness of the HMM size selection and the accuracy of the HMM parameter value estimation, using 4-state generating HMMs of varying difficulty levels. The difficulty level for the generating HMMs is controlled by two variables, δ_1 and δ_2. For each pair of (δ_1, δ_2) values, five different generating HMMs were created. One data set is generated based on each generating HMM.

Table 3 presents the mean and standard deviation values of the HMM sizes selected for generating HMMs of different δ_1 and δ_2 values. It is observed that:

- the HMM size selection method is quite effective. In almost all cases, it correctly rediscovers the HMM sizes for generating HMMs with δ_1 values between 1 to 3, and δ_2 values between 1 and 3; and

Table 3
HMM sizes rediscovered for data generated based on different δ_1 and δ_2 values. Mean and standard deviation values (in parenthesis) are computed from data set generated from five different HMMs with the same (δ_1, δ_2) values

HMM size		δ_2				
	↘	3	2	1	0.5	0.1
δ_1	3	4	4	4	3.2	3.4
		(0)	(0)	(0)	(0.285)	(1.1)
	2	4	4	4	4	3.4
		(0)	(0)	(0)	(0)	(1.1)
	1	3.8	4	4	3.8	3.2
		(0.285)	(0)	(0)	(0.285)	(0.285)
	0.5	3.2	4	3.8	3.8	2.4
		(1.67)	(0)	(0.285)	(0.285)	(0.55)

- the performance of the model selection algorithm is dependent on the separability of the state definitions of the model. The closer the state definitions are to each other, the harder it is to rediscover the correct model size. For any fixed δ_1 value, as the δ_2 value decreases from 3 to 0.1, the average size of the derived HMMs deviate more and more from the true model size. Similarly, when δ_2 is fixed and the value of δ_1 is decreased from 3 to 0.5, the average sizes of the derived HMMs deviate more and more from the true model size. In the first case, the obvious break point, i.e., from which point on most derived models fail to recover the correct model size, is $\delta_2 = 0.1$. In the second case, the break point occurs when $\delta_1 = 0.5$. The performance of the algorithm is the worst when both δ_1 and δ_2 are reduced to the smallest value, i.e., when $\delta_1 = 0.5$ and $\delta_2 = 0.1$.

Next, we compare the parameter values of the rediscovered HMMs to those of the generating HMMs. Comparisons are only made for rediscovered HMMs having size equal to that of the generating HMMs. Results are not shown for the $\delta_1 = 0.5, \delta_2 = 0.1$ runs because HMMs derived for all five data sets have sizes different from those of the generating HMMs.

Tables 4 and 5 show the mean and standard deviation values of the normalized differences of the emission parameter values and the transition parameter values between the derived and the generating HMMs, respectively. It is observed that:

- the normalized differences in both sets of parameters are within a reasonable range. All differences for emission parameters are between 0.008 and 0.04, a majority of which are around 0.01. The differences in transition parameters are between 0.1 and 0.72, with a majority of differences in the range between 0.1 to 0.2; and

Table 4
Normalized differences (mean and standard deviation values) in HMM emission parameter values between the derived and the true HMMs

Normalized difference	↘	δ_2 = 3	2	1	0.5	0.1
δ_1	3	0.009 (0.01)	0.01 (0.01)	0.01 (0.01)	0.01 (0.01)	0.01 (0.01)
	2	0.008 (0.01)	0.01 (0.02)	0.016 (0.02)	0.01 (0.02)	0.02 (0.02)
	1	0.01 (0.01)	0.01 (0.01)	0.01 (0.01)	0.03 (0.04)	0.02 (0.04)
	0.5	0.01 (0.01)	0.02 (0.04)	0.02 (0.03)	0.04 (0.06)	– –

Table 5
Normalized differences (mean and standard deviation values) in HMM transition probabilities between the derived and the true HMMs

Normalized Difference	↘	δ_2 = 3	2	1	0.5	0.1
	3	0.10 (0.1)	0.13 (0.012)	0.15 (0.19)	0.10 (0.17)	0.15 (0.22)
	2	0.09 (0.1)	0.19 (0.27)	0.14 (0.17)	0.15 (0.18)	0.65 (1.75)
δ_1	1	0.13 (0.16)	0.19 (0.28)	0.21 (0.22)	0.22 (0.23)	0.2 (0.2)
	0.5	0.23 (0.41)	0.17 (0.28)	0.25 (0.46)	0.72 (1.48)	– –

– the overall normalized differences in emission parameters are an order of magnitude smaller than the differences in transition parameters. This is because the focus of the *Clustering* parameter initialization is on determining the state definition of the HMMs. Transition probabilities are computed based on the frequency counts of sequence values changing states in consecutive time steps. Any small error that occurs in estimating the state emission definitions will be magnified when estimating the transition probabilities. This also explains the observation that, as the difficulty level for the generating HMMs increases, the normalized differences of the emission parameters are less significant than the differences observed in the transition parameters.

Finally, we study the two exceptional cases that we observed in this experiment. The first case is observed for $(\delta_1 = 1, \delta_2 = 3)$. Out of the five data generated for this case, the size of the generating HMM was not correctly rediscovered. This is considered exceptional because when δ_1 was kept at 1 and δ_2 was reduced to 2, the optimal sizes of all five generating HMMs were correctly rediscovered. In the second case, when $(\delta_1 = 3, \delta_2 = 0.5)$, the optimal sizes of two of the five generating HMMs were not correctly rediscovered. This is also exceptional because when δ_2 was kept at 0.5 and δ_1 was reduced to 2, the optimal sizes of all five generating HMMs were correctly rediscovered. A closer examination of these exceptional cases show that they are all due to the same problem that is the Baum Welch procedure converges to local maxima parameter values. For example, for the case observed for $(\delta_1 = 1, \delta_2 = 3)$, when the data was modeled with a 3-state HMM, because of the close proximity between the base pair of states, the two were combined into a single state, while the contrasting pair of states were correctly modeled with separate states. When the size of the HMM was increased to four, the converged HMM parameter values represented a local maxima parameter configuration. One of the contrasting state split into two states, while the base pair of states remained in a single state. The BIC value for this 4-state HMM was smaller than that of the 3-state HMM. Therefore, 3 was picked by the HMM model size selection procedure as the optimal HMM size for the data. Further analysis of the generating HMM reveals the reason for the convergence to the local maxima parameter configuration. For this generating HMM, σ^x and σ^y associated with one of the contrasting states are large (close to 5), yet the σ values associated with the two base states are rather small (around 1). When the *Clustering* param-

eter initialization method was used to determine the initial parameters of the 4-state HMM, the parameters corresponding to the local maxima configuration was favored because the mean squared error from this configuration was smaller than that of the optimal configuration. Once this inferior initial parameter configuration is adopted, it is very difficult for the hill climbing *Baum Welch* procedure to leap out of the local maxima configuration region.

5. Conclusions and future work

Our focus in this paper is on modeling dynamic system behavior by learning the HMMs from temporal sequence data collected from the system. One problem that prevented the HMM methodology from being used in a wider range of domains is the requirement of predefining the structure of the HMM to be learned, i.e., the number of states in the model. Our Bayesian HMM learning approach addresses this problem by adopting the Bayesian model selection methodology for selecting the best size of the HMM learned from data. To take advantage of the characteristics of the Bayesian model selection criterion functions, a model expansion search control structure is employed that examines HMMs of gradually larger sizes, and stops the search process when further expansion of model size does not improve the quality of the model. In addition to the heuristic procedure on HMM size selection, we also introduced a new HMM parameter initialization procedure that is based on the static numeric data clustering algorithm.

We experimentally demonstrated the effectiveness of the proposed Bayesian HMM learning method using artificially generated data. We compared the BIC and CS measures for the model selection task and concluded that the BIC measure produces more consistent results. We also compared the *Clustering* parameter initialization method with the traditional *Viterbi* initialization method and showed that the *Clustering* parameter initialization method leads to better quality HMM parameter values in shorter time period than the *Viterbi* initialization. From the study of the overall performance of our HMM learning method, we conclude that this procedure is able to find the correct model sizes and parameter values quite accurately for generating HMMs of a wide range of difficulty levels.

The next step is to apply this method to real world examples. When dealing with real world examples, there are a number of issues that needs to be addressed. First of all, in all our experiments, we made the simplifying assumption that all temporal features are independent of each other. This may not be true for real world data. Therefore, the state emission definitions need to be modified to accommodate feature dependence relations. Secondly, real world data tends to contain noises. As discussed in the experiment section, the Baum Welch parameter estimation procedure is quite sensitive to the data and may converge to local maxima parameter configuration. Noise in data makes the accurate parameter estimation task even more difficult. Finally, it is the problem of model interpretation. The goal of applying this method to real world data is to be able to understand the dynamic behavior of the system being modeled by studying the HMM derived from the system data. This typically should involve two steps. First, assign domain specific labels/meanings to the HMM states by interpreting the state emission definitions using domain knowledge. These correspond to the set of stages going through by the system. Then, interpret the dynamics of the system in terms of how it moves among the set of stages by studying the state transition probabilities.

References

[1] D. Jurafsky and J.H. Martin, *Speech and Language Processing*, Prentice Hall, 2000.

[2] S. Rogic, A.K. Mackworth and F.B.F. Ouellette, Evaluation of gene-finding programs on mammalian sequences, *Genomic Research* **11** (2001), 817–832.

[3] L.R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, *Proceedings of the IEEE* **77**(2) (Feb. 1989), 257–285.

[4] W.J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, 1994.

[5] C. Li and G. Biswas, Bayesian temporal data clustering using hidden markov model representation, in: *Proceedings of the 17th International conference on Machine Learning*, P. Langley, ed., Morgan Kaufmann Publishers, 2000, pp. 543–550.

[6] L.R. Bahl, P.F. Brown, P.V. De Souza and R.L. Mercer, Maximum mutual information estimation of hidden markov model parameters, in: *Proceedings of the IEEE-IECEJ-AS International Conference on Acoustics, Speech, and Signal Processing*, (Vol. 1), 1986, pp. 49–52.

[7] Y. Singer and M. Warmuth, Training algorithms for hidden markov models using entropy based distance functions, *Advances in Neural Information Processing Systems* **9** (1996), 641–647.

[8] S. Kwong, Q.H. He and K.F. Man, Training approaches for hidden markov models, *Electronics Letters* **32**(17) (1996), 1554–1555.

[9] L.E. Baum, T. Petrie, G. Soules and N. Weiss, A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains, *The Annuals of Mathematical Statistics* **4**(1) (1970), 164–171.

[10] A.P. Dempster, N.M. Laird and D.B. Rubin, Maximum likelihood from incomplete data via the em algorithm, *Journal of Royal Statistical Society Series B*(*methodological*) **39** (1977), 1–38.
[11] Z. Ghahramani and M.I. Jordan, Factorial hidden markov models, *Machine Leaning* **29** (1997), 245–273.
[12] A. Stolcke and S.M. Omohundro, Best-first model merging for hidden markov model induction, Tech. Rep. TR-94-003, International Computer Science Institute, 1947 Center St., Suite 600, Berkeley, CA 94704-1198, Jan. 1994.
[13] F. Casacuberta, E. Vidal and B. Mas, Learning the structure of hmm's through grammatical inference techniques, in: *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, 1990, pp. 717–720.
[14] M. Brand, Structure learning in conditional probability models via an entropic prior and parameter extinction, *Neural Computation* **11** (1999), 1155–1182.
[15] J. Takami and S. Sagayama, A successive state splitting algorithm for efficient allophone modeling, in: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 1*, 1992, pp. 573–576.
[16] M. Ostendorf and H. Singer, Hmm topology design using maximum likelihood successive state splitting, *Computer Speech and Language* **11** (1997), 17–41.
[17] Z. Ghahramani and M.J. Beal, Graphical models and variational methods, in: *Advanced Mean Field Methods – Theory and Practice*, D. Saad and M. Opper, eds, MIT press, 1999.
[18] Z. Ghahramani and M.J. Beal, Variational inference for bayesian mixtures of factor analysers, in: *Advances in Neural Information Processing Systems*, (Vol. 12), S.A. Solla, T.K. Leen and K.R. Muller, eds, MIT press, Cambridge, MA, 1999.
[19] H. Attias, A variational bayesian framework for graphical models, in: *Advances in Neural Information Processing Systems*, T. et al. Leen, ed., MIT press, Cambridge, MA, 2000.
[20] S. Chib, Marginal likelihood from the gibbs sampling, *Journal of the American Statistical Association* (Dec. 1995), 1313–1321.
[21] G.F. Cooper and E. Herskovits, A bayesian method for the induction of probabilistic network from data, *Machine Learning* **9** (1992), 309–347.
[22] D. Heckerman, D. Geiger and D.M. Chickering, A tutorial on learning with bayesian networks, *Machine Learning* **20** (1995), 197–243.
[23] R.E. Kass and A.E. Raftery, Bayes factor, *Journal of the American Statistical Association* (June 1995), 773–795.
[24] G. Casella and E.I. George, Explaining the gibbs sampler, *The American Statistician* **46**(3) (Aug. 1992), 167–174.
[25] G. Schwarz, Estimating the dimension of a model, *Annuals of Statistics* **6** (1978), 461–464.
[26] P. Cheeseman and J. Stutz, Bayesian classification(autoclass): Theory and results, in: *Advances in Knowledge Discovery and Data Mining*, (Chapter 6), U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, eds, AAAI-MIT press, 1996, pp. 153–180.
[27] J. Rissanen, A universal prior for integers and estimation by minimum description length, *Annual of Statistics* **11** (1983), 416–431.
[28] S. Manganaris, *Supervised Classification with Temporal Data*, Phd dissertation, Vanderbilt University, Dec. 1997.
[29] D.M. Chickering and D. Heckerman, Efficient approximations for the marginal likelihood of bayesian networks with hidden variables, *Machine Learning* **29** (1997), 181–212.
[30] A. Gelman, J.B. Carlin and D.B. Rubin, *Bayesian Data Analysis*, Chapman & Hall/CRC, Boca Raton, 1995.
[31] T. Allen and R. Greiner, Model selection criteria for learning belif nets: An empirical comparison, in: *Proceedings of the Seventeenth International Conference on Machine Learning*, Pat Langley, ed., 2000.
[32] A.N. Jain and B. Chandrasekaran, Dimensionality and sample size considerations in pattern recognition practice, in: *Handbook of statistics*, P.R. Krishnaiah and L.N. Kanal, eds, North-Holland Publishing Company, Amsterdam, 1982, pp. 835–855.
[33] A.K. Jain and D.C. Dubes, *Algorithms for clustering data*, Prentice Hall, 1988.