Knowl Inf Syst (2008) 17:121–133
DOI 10.1007/s10115-008-0124-8

REGULAR PAPER

Adaptive learning of dynamic Bayesian networks with changing structures by detecting geometric structures of time series

Kaijun Wang · Junying Zhang · Fengshan Shen · Lingfeng Shi

Received: 1 January 2007 / Revised: 9 January 2008 / Accepted: 19 January 2008 /
Published online: 26 February 2008

Abstract A dynamic Bayesian network (DBN) is one of popular approaches for relational knowledge discovery such as modeling relations or dependencies, which change over time, between variables of a dynamic system. In this paper, we propose an adaptive learning method (autoDBN) to learn DBNs with changing structures from multivariate time series. In autoDBN, segmentation of time series is achieved first through detecting geometric structures transformed from time series, and then model regions are found from the segmentation by designed finding strategies; in each found model region, a DBN model is established by existing structure learning methods; finally, model revisiting is developed to refine model regions and improve DBN models. These techniques provide a special mechanism to find accurate model regions and discover a sequence of DBNs with changing structures, which are adaptive to changing relations between multivariate time series. Experimental results on simulated and real time series show that autoDBN is very effective in finding accurate/reasonable model regions and gives lower error rates, outperforming the switching linear dynamic system method and moving window method.

Keywords Dynamic Bayesian network · Adaptive learning · Geometric structures of time series · Changing structures · Unsupervised structure learning

1 Introduction

Relational knowledge discovery about relationships among objects is important and useful in many applications [1–4]. Bayesian network (BN) [5–7] is one of the popular tools that exploit

K. Wang (✉) · J. Zhang · F. Shen
School of Computer Science and Technology, Xidian University,
Xian 710071, People's Republic of China
e-mail: sunice9@yahoo.com

L. Shi
School of Electronical and Mechanical Engineering, Xidian University,
Xian 710071, People's Republic of China

relational structures or statistical patterns in real-world applications. A Bayesian network is a directed acyclic graph whose nodes represent variables and directed arcs denote statistical dependence relations between variables, and a probability distribution is specified over these variables. For exploiting relational models from time-series data (called discovery task), it is natural to use a dynamic Bayesian network (DBN) [1,8,9], a directed graphical model whose nodes are across different time slices, to learn and reason about dynamic systems. DBNs model a temporal process by using a conditional probability distribution for each node and a probabilistic transition between time slices.

It is normally assumed for a DBN that its model structure and parameters do not change over time, i.e., the model is time-invariant [1,2]. However, high-fidelity dynamic equations (or models) representing complex real-world systems tend to be nonlinear and time-varying [10], or dependencies and relations between variables change over time. For constructing autonomous DBNs with time-varying structures (but time-invariant within a certain period of time) to follow a complex dynamic system, the key work is to find changing time points between neighbor DBN models with changing structures. Thus, a DBN model will be constructed with the time-series data in a time region between two changing time points (called model region).

This unsupervised discovery task is very difficult, since finding of changing points (or model regions) depends on known model structures, but construction of correct model structures depends on known model regions. The accurate model regions and model structures are dependent on each other, unfortunately, either of them is unknown beforehand. Hence, even a rough approximation of such DBNs with changing structures is not an easy work.

There have been some papers devoted to this topic. A valuable method [11] has been developed for multivariate time series, where dependencies between variables change over time, and it uses a window moving over time slices to search dependency changes of variables and has a good effect. Nevertheless, it has an unsolved problem: how to choose a window size and a moving step that influence model structures and quality much. Another good method—switching linear dynamic system model (SLDS) [12–14] is put forth for dynamic systems, and may be applied to find DBNs with changing structures. SLDS finds switching DBN models by approximate inference, maximum likelihood learning, filtering and smoothing techniques. However, SLDS employs same working procedures at every time point and does not distinguish different model regions, resulting in that many models are constructed with the data across different model regions and tend to be inaccurate.

We propose an adaptive learning method for learning DBNs with changing structures (autoDBN) from time series, focusing on mining geometric structures of time series (heuristic information) to find accurate model regions and their boundaries to overcome the limitations mentioned above. The autoDBN consists of four parts: segmentation of time series by detecting geometric structures of time series, finding reasonable model regions from the found segmentation, DBN learning in found model regions, and model revisiting to refine model regions and rectify possible errors. Our contributions include the above parts except DBN learning in model regions, which is summarized from related literature.

In order to divide time series, we resort to manifold theory [15] to transform time series to curve manifolds, and then detect/find geometric structures of curve manifolds as segmentation of time series. To find reasonable model regions from these segments, three finding strategies are designed for different application cases. In found model regions DBNs are constructed. Finally, model revisiting based on competition F-test is developed to refine model regions and improve models. With these techniques, autoDBN will learn a sequence of DBNs with changing structures adaptive to changing relations between multivariate time series.

The geometric structures and segmentation of time series are proposed in Sect. 2, and how to learn DBNs with changing structures is presented in Sect. 3. Experimental results are shown in Sect. 4. Finally, Sects. 5 and 6 give the discussion and conclusion, respectively.

2 Geometric structures and segmentation of time series

To establish DBN models, it needs to divide time series into segments as potential model regions, comprising two parts: making many piecewise curves fitting local time series and connecting them into one geometric curve (called curve manifold M) along time axis; and cutting M to segments to give segmentation of time series. In addition, different variables might have different behaviors and varying periods, so we deal with time series of each variable separately.

2.1 Projection of one time-series to one curve manifold

Usually, we like to use one polynomial curve to represent complex time series of a variable. However, the polynomial curve found under minimal fitting errors is usually a higher degree polynomial curve, which easily leads to oscillation and over-fitting phenomena [16] when it fits complex time series. Is there any good way to avoid this bad case? Alternatively, we will describe complex time series with many piecewise curves that fit local time series, and low degree polynomials are preferred as piecewise curves. This method is very flexible to employ short curves to fit local time series well without oscillation and over-fitting.

Let a one-dimensional variable such as Y have n time series data, $y_j \in \mathrm{R}(j = 1, 2, \ldots, n)$, and $\{y_j\}$ be divided equally into k groups along time axis t, and there be m data points $y_j (j = (i-1)m+1, (i-1)m+2, \ldots, (i-1)m+m)$ in the ith group of data (called local data). Among low degree (two and three) polynomials, which have similar geometric property when they fit fewer data, quadratic polynomial has fewer parameters and easier computation. Hence, the quadratic polynomial is selected as $f_i(t)$ for the piecewise curve $y = f_i(t)$, $y = at^2+bt+c$, which can be solved with m local data under least square errors [16].

The construction of piecewise/local curves (that fit data groups) and their connections into curve manifold M along time axis t will be discussed within the framework of manifold theory [15]. A local quadratic polynomial curve $C_i : y = f_i(u) (u \in [1, m])$ is first made with the ith group of data (or C_i fitting m local data) under a new Cartesian coordinate system $y\mathrm{O}u$. Then, geometric properties of C_i are taken as geometric properties of local region U_i of M, which is called to be local homeomorphism projection [15]. Thus, all the k local regions and their geometric properties corresponding to k data groups are ready.

Finally, the curve manifold M is constructed when all the local regions of M are connected along axis t, i.e., the n data of Y are projected to M:

$$\mathrm{P} : \{y_j\}_{j=1\sim n} \rightarrow M : M = \cup U_i .$$

It is very hard to realize smooth transition connections between regions of M, while our purpose is to find out scopes of geometric structures of M (but not to make a beautiful geometric curve). For forming good continuity between neighbor local regions, approximate connections are workable and achieved by improving above designs of local curves and regions: using more local data ($3m$ data points) to make local curve C_i, i.e., for each data group we merge its two neighbor data groups to make C_i; and adopting central part of C_i to form U_i. For example, data groups 1, 2 and 3 are merged to make C_2, and only the part of C_2 corresponding to data group 2 is used to form U_2. This approximate connection (no smooth

connection at joints) has little influence on signs of tangent vectors (differential features of M).

In addition, it is important to emphasize that the length of every local curve is crucial for quality of M, and this length is related with the above k and m. Hence, a parameter v, the unified length of every local region, is defined to describe them. Thus, parameter v determines the length of every local curve (or $3v$), number of local curves (or $k = n/v$) and number of data (or $m = v$) in each group.

2.2 Division of a curve manifold to give segmentation of a time series

To divide M to segments, we need define waves and steady regions (called geometric structures of M) to describe the M, detect geometric structures and their scopes with tangent vectors on M, and cut out detected geometric structures to give corresponding segments of a time series.

When we use parameter curves in Definition 2.1, local region U_i is described by $r(u) = (f_i(u), u)$, and then any tangent vector on U_i is $(f_i'(u), 1)$. We use $f_i'(u)$ instead of $(f_i'(u), 1)$ for convenience.

Definition 2.1 (*Tangent vector*) [17]. Let (a, b) be a domain of R, the C^k mapping (or it has successive partial differential with order $k \geq 3$) from (a, b) to R^m be $r : (a, b) \to R^m$, namely $r(u) = (x^1(u), x^2(u), \ldots, x^m(u))$, then $r(u)$ is called a parameter curve in R^m and u is the parameter. And $r'(u_0) = \frac{\mathrm{d}r}{\mathrm{d}u}(u_0)$ is called a tangent vector of $r(u)$ at $r(u_0)$.

Definition 2.2 (*Ascending/descending wave*). Let U be a region of curve manifold M, and U_1 and U_2 be small proximate regions in both sides of U, and let U be a positive or negative domain when tangent vectors on U are all positive or negative. If U is a positive or negative domain but U_1 and U_2 are different domains, then U is called an ascending wave or descending wave of M.

Definition 2.3 (*Wave*). Let U be a domain of curve manifold M. If U contains both ascending waves and descending waves, and all the ascending waves are in front of descending waves under time direction, then U is called a wave of M.

It may be seen that ascending/descending waves are components of waves in Definition 2.3, and are used independently only in special cases (e.g., when they appear in terminals of M). In practice, we detect a wave and find its scope with continuous positive or negative signs of tangent vectors on M (at every time point) according to Definitions 2.2 and 2.3; and a small proximate region in Definition 2.2 is set to contain 2 time points at least.

Based on above definitions, a steady region is defined as a region between two neighbor waves. The steady regions are easily found after all the waves of M are detected. Now, it is straightforward to cut out every wave and steady region from M according to their scopes, and the time series of variable Y is divided into some segments, each of which corresponds to a wave or steady region of M.

3 Adaptive learning of DBNs with changing structures

In this section we will discuss three issues: finding strategies of model regions from segmentation of time series, DBN learning in given model regions, and model revisiting to refine model regions and rectify possible errors derived from incorrect model regions.

3.1 Finding strategies of model regions

A segment of time series obtained in last section is regarded as a potential model region for establishing a DBN model, and time points between model regions are potential changing points of DBN structures. The reason is that a geometric structure naturally represents a varying period of behaviors of a variable, and dependencies between this variable and related variables within such a model region may be described by a fixed DBN model. This is reasonable under the basic assumption: behavior/state changes of a variable are caused by a stationary dependency relation within a model region, which is consistent with the basic assumption of the DBN theory—a process of change is governed by laws that do not themselves change over time [2].

Under this assumption there are four applicable cases: (1) behaviors of all the variables have consistent varying periods or changing points; (2) majority of variables have consistent varying periods or changing points (the following simulated example is this case); (3) one variable (or several variables with consistent varying periods) and its dependency relations are the focus (the following real examples are this case); (4) several variables with inconsistent/interlaced varying periods are the focus. These different cases are from that segmentations of different variables might be consistent or inconsistent in time axis.

For these cases, we design three strategies to find reasonable model regions: the first strategy [for cases (3) and (4)] is to choose one focus variable according to our purpose and prior knowledge, and adopt segments of time series of this variable as model regions; the second strategy [for cases (1) and (2)] is to select a representative variable that has the most changing points consistent with all the changing points occurring twice and more, and use segments of time series of the representative variable as model regions; the third strategy [for case (2)] is to find out those potential changing points occurring frequently, and then take time regions between frequent changing points as model regions. In addition, one series of model regions and models for one focus variable are recommended for case (4), since it is impossible for any model region to both contain and differentiate two different types of varying periods in most cases.

3.2 Structure learning of DBN in model regions

A DBN model is learned from the data within each model region (or structure learning of DBNs). The learning process includes specifying DBN structure space, choosing a scoring criterion, and employing a searching procedure [1].

A DBN is a pair (B_0, B), where B_0 is a prior BN and B is a two-slice temporal BN (2TBN), which defines $P(X_t|X_{t-1})$ by means of a directed acyclic graph (DAG) as follows [1]:

$$P(X_t|X_{t-1}) = \prod_{i=1}^{q} P(X_t^i|\mathrm{Pa}(X_t^i)),$$

where X_t^i is the ith node at time t and $\mathrm{Pa}(X_t^i)$ are the parents of X_t^i in the DAG.

When learning a DBN from given data, we indicate learning 2TBN. The 2TBN is a first-order Markov DBN, where a conditional probability distribution $P(X_t^i|\mathrm{Pa}(X_t^i))$ is specified only for each node X_t^i in the second slice t. The structure learning task aims to find the optimal DBN structure in the structure space, which contains all possible DAG structures. A scoring criterion such as Bayesian information criterion (BIC) is adopted to evaluate which

candidate model is optimal, and a procedure is employed to search the structure space for an optimal DBN structure with the highest score.

The structure learning method based on a local or global search algorithm that searches through structure space (called space-search learning) is applicable to the case that all the variables are observable, while the structure learning method based on the expectation maximization (EM) algorithm and searching of structure space (called EM-search learning) is applicable to the case that some variables are observable and some are unobservable/hidden. Which learning method should be adopted depends on a special application. If we judge that the adding of hidden variables to a DBN model is more reasonable to solve a problem, we invent hidden nodes in the model and adopt the EM-search learning method.

3.3 Model revisiting

The model revisiting is necessary to refine model regions and rectify possible errors derived from incorrect model regions, including: check whether there are inappropriate model regions and rectify them if any, and detect whether neighbor DBNs are the same and merge them if yes.

The revisiting of every found DBN model and its model region is carried out by a competition process, and starts from boundaries between neighbor model regions. Let "governing" model G be in its model region R (covering time region $[t_1, t_n]$), and L_m be a shortest region size for model revisiting. The model revisiting procedure is designed in the following:

(1) cutting R to $k = (t_n - t_1)/L_m$ sub-regions (no cutting if $t_n - t_1 < 2L_m$) starting from its left boundary: $R_1 = [t_1, t_2]$, $R_2 = [t_2 + 1, t_3]$, ..., $R_c = [t_c + 1, t_{c+1}]$, where $t_c + 1 < t_n/2, t_{c+1} >= t_n/2$ and the length of each R_i is L_m, and the left $k/2$ sub-regions are used; Similarly, cutting R to k sub-regions starting from right boundary and the right $k/2$ sub-regions are ready. This design indicates left and right revisiting;
(2) in the k sub-regions, k DBN models $\{G_i\}$ are established respectively, and G_i is a competitor of G if it is different with G;
(3) running a competition F-test process in every sub-region, where each competitor G_i competes for replacing G in R_i. If G_i wins, there is a new model region R_i, otherwise no adjustment of R;
(4) if some new model regions emerge, updating model regions, and reconstructing a series of (improved) DBN models in refined model regions. Otherwise, go to step (5);
(5) detecting if structures of neighbor DBNs are identical, and merging their model regions if yes.

The next work is to design the hypothesis-test on quality of models (or prediction accuracy of a model) to realize the competition process. For the rectification task, model G takes priority in the competition, and its rival G_i can win only when the quality of G_i is obviously better than that of G. For prediction accuracy of a model, it is usual to calculate error sum of squares (SSE) between observations and predictions by a model. The errors between observations and predictions by a good model vary around zero, and may be regarded from a Gaussian distribution with zero mean. Hence, we compare quality between two models by variances of their prediction errors (Var), which are as same as prediction SSEs. Thus, the competition rule is: a competitor (G_i) wins only when its error variance Var(G_i, R_i) is significantly smaller than Var(G, R_i) of G in sub-region R_i. Whether Var(G_i, R_i) is significantly smaller than Var(G, R_i) will be judged by the hypothesis-test method—F-test on two variances from two independent distributions at the commonly-used level of significance $\alpha = 0.05$ [18].

The competition F-test process is designed in the following:

(1) for each variable, calculate the prediction errors $\{e_j\}$ and error variance var($\{e_j\}$) by model G and data in R_i; and then sum var($\{e_j\}$) of all the variables (except the notice below) to give Var(G, R_i). Similarly, Var(G_i, R_i) by G_i and data in R_i is ready. It is noticed that this sum excludes the variables with same parent nodes in both G and G_i, focusing on differences of two models;
(2) in sub-region R_i, null hypothesis H_0 is set for G, and alternative hypothesis H_1 is set for its rival G_i, i.e., H_0 : Var(G_i, R_i) >= Var(G, R_i), and H_1 : Var(G_i, R_i) < Var(G, R_i);
(3) compute ratio $F = \text{Var}(G_i, R_i)/\text{Var}(G, R_i)$, a value of F-distribution with $n_1 - 1$ and $n_2 - 1$ degree of freedom (here $n_1 = n_2$ is the sample size in sub-region R_i);
(4) let critical value $F_{1-\alpha}(n_1-1, n_2-1)$ of F-distribution be ready. If $F >= F_{1-\alpha}(n_1-1, n_2-1)$, reject H_0 and competitor G_i wins; otherwise, do not reject H_0 and competitor G_i fails.

4 Experimental results

In this section, we test the autoDBN about its performance of finding DBNs with changing structures. For comparison, the window method and DBN-based SLDS are also used in the experiments. For learning DBN structures in a given model region, all three methods use the same learning procedure. In addition, every method employs a hidden state variable to record which of the candidate models it yields at each time point. We implement the autoDBN (with Matlab) except the construction of DBN models, which is completed by Bayes net toolbox [19] and the structure learning package [20] that provide necessary functions for building DBNs.

The artificial time-series data added with Gaussian noise of variance 0.1 for three variables are generated from four different models in corresponding time periods: $Z(t) = 4Y(t) - 20$ where $Y(t) = 9.5+5\ \sin(0.031t)$ for $t = 1-100$; $Z(t) = 2X(t)-9.5$ where $X(t) = 14.5+5\ \sin(0.065t-0.5)$ for $t = 201-250$; $Z(t) = 2X(t)-9.5$ where $X(t) = 12+6\ \sin(0.028t-0.4)$ for $t = 260-330$; $Z(t) = X(t)+Y(t)-4.5$ where $X(t) = 19.5+5\ \sin(0.065t+3.6)$ and $Y(t) = 0.12t - 39$ for t=401-600; and others are from constant values (or none model) where $X(t) = 15$, $Y(t) = 10$ and $Z(t) = 20$ for other t. The knowledge discovery task is to find changing linear relations (or models) between the three variables from the time series.

The first real data set is about the interest rates of Australia, France, UK and US in 240 months from July 1980 to June 2000 [21], where the interest-rate correlations or dependencies between the four countries change over time. The knowledge discovery task is to discover changing interest-rate correlations between US and other countries.

Another real data set is about daily stock prices of ten aerospace companies from January 1988 to October 1991 [22], where stock-price correlations between ten companies change over time. The knowledge discovery task is to discover which stocks in which time periods have price-varying correlations with that of the fifth company. As some stocks have similar price behaviors, we adopt K-means clustering method [23] to cluster ten stocks into six clusters, and then choose a representative stock for each cluster. Thus, we can achieve the knowledge discovery task about ten stocks by six representative variables.

In the experiments, different window sizes and minimal sizes of model regions are used to test all three methods, and once moving step of a window is set to be half of a window size. The minimal region sizes for model revisiting are set to be around half the median of

all the model region sizes (or 45, 20 and 23 for the artificial, interest-rate and stock datasets respectively), and is determined finally based on application backgrounds: 18 or 12 (1.5 or 1 year) for the interest-rate data and 20 (1 month) for the stock data. Moreover, the second finding strategy is used for the artificial data, while the first finding strategy is adopted for the interest-rate and stock data sets. In addition, considering that the discovery tasks in the real-dataset experiments are to discover simple correlations between a pre-specified variable and other variables, we preset the maximum parent-set size of 1 for every node.

The initialization of SLDS includes: the prior probability of every switch state (corresponds to a possible BN structure) is set to meet the uniform distribution; the preset minimal sizes of model regions are listed in the following tables, and the preset maximal one equals the double minimal size. For searching a DBN model with the highest probability at each t, the size scope of possible model regions around t is limited by the preset minimal and maximal sizes.

The error rates for all three methods in the following tables are the percentage of wrong models among all the found models at every time point. It is noted that transformable linear relations (e.g., $Z(t) = 4Y(t) - 20$ and $Y(t) = 0.25Z(t) + 5$) are regarded as a same model, thus an involvement criterion is adopted in experiments: a model is correct if it belongs to any form of transformable relations.

Table 1 gives error rates of the three methods under different window/region sizes for the simulated data. One can see that autoDBN is the best (note that a same solution is gained at sizes from 30 to 50 but not listed), while SLDS has error rates of 27.3–56.0% and window method 20.0–55.0%. Table 2 lists the outcome for the interest-rate data and shows: autoDBN has the best solutions with error rates of 7.1 and 9.6% at its optimal sizes; while SLDS has error rates of 11.7–19.6% and window method 10.8–17.9%. Table 3 displays the result for the stock data, where we can find: autoDBN has the best solutions with error rates of 8.8% at its optimal size of 20; while SLDS has error rates of 23.3–46.3% and window method 23.7–43.5%. Figure 1 is an illustration of model regions found by the autoDBN and window method for the simulated data.

Table 1 Error rates (%), number of found model regions (R), and number of found models (M) by three methods under different region/window size (Size) for simulated data

Size	20	40	60	80	100	120
autoDBN	/	2.0%, 5R	/	/	/	/
SLDS	56.0%, 16M	37.8%, 21M	28.3%, 15M	27.3%, 13M	38.3%, 10M	42.2%, 10M
Window	55.0%, 36R	31.7%, 16R	21.7%, 12R	21.7%, 8R	20.0%, 5R	33.3%, 6R

Table 2 Error rates (%), number of found model regions (R), and number of found models (M) by three methods under different region/window size (Size) for interest-rate data

Size	12	18	24	30	36
autoDBN	7.1%, 7R	9.6%, 6R	/	/	/
SLDS	17.1%, 44M	11.7%, 30M	14.6%, 10M	15.4%, 9M	19.6%, 8M
Window	17.9%, 30R	14.6%, 21R	14.6%, 14R	10.8%, 12R	16.3%, 11R

Table 3 Error rates (%), number of found model regions (R), and number of found models (M) by three methods under different region/window size (Size) for stock data

Size	20	50	100	150	200	300
autoDBN	8.8%, 19R	/	/	/	/	/
SLDS	46.3%, 62M	42.5%, 48M	23.3%, 26M	31.9%, 25M	35.1%, 21M	39.7%, 14M
Window	43.5%, 79R	27.7%, 33R	32.1%, 15R	26.8%, 11R	23.7%, 9R	34.8%, 6R

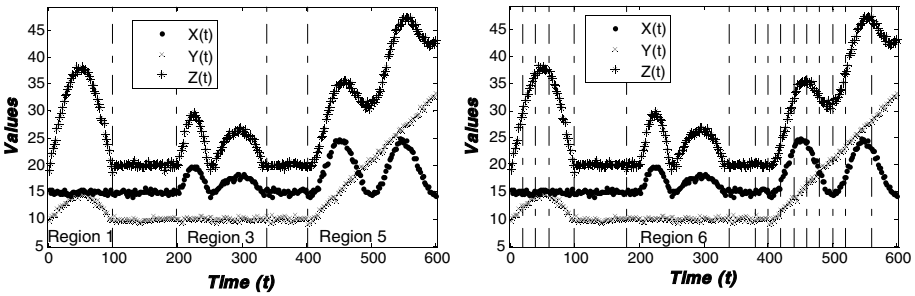

Fig. 1 Model regions divided by dash-dot lines are found for simulated time series. *Left*: 5 model regions of autoDBN; *Right*: 16 model regions of window method under window size of 40

5 Discussion

It is seen that the performance of window method (error rates and number of found model regions) depends greatly on preset window sizes, but the optimal sizes are unknown for an unsupervised discovery task. Thus, the window method works under blind search so that it is hard to display its best performance. Furthermore, the window method can not distinguish boundaries of model regions well, since good discrimination of boundaries requires small window sizes, but models established in small window sizes tend to overfit to few local data and give worse performance. For SLDS, it is also seen that its performance (error rates and number of switch models) depends much on preset minimal sizes of model regions. However, it is hard to known an optimal size beforehand so that SLDS can not exhibit its best performance generally. Moreover, as forward and backward learning (or filtering and smoothing) processes of SLDS work samely at every time point and model regions are unknown, switching models in and nearby true boundaries of model regions are probably constructed with the data crossing two or more model regions, resulting in inaccuracy of models and poor discrimination of boundaries. In brief, the two methods have no special mechanism of detecting boundaries of model regions, and this limitation deteriorates quality of models nearby boundaries so that they can not achieve low error rates for unsupervised discovery tasks, which has been shown in the experiments.

In contrast, autoDBN focuses on finding reasonable model regions and their boundaries based on detecting geometric structures of curve manifolds. It is a heuristic method to find potential model regions from curve manifolds, since a curve manifold (and its geometric structures) outlines both varying tendencies of behaviors of a variable and their changing points. Furthermore, the model revisiting is designed to refine found model regions and rectify possible errors derived from incorrect model regions. Therefore, with these good mechanisms of finding reasonable model regions, autoDBN can outperform two comparison

methods, which has been demonstrated in the experiments (lower error rates of autoDBN in the experiments).

Now we discuss parameters in autoDBN. The scope of v is set from 2 to 7 (7 is enough for forming obvious geometric structures as per our experiences), and the optimal v is found at the v where the number of waves n_{w} is the minimum for $v = 2 \sim 5$ or at the first v where n_{w} is stable with rise of v ("stable" means that the increase of n_{w} is less than 10% when $v = v + 1$ after the first v). This design indicates that parameter v is determined adaptively.

Another discrimination parameter w, a ratio of variance of a wave to variance median of all the waves, is used to identify relatively large varying behaviors (while weak varying behaviors fall in steady regions). The default $w = 0.1$ need not be adjusted in general, since 0.1 is such a small percentage that does not influence the discovery of large varying behaviors, which are our focus. It is worth notice that concept "weak" itself is vague and depends on a special application and our purpose. If one focuses on weak varying behaviors, he may lower the w to such as $w/2$ and $w/4$, and then shorter model regions containing weak varying behaviors will appear if any.

For the third finding strategy, parameter pw is designed to group two time points with a distance less than pw to one unified changing point, and default pw $= 10$ is workable in most cases. However, how close changing points should be distinguished or merged depends on a special application and our purpose. To contain farther time points as a same changing point when one pays more attention to wider varying of a system, increase pw; inversely, decrease pw to only include closer time points as a same changing point if one cares about fine varying of a system, but to decrease pw is not recommended due to its tendency of overfitting to few data.

Minimal region size L_{m} for model revisiting needs to be preset. In order to avoid visiting short regions where constructed models tend to overfit to local data, a good choice is to set L_{m} equal to or around L_{h} (half the median of all the model region sizes) based on application background. In addition, small variation of L_{m} around L_{h} influences performance less, e.g., error rate $E = 9.6\%$ at $L_{\mathrm{m}} = 18$ and $E = 13.7\%$ at $L_{\mathrm{m}} = 22$ when $L_{\mathrm{h}} = 20$ with $E = 10.4\%$ in the interest-rate experiment.

In a word, all the parameters need not be set by users generally, except that it is better to choose parameter L_{m} around L_{h} based on application background. In a special case, one may adjust parameters w and pw as per the guidance in above discussion.

The three finding strategies can work well for the four applicable cases mentioned in Sect. 3. However, for the most complex case where many variables are the focus and have inconsistent and interlaced varying periods, it is hard to satisfy all the focus variables with one unified DBN model, where (large) information loss for some variables is unavoidable, in any model region. Hence, one series of model regions and models for one focus variable is an accurate modeling way for this case. For example, that a series of DBNs is constructed for Au (the variable of interest rates of Australia) and another series of DBNs for Fr (the variable of interest rates of France) is an accurate modeling way in the interest-rate example when both the Au and Fr are our focus.

As expected, in the experiments autoDBN gives better results, indicating its success in finding reasonable model regions and learning a series of DBNs with changing structures.

6 Conclusion

A DBN has been used to exploit relational models from multivariate time-series data, but the adaptive learning of DBNs with changing structures is still a challenging task. For this task,

we propose autoDBN method. By means of detecting geometric structures of time series for accurate model regions, autoDBN can find a sequence of DBNs with changing structures adaptive to changing dependencies between multivariate time series. The working assumption of autoDBN is that state changes of a variable are caused by a stationary dependency relation within a model region, which is consistent with the basic assumption of the DBN theory.

The autoDBN utilizes heuristic information from varying behaviors of variables, and provides a special mechanism to pursue accurate model regions: segmentation of time series through detecting geometric structures of time series, finding model regions from found segments, and model revisiting to refine model regions. This mechanism is critical for quality of DBN models. In contrast, SLDS does not provide a special technique to detect model regions and their boundaries, but serves equitably at every time point with same learning procedures; and the window method finds model regions depending on given window size and moving step so that it is a blind/gambly search. These limitations will deteriorate quality of models nearby true boundaries of model regions for the two methods. Therefore, autoDBN is more accurate and efficient than SLDS and moving window method for unsupervised discovery task, which is demonstrated in the experiments.

On the other hand, autoDBN is inapplicable to frequent changing cases that state changes within a model region are caused by changing dependency relations, which is out of the basic assumptions of this paper. In addition, autoDBN is developed only for continuous variables at present.

Acknowledgments The authors would like to acknowledge Allan Tucker and Chao Wang for their valuable comments, and especially, appreciate the anonymous reviewers for their helpful suggestions and comments. This work is supported in part by Natural Science Found of China (No. 60574039 and No. 60371044)

References

1. Murphy KP (2002) Dynamic Bayesian networks: representation, inference and learning. PhD thesis, University of California Berkeley. http://www.cs.ubc.ca/~murphyk
2. Russell S, Norvig P (2003) Artificial intelligence: a modern approach, 2nd edn. Pearson Education, Inc., Upper Saddle River
3. Schuster A, Wolff R, Trock D (2005) A high-performance distributed algorithm for mining association rules. Knowl Inf Syst 7(4):458–475
4. Berti-Equille L (2007) Data quality awareness: a case study for cost optimal association rule mining. Knowl Inf Syst 11(2):191–215
5. Pearl J (1988) Probabilistic reasoning in intelligent systems. Morgan Kauffman, San Mateo
6. Neapolitan RE (2003) Learning Bayesian networks. Prentice Hall, Englewood Cliffs
7. Chen R, Sivakumar K, Kargupta H (2004) Collective mining of Bayesian networks from distributed heterogeneous data. Knowl Inf Syst 6(2):164–187
8. Pena JM, BjÄorkegren J, Tegner J (2005) Learning dynamic Bayesian network models via cross-validation. Pattern Recognit Lett 26(14):2295–2308
9. Yu J, Smith VA, Wang PP, Hartemink AJ, Jarvis ED (2004) Advances to Bayesian network inference for generating causal networks from observational biological data. Bioinformatics 20(18):3594–3603
10. Ledin J (2001) Simulation engineering. CMP Books, The Netherlands
11. Tucker A, Liu X (2004) A Bayesian network approach to explaining time series with changing structure. Intell Data Anal 8(5):469–480
12. Pavlovic V, Rehg JM, Cham TJ, Murphy KP (1999) A dynamic Bayesian network approach to figure tracking using learned dynamic models. In: ICCV 1999, pp 94–101
13. Pavlovic V, Rehg JM, MacCormick J (2000) Learning switching linear models of human motion. In: NIPS 2000, pp 981–987
14. Barber D (2006) Expectation correction for smoothed inference in switching linear dynamical systems. J Mach Learn Res 7:2515–2540

15. Chen W (2001) An introduction to differential manifold. High Education Press, Beijing
16. Mo G, Liu K (2003) Methodology of function approximation. Science Press, Beijing
17. Mei X, Huang M (2003) Differential geometry. Beijing Normal University Press, Beijing
18. Walpole RE, Myers RH, Myers SL, Ye K (2002) Probability and statistics for engineers and scientists, 7th edn. Pearson Education, Inc., Upper Saddle River
19. Murphy KP (2006). Bayes net toolbox for MATLAB. http://bnt.sourceforge.net/.
20. Leray P, Francois O (2004) BNT structure learning package. http://bnt.insa-rouen.fr/ajouts.html
21. Maharaj E (2002) A pattern recognition of time series using wavelets. In: 15th Computational Statistics Conference of the International Association of Statistical Computing, Berlin
22. StatLib repository: http://lib.stat.cmu.edu/; http://www.liacc.up.pt/~ltorgo/Regression/stock.tgz
23. Jin R, Goswami A, Agrawal G (2007) Fast and exact out-of-core and distributed k-means clustering. Knowl Inf Syst 10(1):17–40

Author Biographies

Kaijun Wang received his B.S. degree in power mechanical engineering from Shanghai Jiao Tong University and his M.S. degree in Computer Science from Xidian University, China in 2002. He is currently a Ph.D. student in the School of Computer Science and Technology, Xidian University, Xian, P. R. China. His research interests include artificial intelligence, data mining, pattern recognition and bioinformatics.

Junying Zhang received her Ph.D. degree in Signal and Information Processing from Xidian University, Xi'an, China, in 1998. From 2001 to 2002, she was a visiting scholar at the Department of Electrical Engineering and Computer Science, Catholic University of America, Washington, DC, USA, and at the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, USA, in 2007. She is currently a Professor in the School of Computer Science and Technology, Xidian University. Her research interests cover intelligent information processing, including machine learning and its application to cancer related bioinformatics, image processing, radar automatic target recognition, and pattern recognition.

Fengshan Shen received a B.S. degree from the Liberation Army Information Engineering University, Zhengzhou, China in 1993 and an M.S. degree from Zhengzhou University in 1999. From 2000 to 2005, he did research work in the College of Information Engineering of Zhengzhou University, Zhengzhou, P. R. China. He is currently a Ph.D. student in the School of Computer Science and Technology, Xidian University, Xi'an, P. R. China. His research interests include pattern recognition and computer software.

Lingfeng Shi is currently an associate professor and advisor for master in the School of Electronical and Mechanical Engineering, Xidian University, P. R. China. He received his B.S. degree from Measured Institute of China in 1995 and M.S. degree in Electric Engineering from Xidian University in 2003. His interests include radar imaging, circuit and system, ground-penetrating radar, and signal processing. He has authored over 10 technical papers and owned a patent.