Predictive complex event processing based on evolving Bayesian networks

Yongheng Wang, Hui Gao, Guidan Chen

PII: S0167-8655(17)30148-4
DOI: 10.1016/j.patrec.2017.05.008
Reference: PATREC 6814

To appear in: Pattern Recognition Letters

Received date: 21 January 2017
Revised date: 3 April 2017

Please cite this article as: Yongheng Wang, Hui Gao, Guidan Chen, Predictive complex event processing based on evolving Bayesian networks, Pattern Recognition Letters (2017), doi: 10.1016/j.patrec.2017.05.008

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
Highlights

- Bayesian network model uses Gaussian mixture model and EM algorithm for approximate inference.
- Incremental calculation of score metric for updating Bayesian network structure.
- Evolving Bayesian network structure using hill-climbing algorithm
- Hill-climbing algorithms that explores multiple peaks parallel.
- Updating score metrics based on changed edges only.
Predictive complex event processing based on evolving Bayesian networks

Yongheng Wang *, Hui Gao and Guidan Chen

College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

ABSTRACT

In the Big Data era, large volumes of data are continuously and rapidly generated from sensor networks, social network, the Internet, etc. Predicting from online event stream is an important task since users usually need to predict some future states and take some actions in advance. Many applications need online prediction models which can evolve automatically with data distribution drift and algorithms which can support single-pass processing of data, which are still faced with many challenges. In this paper, the authors propose a predictive complex event processing method based on evolving Bayesian networks. The Bayesian model is designed based on event type and time with inference method based on Gaussian mixture model and EM algorithm. When learning the structure of Bayesian network from event streams, this method supports calculating score metric incrementally when new data is arrived or edges in the network are changed. Evolving Bayesian network structure is supported based on hill-climbing method. The system can continuously monitor the Bayesian network model and modify it if it is found to be not appropriate for the new incoming data. The method of this paper is evaluated in road traffic domain with both real application data and data produced by a simulated transportation system. The total percentage error is 8.12% for real data and 7.78% for simulated data, while the best result for other methods is 11.79% for real data and 14.59% for simulated data. The experimental evaluations show that this method is effective for predictive complex event processing and it outperforms other popular methods when processing traffic prediction in intelligent transportation systems.

Keywords: Event Stream; Predictive Complex Event Processing; Evolving Bayesian Networks

1. Introduction

In the Big Data era users require new technology to process data stream with high speed and variety of data type. Besides the traditional stream processing technology, it is important to catch the relations inside online streaming data. Most of the data streams are composed of primitive events that produced by sensor networks, internet, social networks, etc. The semantic information inside primitive events is quite limited. In real application, people usually pay more attention to higher level information such as business logic and rules. For example, enormous events are generated in a trading system, but a fraud detecting system only care about the events that can cause frauds. Complex Event Processing (CEP) [1] is the technology that interprets and combines streams of primitive events to identify higher level composite events. CEP has been used in many areas, such as sensor networks for environmental monitoring, continuous analyzing of stocks to detect trends, etc.

Predictive Complex Event Processing is the technology to identify events before they have happened, so that they can be eliminated or their effects mitigated [2]. For example, a financial institution wishes to detect frauds or a financial regulator wishes to catch illegal trading patterns in advance. Another example is to predict the traffic status in road networks and take some actions proactively to mitigate or eliminate undesired future states. Simple predictive CEP can be supported by rule-based method which means users define some patterns and the system then continuously monitor the event streams to predict future events. However, for complex cases it is not easy to define predictive CEP patterns exactly. In such circumstance, predictive analytics technology can be applied to support predictive CEP. Predictive analytics applies several statistical and data mining techniques such as clustering, classification, regression and so on. Recently, neural networks (especially deep neural networks) and Bayesian Networks (BN) are commonly used as prediction models. BN and it variations, such as Dynamic Bayesian Networks (DBN) [3], Adaptive Bayesian Networks (ABN) [4], etc., are widely used in predictive analytics because they have the following advantages: (i) they allow to express directly the fundamental qualitative relationship of direct causation, (ii) there exists mathematical methods to estimate the state of certain variables given the state of other variables, (iii) there are methods in order to explain to the user how the system came to its conclusions [5].

Currently predictive CEP with predictive analytics has some challenges. First, traditional predictive analytics methods are designed for database which means they assume all data is available at any time. However, in predictive CEP the system can only process data on single-pass and cannot control over the order of samples that arrive over time. Stream-based predictive analytics methods are needed which is more difficult. Second, the distribution of data can change over time. A model learned from historical data may not fit the new coming data well. This means real time modeling and learning algorithms are needed. In order to support predictive CEP with BN, an Evolving Bayesian Networks (EBN) model is needed that can adjust itself automatically when the distribution of data changes. There are some works on learning BN model incrementally but they simply
assume all data is in memory and add new incoming data into existing data in each step. Finally, event streams often have high incoming rate, especially the event streams produced by wireless sensor networks. Predictive CEP is usually used to support online decision support system which need high performance even for large scale distributed event streams.

When used to handle the predictive CEP issue, the main weakness of the current predictive analytics methods is that they cannot get acceptable result due to the distribution drift of streaming event data. To address the challenges and overcome the weakness of current work, in this paper the authors propose a Predictive Complex Event Processing method based on Evolving Bayesian Networks (PCEP-EBN). The main contribution of this paper includes the following:

— The authors propose a predictive CEP framework based on BN model which uses Gaussian mixture model and EM algorithm for approximate inference.

— When learning and updating the BN structure, the authors propose incremental calculation methods for the score metric which support single-pass processing of event stream.

— The authors propose algorithms to support evolving BN structure based on hill-climbing method. The hill-climbing algorithm of this paper supports exploring multiple peaks on parallel and updating the score metric based on changed edges only. Evolving BN parameters is also supported using an incremental EM algorithm for Gaussian mixture model.

The authors evaluated the PCEP-EBN method in road traffic domain with both real application data and data produced by a simulated transportation system. The results show that PCEP-EBN is an effective method for predictive complex event processing and it outperforms other popular methods when processing traffic prediction in intelligent transportation systems.

2. Related work

2.1. Complex event processing and predictive analytics

CEP is a technique that recognizes complex events based on a set/sequence of occurrences of single events by continuously monitoring the event stream. Etzion et al. defined the basic concepts and architecture of complex event processing in their book [2]. The main component in CEP is Event Processing Agent (EPA) which can generate a set of complex events as output from a set of input events by applying some logic. Event Processing Network (EPN) is composed of a collection of EPAs, event producers, and event consumers linked by channels. EPN is used to define the event processing flow execution. Luckman introduced the event processing network first in the field of modeling [1]. Based on this idea, the conceptual model of EPN was further elaborated by Sharon and Etzion [6]. The key ideas of CEP have four steps: (1) Primitive events are extracted from large volume data stream; (2) Event correlation or event aggregation is detected to create business logic with event operators according to specific rules; (3) Primitive or composite events are processed to extract their time, causal, hierarchical and other semantic relationships; (4) High level complex events are sent to the subscribers which can be used for decision making.

CEP engine needs to process streams of events with time stamp and, therefore, numerous event pattern recognition methods are proposed based on Non-deterministic Finite Automaton (NFA). SASE [7] is a high performance query-plan based complex event processing method that uses NFA. Agrawal et al. proposed an improved pattern matching model and a query evaluation framework based on NFA to support more powerful event processing [8]. In their work a suite of techniques are proposed to improve runtime efficiency by exploiting sharing in storage and processing.

The basic CEP method detects event patterns from online event stream. Another type of CEP is predictive CEP which can predict future events before they have happened [2]. Etzion et al. believe predictive CEP and intelligent CEP are main parts of future directions in event processing technology. Currently most of the work on predictive CEP considers the task as a combine of CEP and predictive Analytics. Fülöp et al. proposed a conceptual framework of predictive CEP by combining CEP and predictive analytics [9]. In their framework, prediction models are trained using historical events. Online complex events produced by EPN are sent to predictive analytics component to predict future events. The predicted future events are then sent back to EPN for further step process. Tóth et al. proposed a framework that includes the predictive analytics technologies into CEP applications [10]. They extended a CEP solution with predictive capabilities, defined the key aspects of the combination of these techniques, and summarized how CEP and PA could gain from the joint solution.

Predictive analytics has been studied for many years and a series of models and algorithms are proposed. Some work [35,36] used optimized value methods to improve the results which is related to this paper. Recently, deep neural networks and BN are widely used and be more suitable for predictive CEP. Huang et al. proposed a predictive analytics method based on deep belief networks [11]. Their method uses deep belief networks in the bottom level and a regression model is used on the top level to support the final prediction. Multitask learning is supported in the top regression model to improve performance. In the work of Lv et al., a stacked autoencoder model is used to learn generic event stream features, and it is trained in a greedy layer-wise fashion [12]. Predictive analytics with deep learning model usually can achieve good accuracy, but the training of the model is complex and it prone to the problem of over fitting.

As a valid model for uncertain knowledge representation and inference, Bayesian Network is widely used in predictive analytics. Castillo et al. proposed a Bayesian network model for traffic flow prediction which is able to take into account the random character of the level of total mean flow and the variability of origin-destinations (OD) pair flows, together with the random violation of the balance equations for OD pairs [13]. This model provides the joint density of all unobserved variables and in particular the corresponding conditional and marginal densities, which allow not only joint predictions, but also probability intervals. Zhu et al. proposed a Bayesian network model for traffic prediction using prior link flows [14]. Their model sets link flows as parents of the OD flows. Using prior link flows, the prior distribution of all the variables is determined. To remove inconsistencies in OD matrices estimation and traffic assignment, a combined BN and stochastic user equilibrium model is proposed, in which the equilibrium solution is obtained through iterations.

2.2. Evolving systems and evolving Bayesian networks

Evolving systems are developed to address the distribution drifts in data stream. Angelov et al. proposed an evolving intelligent system which supports learning and adjusting model from data stream [30]. Recently many incremental machine learning methods [31,32] are proposed which are suitable for learning from data stream with distribution drifts. Another related work is evolving neural network which uses evolutionary algorithms to optimize the structure and parameters of neural networks [33,34].
When using BN, predictive analytics for event stream can be looked upon as the learning of BN with respect to these inherent characteristics of the sample data. Learning BN from data is the key technique for modeling uncertain knowledge, the basis of uncertainty inferences, and corresponding applications. Learning the BN structure from data is an NP-hard problem as the number of possible structures grows super-exponentially by the number of nodes. Approximate structure learning algorithms are usually categorized into two groups of constraint-based and search-and-score (S&S) approaches. Constraint-based algorithms rely on a number of conditional independency tests to determine whether two variables are independent given a set of conditioned variables. Constraint-based approaches are generally asymptotically correct and have a well-defined stopping criterion [15]. Being a more popular method, S&S approach involves a strategy for searching through the space of possible structures and a scoring function measuring the fitness of each structure to the data. The commonly used score metrics include Minimum Description Length (MDL) [16], Bayesian Information Criterion (BIC) [17], Bayesian Dirichlet equivalent (BDe) [18], etc. In order to address the problem of huge search space, most of the S&S methods use heuristic-based optimization or approximate searching algorithms. Some recently examples include hill climbing [19], simulated annealing [20] and Genetic algorithms [15].

The issue of distribution drift reflects a situation in which the event stream does not follow a fixed and predictable data distribution. In order to address the distribution drift in event stream, EBN is needed. EBN offers a flexible working framework which adapts to any variation in data trends or expansion of system states by incremental structure and parameter adaption. For abrupt drifts, a common idea for EBN is to learn different BN models and switch among models when data distribution is changed. Pascale et al. proposed an adaptive BN model for traffic flow prediction [4]. In their work, two major stationary areas are recognized as principal phases of traffic flows. BN models are learned for each phase and the adaptive method select BN model based on mutual information learning.

Incremental drifts are more difficult to be handled because it cannot be detected by global drift detection approaches in the structural learning scenario and cannot be addressed by the parameter learning scenario either. These drifts gradually interfere with current data distribution, which undermines the predictive accuracy more severely. Some researchers use sampling-based method for EBN. For example, Robinson et al. proposed a Markov chain Monte Carlo (MCMC) sampling algorithm for learning the structure of the model from time-series data under different assumptions [21]. Sampling methods allow the probability or importance of individual edges to be evaluated over all possible networks. However, since only the best network structure is needed in predictive CEP, heuristic search-based methods will typically find them more quickly. Yasin et al. proposed an incremental algorithm for BN structure learning which can deal with high dimensional domains [22]. Their algorithm learns local models by limiting search space and performs a constrained greedy hill-climbing search to obtain a global model. Yue et al. proposed a parallel and incremental approach for data-intensive learning of BNs from massive, distributed, and dynamically changing data by extending the classical scoring and search algorithm and using MapReduce [23]. They used MDL scoring metric and proposed a two-pass MapReduce-based algorithms for computing the required marginal probabilities and scoring the candidate graphical model from sample data. An extended hill-climbing algorithm is then used to obtain the optimal structure. Li et al. proposed a framework of incremental BN structure learning and a new evaluation criterion “ABIC” (Adopt Bayesian Information Criterion) based on the BIC metric [24]. Then a three phase algorithm is used to learn the BN structure. Compared to this paper, their methods are not designed for data stream and cannot process data in single-pass manner. Acharya et al. studied the issue of modeling causal relationships over event streams where data are unbounded. Their method infers causality by learning the temporal precedence relationships using an incremental temporal network construction algorithm called Incremental Hill-Climbing Monte Carlo [25]. Compared to this paper, the work of Acharya et al. did not considered how to calculate the score metric incrementally. Furthermore, their incremental hill-climbing algorithm that starts from only the highest local peak is prone to miss best result near other local peaks.

3. Predictive CEP with Bayesian Networks

3.1. System architecture and basic CEP

The system architecture is shown in Fig. 1. Primitive event streams generated by various event sources are processed by CEP engine. The BEPA means basic event processing agent which recognize complex event based on patterns from event stream. Multiple EPAs are connected by event channels (ECs) to create an EPN. The skeleton in PEPA is a statistical summary of the dataset which contains all information necessary to calculate the scores of the Bayesian network. The predictive complex events generated by BEPA are sent to the event consumer, which is usually decision support system, to make some decisions. Based on the decisions some actions can be executed to affect the event sources proactively to eliminate some unwanted states. The decision support system is beyond the scope of this paper.

![Fig. 1. The system architecture.](image-url)

**Definition 1 (primitive event).** A primitive event is an atomic occurrence of interest in time. A primitive event is represented as \(<A,T>\) where \(A\) is the set of attributes and \(T\) is the timestamp that the event occurs.

**Definition 2 (complex event).** A complex event is a combination of primitive events or complex events by some rule. A complex event is represented as \(<E,R,T>\) where \(E\) represents the elements that compose the complex event, \(R\) represents the rule of the combination, \(T\) represents the time span of the complex event.

For example, each read operation of RFID reader generates a primitive event. Event type is a specification for a set of events that have the same semantic intent and the same structure. An
event context is a named specification of conditions that groups event instances so that they can be processed in a related way. The main complex event patterns include ALL, ANY, COUNT, SEQ, etc. The detailed information about the concepts and patterns of CEP is described in [2]. The authors have developed a high-performance Distributed Probabilistic Complex Event Processing method (DPCEP) [26]. Based on NFA, parallel algorithm is designed to improve the performance for both single and distributed streams. A query plan based method is used to process hierarchical complex event from distributed event streams. Query plan optimization is proposed based on query optimization technology of probabilistic databases.

3.2. Bayesian network model for predictive CEP

Definition 3 (predictive complex event). A predictive complex event is a future event that can be predicted from historical event stream. A predictive complex event is represented as \( e_{ij,t} = (e_{ij1,t}, \ldots, e_{ij8,t}; \Theta) \) where \( t \) is time stamp, \( h() \) is the prediction function and \( \Theta \) is the set of parameters.

Definition 4 (Bayesian network, BN). A Bayesian network is represented as \( B=(G, \Theta) \) where \( G \) is a directed acyclic graph (DAG) that represent the BN structure and \( \Theta \) is the set of parameters. The DAG can be represented by \( G=(V,E) \) where \( V \) is the set of random variables which makes up the nodes of the network, and \( E \) is the set of directed edges connecting pairs of nodes.

An arrow from node \( X \) to node \( Y \) means that \( X \) has a direct influence on \( Y \) (\( X, Y \in V \) and \( X \neq Y \)). The parameters represent the probabilities of each random variable given its set of parents: \( \theta \) = \( \text{Pr}(x|X_{\text{pa}(x)}) \) where \( x \) is a random variable and \( X_{\text{pa}(x)} \) is the set of parents of \( x \).

\[ p(X) = \prod_{i,t} p(X_{i,t} | X_{pa(i,t)}) \]  \hspace{1cm} (1)

The conditional probability \( p(X_{i,t}|X_{pa(i,t)}) \) can be calculates as:

\[ p(X_{i,t}|X_{pa(i,t)}) = p(X_{i,t},X_{pa(i,t)}) / p(X_{pa(i,t)}) \]  \hspace{1cm} (2)

Since it is not easy to calculate the joint distribution \( p(x_{i},X_{pa(i)}) \), the authors use Gaussian Mixture Model (GMM) [27] to approximate it like the following:

\[ p(x_{i,t}, X_{pa(i,t)}) = \sum_{a=1}^{M} \pi_{ag}(x_{i,t},X_{pa(i,t)} | \mu_{ag}, \Sigma_{ag}) \]  \hspace{1cm} (3)

where \( M \) is the number of Gaussian models and \( \pi_{ag} (\mu_{ag}, \Sigma_{ag}) \) is the \( m \)-th Gaussian distribution with \( (N_p + 1) \times 1 \) vector of mean values \( \mu_{ag} \) and \( (N_p + 1) \times (N_p + 1) \) covariance matrix \( \Sigma_{ag} \). Using EM algorithm, the parameters \( \{\mu_{ag}, \Sigma_{ag}\} \) can be inferred from historical data [27]. Then the conditional distribution \( p(x_{i} | X_{pa(i)}) \) can be derived from \( p(x_{i},X_{pa(i)}) \) and the prediction \( \hat{x}_{i,t} \) can be calculated from \( X_{pa(i)} \) with minimum mean square error (MMSE) method.

4. Evolving Bayesian Network

4.1. Bayesian network structure learning

Definition 5 (evolving Bayesian network, EBN). An evolving Bayesian network is a Bayesian network whose structure and parameters are changing according to the drift of data distribution. An EBN is represented by \( (\mathcal{G}(t), \Theta(t)) \) where \( \mathcal{G}(t) \) and \( \Theta(t) \) are the structure and parameters at time \( t \) respectively.

The EBN includes evolving of BN structure and parameters. For structure evolving, since search-and-score method is more suitable for evolving than constraint-based method, the authors use a search-and-score method with BDE metric [18] score function. The BDE metric is defined as following:

\[ P(\mathcal{G}|D) = \prod_{i,j} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ijk})} \frac{\Gamma(\alpha_{ij} + N_{ijk})}{\Gamma(\alpha_{ij})} \]  \hspace{1cm} (4)

Where \( D \) is data set, \( q_{ij} \) is the number of configurations of the parent set \( pa_{ij} \), \( r_{ij} \) is the number of discrete states of variable \( x_{ij} \), \( N_{ijk} \) is the number of times \( x_{ij} \) took on the value \( k \) given the parent configuration \( j \). \( N_{ijk} = \sum_{j} N_{ijk} \) and \( a_{ijk} \) and \( b_{ijk} \) are Dirichlet hyper-parameters on various entries in the BN parameters, and \( \Gamma(v) \) is the Gamma function which for \( v \in \mathbb{N} \) is given by \( \Gamma(v) = (v-1)! \). In the BDE metric, \( b_{ijk} \) is set to \( a/(a+r_{ij}) \) which means parameter priors are all controlled by a single hyper-parameter \( a \).

BN structure learning is the task of selecting a BN structure \( G \) that explains a given data set \( D \). It has been shown that searching the huge space of DAGs for the optimal structure is a NP-hard problem. The BN model designed in this paper shown in Fig. 2 is a constrained DAG. When a directed edge from node \( (j,t) \) to node \( (i,t) \) is drawn, it must satisfies \( t_1 < t_2 \). Even the maximum value of the \( \Delta t = t_2 - t_1 \) is set, it is still a NP-hard problem. Therefore greedy local search method is used in this paper.

In order to calculate the BDE metric incrementally, an algorithm is proposed as shown in algorithm 1. In this algorithm the BN structure is assumed to be static. The idea behind this algorithm is that, when new data comes the metric is not needed to be calculated from scratch but "replace" the part that affected by new data, as shown in fig. 3.
In this paper the objective quality function is the BDe metric. It is also needed to define a set of traverse operators \( OP = \{ op^1, \ldots, op^t \} \) that can be executed at each step. Based on the property of the BN model in fig. 2, the authors define a simple but effective one edge neighborhood (OEN).

**Definition 6** (one edge neighborhood, OEN). An one edge neighborhood of a given model \( M \) is the set of all the alternative models that can be built from \( M \) through adding or removing one edge, \( \text{OEN}(M) = \{ M' | \text{op}(M') \in \text{op} \} \) (add an edge, remove an edge).

In algorithm 1 the BN structure is assumed to be static. In order to calculate the BDe metric incrementally with new edge when data is not changed, the authors use the following equation:

\[
\Delta BDe(u, v) = \frac{\Delta BDe(u, v)}{BDe(u, v)}
\]

The hill-climbing search algorithm (HCS) works in a step-by-step fashion to generate a model. At each step it makes the maximum possible improvement in an objective quality function.

Algorithm 1: Incremental calculation of BDe metric with new data (BN structure is not changed)

**Input:** Bayesian network structure \( G \), old data set \( D \), old metric \( \text{BDe}_{\text{old}} \), new data get \( \Delta D \)

**Output:** new metric \( \text{BDe}_{\text{new}} \)

1. \( \Delta \text{edge} \leftarrow \{ e \mid \text{edge } e \text{ is affected by } \Delta D \} \)
2. if \( |\Delta \text{edge}|/\text{EdgeNumber}(G) > 0 \) then
   3. calculate \( \text{BDe}_{\text{new}} \) directly from skeleton
4. return \( \text{BDe}_{\text{new}} \)
5. end if
6. \( \Delta p \leftarrow \{ i, j, k \mid N_{ij} \text{ is changed by } \Delta D \} \)
7. \( \Delta \text{BDe}_{\text{old}} \leftarrow 1, \Delta \text{BDe}_{\text{new}} \leftarrow 1 \)
8. for each pair \( (i, j) \) in \( \Delta p \)
   9. \( N_{ij} \leftarrow N_{ij} \) new
10. end for
11. for each \( k \) in \( \{i, j, k\} \) triple of \( \Delta p \)
12. \( N_{ijk} \leftarrow N_{ijk} \) new
13. end for
14. \( \Delta \text{BDe}_{\text{old}} \leftarrow \Delta \text{BDe}_{\text{old}} \cdot \Gamma(\alpha_{ij} + N_{ijk})/\Gamma(\alpha_{ij}) \)
15. \( \Delta \text{BDe}_{\text{new}} \leftarrow \Delta \text{BDe}_{\text{new}} \cdot \Gamma(\alpha_{ij} + N_{ijk})/\Gamma(\alpha_{ij}) \)
16. end for
17. \( \Delta \text{BDe}_{\text{old}} \leftarrow \Delta \text{BDe}_{\text{old}} \cdot \Gamma(\alpha_{ij})/\Gamma(\alpha_{ij} + N_{ij}) \)
18. \( \Delta \text{BDe}_{\text{new}} \leftarrow \Delta \text{BDe}_{\text{new}} \cdot \Gamma(\alpha_{ij})/\Gamma(\alpha_{ij} + N_{ij}) \)

In\( \Delta \text{BDe}_{\text{old}} \leftarrow 1, \Delta \text{BDe}_{\text{new}} \leftarrow 1 \) end if

19. update \( N_{ij} \) with \( N_{ij} \) new
20. end for
21. \( \text{BDe}_{\text{new}} \leftarrow \text{BDe}_{\text{old}} \cdot \Delta \text{BDe}_{\text{new}} / \Delta \text{BDe}_{\text{old}} \)
22. return \( \text{BDe}_{\text{new}} \)

The hill-climbing search algorithm (HCS) works in a step-by-step fashion to generate a model. At each step it makes the maximum possible improvement in an objective quality function.

4.2. Evolve Bayesian network structure

The authors use an evolving MMHC algorithm (EMMHIC) to evolve BN structure as shown in algorithm 3. The current BN structure \( G_{\text{cur}} \) has been learnt by the MMHC algorithm. Existing methods usually merge the new data into existing data which makes the system has bad scalability. The BDe value of the BN structure is updated using equation 5, and the skeleton is also updated. The EMMHC algorithm tries to find the optimized BN structure given an incremental data set \( \Delta D \). If the change of score when applying \( \Delta D \) is smaller than a threshold value \( \delta \), then the current BN structure need not be changed (lines 4 – 6). The BDe(\( \text{new} \), \( \Delta D \)) is calculated using algorithm 1. The EMMHC
algorithm creates k subtasks to incrementally search for
optimized BN structure starting from the structures with top-k
score in the localMaxList (lines 7 – 9). The subtasks are executed
parallel and the result with highest score is selected.

Algorithm 3: EMMHC
input: current BN structure \( G_{\text{curr}} \), statistic information in skeleton
\( I_{\text{curr}} \), incremental data set \( \Delta D \), localMaxList
output: new BN structure \( G_{\text{new}} \)
1. currentScore ← BDe(\( G_{\text{curr}} \))
2. \( G_{\text{new}} \leftarrow G_{\text{curr}} \)
3. if (BDe(\( G_{\text{new}} \), \( \Delta D \)) - currentScore / currentScore < \delta) then
   4. return \( G_{\text{new}} \)
5. end if
6. for each \( G \) in top-k of localMaxList
7. create a new subtask by calling LocalSearch with
   parameters \( G \), \( I_{\text{curr}} \), \( \Delta D \), currentScore
8. end for
9. run all sub tasks parallel and wait the result from all sub tasks
10. \( G_{\text{max}} \leftarrow \) the BN structure with the highest score in the sub tasks
11. if (BDe(\( G_{\text{max}} \)) > BDe(\( G_{\text{new}} \))) then
12. \( G_{\text{new}} \leftarrow G_{\text{max}} \)
13. end if
14. return \( G_{\text{new}} \)

The subtask LocalSearch is shown in algorithm 4. The idea
behind this algorithm is that, since only part of the graph is
affected by the new data, the algorithm only need to add or
remove edges inside the affected part to maximum the score.
This algorithm has two stages: expansion stage and contraction stage.
During the expansion stage, an edge is repeatedly added that
can maximum the score function until the score cannot be increased.
During the contraction stage, an edge is repeatedly removed that
can maximum the score function until the score begin to decrease.
In line 2, the getCandidateEdgesToAdd function gets the node
pairs that in affectedNodes but have no edge between them (edge
can be added according to the structure of fig. 2). In line 3, the
cCandidateEdgesToRemove function gets the edges that are in
current BN structure and affected by incremental data set \( \Delta D \).

Algorithm 4: LocalSearch
input: current BN structure \( G_{\text{curr}} \), statistic information in skeleton
\( I_{\text{curr}} \), incremental data set \( \Delta D \), current score value currentScore
output: new BN structure \( G_{\text{new}} \)
1. affectedNodes ← getAffectedNodes(\( \Delta D \))
2. candidateEdgesToAdd ← getCandidateEdgesToAdd
   (affectedNodes, \( G_{\text{curr}} \))
3. candidateEdgesToRemove ← getCandidateEdgesToRemove
   (affectedNodes, \( G_{\text{curr}} \))
4. expansionStage, contractionStage ← true
5. while (expansionStage)
6. maxScore ← the max score of the BN by adding an edge
   in candidateEdgesToAdd
7. if (maxScore > currentScore) then
   8. \( G_{\text{new}} \leftarrow \) the BN structure that get max score
9. else
10. expansionStage ← false
11. end if
12. end while
13. while (contractionStage)
14. maxScore ← the max score of the BN by removing an edge
   in candidateEdgesToRemove
15. if (maxScore < currentScore) then
16. \( G_{\text{new}} \leftarrow \) the BN structure that get max score
17. else
18. contractionStage ← false
19. end if
20. end while
21. return \( G_{\text{new}} \)

The total learning process is divided into phases \( p_0 \), \( p_1 \), \ldots , \( p_n \).
Phases \( p_0 \) is a static phase in which the BN structure is learned
using algorithm 2 from existing data. Phases \( p_1 \), \ldots , \( p_n \) are
dynamic phase in which BN structure is updated using algorithm
3 based on new incoming data.

4.3. Evolve Bayesian network parameters

Finally, the parameter evolving method is proposed. Since
GMM is used to approximate the BN inference, the key problem
is to update the GMM parameters. Based on the EM algorithm
for GMM [27], the BN parameters updating algorithm is shown
in algorithm 5. When using EM algorithm to learn the BN
parameters, the final values of parameters and intermediate
results are stored in memory. In order to update the parameters
of GMM, the current values of \( \pi_{\text{m}}, \mu_{\text{m}}, \Sigma_{\text{m}} \) are used to calculate
the distribution of latent variables from the updated data (line 4).
Then new \( \pi_{\text{m}}, \mu_{\text{m}}, \Sigma_{\text{m}} \) can be calculated using maximum
likelihood based on the distribution of latent variables and
parameters are updated (lines 5 – 6). If the parameters are not
changed obviously, the algorithm will terminate (line 7 – 9).
The algorithm repeats the steps for many times until the log
likelihood converges.

Algorithm 5: update BN parameters
input: current data set \( D \), incremental data set \( \Delta D \), current BN
parameters
output: new BN parameters
1. \( D \leftarrow \) removeExpiredData(\( D \))
2. \( D \leftarrow D + \Delta D \)
3. noChange ← true
4. use current parameters \( \pi_{\text{m}}, \mu_{\text{m}}, \Sigma_{\text{m}} \)
5. to calculate the distribution of latent variables based on \( D \)
6. calculate new \( \pi_{\text{m}}, \mu_{\text{m}}, \Sigma_{\text{m}} \)
7. if (\( \forall \pi_{\text{m}} \in [\pi_{\text{m}}, \mu_{\text{m}}, \Sigma_{\text{m}}] \), \( \text{Delta} \mu_{\text{m}}, \text{Delta} \Sigma_{\text{m}}, \text{Delta} \pi_{\text{m}}, \text{Delta} \Sigma_{\text{m}} \))
   then
   8. return \( [\pi_{\text{m}}, \mu_{\text{m}}, \Sigma_{\text{m}}] \)
9. noChange ← false
10. if (the log likelihood does not converges) then
11. goto 3
12. return \( [\pi_{\text{m}}, \mu_{\text{m}}, \Sigma_{\text{m}}] \)

5. Experimental Evaluations

5.1. Evaluations of EBN

The authors evaluated the EBN method separately since it is
the key part of this paper. The BN structure in Fig.2 is a special
structure and the authors can’t find real data with corresponding
result BN structures for accuracy evaluation. Therefore it is
reasonable to draw samples from a known BN, apply structure
learning on the synthentic data, and compare the learned structure
with the original one. The original BN structure contains 783
event types (nodes) and 1752 edges. Since the authors want to
evaluate an evolving BN model, sampling data from all phases is
needed. In the first step, data set are sampled from the original
BN structure. The BN structure is changed randomly at each
dynamic phase and an incremental data set (a data window) is
sampled based on the changes of BN structure. The time window
\( N_T \) is set to 15. In order to evaluate the accuracy of BN structure
learning and updating, the authors use the Structural Hamming Distance (SHD) [28]. The SHD is defined as the number of the
following operators required to make the BNs match: add or
delete an edge. This distance compares the learned and original
BN equivalence class structure without penalizing an algorithm.
for structural differences that cannot be statistically distinguished. Since the real BN structures of each phase are known, the SHD value can be calculated for each phase and the average value is used to evaluate the evolving model accuracy. In order to evaluate the running performance the mean of total learning time is used. A server with Intel Xeon E5 processor and 16GB memory is used in the experiments for EBN.

In the first experiment, the parameters $\delta$ and $\theta$ are evaluated. The sample number for phase $p_i$ is 50,000 and the data window for each dynamic phase contains 2,000 samples. The phase number $n$ is set to 30. The average SHD values and total running time values for different $\delta$ and $\theta$ are shown in Table 1. It can be found that when $\delta$ increases, the accuracy is decreased but the running time is increased. The reason is that when $\delta$ becomes larger, more small changes of BDe metric are ignored. From Table 1, it can also be found that the accuracy is almost not affected by $\theta$. The performance is increased when $\theta$ is increased from 0.1 to 0.4 but decreased when $\theta$ is changed from 0.4 to 0.8. Since $\theta$ is a threshold value that controls when the BDe metric should be calculated directly from skeleton, it has an optimized value. In the following experiments the authors set $\delta = 0.03$ and $\theta = 0.4$ (because this setting can get acceptable compromise between accuracy and running time).

Table 1. Average SHD values and total running time values for different $\delta$ and $\theta$. The values inside the parenthesis are total running time values (seconds).

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0.01</th>
<th>0.02</th>
<th>0.03</th>
<th>0.04</th>
<th>0.05</th>
<th>0.06</th>
<th>0.07</th>
<th>0.08</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>155</td>
<td>162</td>
<td>168</td>
<td>179</td>
<td>194</td>
<td>211</td>
<td>227</td>
<td>245</td>
</tr>
<tr>
<td></td>
<td>(83.5)</td>
<td>(79.1)</td>
<td>(77.5)</td>
<td>(75.2)</td>
<td>(72.3)</td>
<td>(68.8)</td>
<td>(64.6)</td>
<td>(52.5)</td>
</tr>
<tr>
<td>0.2</td>
<td>159</td>
<td>157</td>
<td>174</td>
<td>182</td>
<td>188</td>
<td>214</td>
<td>232</td>
<td>238</td>
</tr>
<tr>
<td></td>
<td>(80.9)</td>
<td>(77.5)</td>
<td>(76.2)</td>
<td>(74.8)</td>
<td>(72.4)</td>
<td>(69.1)</td>
<td>(63.9)</td>
<td>(47.2)</td>
</tr>
<tr>
<td>0.3</td>
<td>162</td>
<td>154</td>
<td>178</td>
<td>188</td>
<td>214</td>
<td>232</td>
<td>238</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>(78.3)</td>
<td>(74.8)</td>
<td>(73.3)</td>
<td>(72.4)</td>
<td>(69.1)</td>
<td>(63.9)</td>
<td>(61.2)</td>
<td>(47.2)</td>
</tr>
<tr>
<td>0.4</td>
<td>148</td>
<td>169</td>
<td>162</td>
<td>172</td>
<td>205</td>
<td>218</td>
<td>219</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>(77.3)</td>
<td>(72.2)</td>
<td>(71.9)</td>
<td>(70.0)</td>
<td>(67.7)</td>
<td>(62.8)</td>
<td>(61.8)</td>
<td>(46.3)</td>
</tr>
<tr>
<td>0.5</td>
<td>151</td>
<td>155</td>
<td>169</td>
<td>171</td>
<td>203</td>
<td>218</td>
<td>231</td>
<td>248</td>
</tr>
<tr>
<td></td>
<td>(77.5)</td>
<td>(75.7)</td>
<td>(74.6)</td>
<td>(72.2)</td>
<td>(68.4)</td>
<td>(64.4)</td>
<td>(63.9)</td>
<td>(48.8)</td>
</tr>
<tr>
<td>0.6</td>
<td>161</td>
<td>170</td>
<td>158</td>
<td>168</td>
<td>209</td>
<td>232</td>
<td>229</td>
<td>248</td>
</tr>
<tr>
<td></td>
<td>(81.2)</td>
<td>(79.9)</td>
<td>(77.7)</td>
<td>(72.8)</td>
<td>(69.9)</td>
<td>(62.0)</td>
<td>(66.1)</td>
<td>(52.2)</td>
</tr>
<tr>
<td>0.7</td>
<td>149</td>
<td>164</td>
<td>155</td>
<td>182</td>
<td>212</td>
<td>225</td>
<td>259</td>
<td>239</td>
</tr>
<tr>
<td></td>
<td>(86.1)</td>
<td>(84.8)</td>
<td>(82.4)</td>
<td>(78.4)</td>
<td>(73.2)</td>
<td>(69.6)</td>
<td>(67.4)</td>
<td>(54.9)</td>
</tr>
<tr>
<td>0.8</td>
<td>153</td>
<td>158</td>
<td>163</td>
<td>175</td>
<td>215</td>
<td>231</td>
<td>244</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>(89.2)</td>
<td>(87.6)</td>
<td>(84.3)</td>
<td>(80.3)</td>
<td>(75.3)</td>
<td>(69.9)</td>
<td>(58.8)</td>
<td></td>
</tr>
</tbody>
</table>

In the next experiment the authors compare the accuracy and performance of the EBN method of this paper with a typical sampling-based method nsDBN [21] and a recent search-based method iHCMC [25]. Constraints are added to the other two methods to make sure directed edges can only be added according to the structure of fig. 2. For nsDBN, the “Unknown Number and Unknown Times of Transitions (UNUT)” setting is used and the parameters are set as $\lambda_s=5$, $\lambda_n=1$. The dynamic phrases number $n$ is set to 30. The accuracy and performance of the three methods are evaluated for different window size and total data size. The window size $w$ is set to 1000, 2000, 3000, 4000 and 5000 at each time. The sample number in static phase $p_i$ is set to 50×$w$. The sample number of nsDBN is set to the same as total data size and 20% of the samples are used for burn-in. In iHCMC, the parameter K is set to 8.

The performance of the 3 methods with different window size and data size is shown in Fig. 5. It can be found that the performance of sampling method nsDBN is obviously worse than the other two methods. The running time of nsDBN increases almost linearly since the sampling increases linearly when the window size increases. If only best BN structure is needed (but not distribution of structures), search-based methods have better performance than sampling-based methods. However, sampling-based methods allow the probability or importance of individual edges to be evaluated over all possible networks. The running time of EBN and iHCMC also increases with window size since more calculations are needed to calculate the scores. However another important factor that affects the performance is how many times the score function is called. That factor is related to the BN structure and data set. It can also be found that the total performance of EBN outperforms iHCMC. The main reason is that EBN uses an incremental score calculation method. Although the structure updating algorithm of EBN searches from multiple local peaks, it searches parallel to improve the performance. From Fig. 4 and Fig. 5 the authors concluded that EBN outperforms the other methods in both accuracy and performance.

5.2. Evaluations of predictive CEP
In this section the experimental evaluation of the complete PCEP-EBN method using both real and simulated data is reported. The real application data comes from the PEMS traffic monitoring network in the area of the Highway 101 to Los Angeles. The real data is stored in files and a player is used to produce event stream from the files. The distribution of this data set is relatively simple. In order to evaluate more complex data distribution, the authors developed a traffic simulation system based on the road traffic simulation package SUMO [29]. The simulation system supports getting event data from the traffic network and executing actions. SUMO supports “induction loops” which can detect vehicles that pass corresponding areas. Getting the location of every vehicle at any time is also supported (this can be used to simulate GPS). The event manager is connected to SUMO through the TraCI (Traffic Control Interface) to get primitive events such as induction loop information and vehicle location information. In the simulation system, the authors set a road network of $15 \times 15$ intersections and 80 thousand vehicles. Every road has 2 lanes and every intersection has traffic lights. In order to simulate real traffic system, a set of rules are defined. Every vehicle has a home location and an office location. A vehicle $v_i$ runs between home and office with probability $p_i$. The vehicles also go to other places such as supermarket, hospital, etc., with corresponding probabilities. The data distribution is changed in different phases. A separate server is used to run the simulation system.

In the experiments, method of this paper is compared with Adaptive Dynamic Bayesian Network (ADBN) [4] and Deep Belief Network (DBN) [11] using both real data and simulated data. The performance is evaluated based on Mean Absolute Percentage Error (MAPE) which is defined by:

$$MAPE(y, y') = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - y'_i}{y_i} \right|$$

where $N$ is the data size, $y$ is the real value of traffic flow and $y'$ is the predicted value. Mean accuracy is calculate by $1 - MAPE$. In PCEP-EBN, the parameter $\lambda$ is evaluated and $\lambda = 0.03$ is selected for the following experiments. The total time span of PEMS data is 24 hours. The time window $\Delta T$ is set to 10 minutes and the predicting phase time span is set to 0.5 hour. In SUMO, 2 days traffic is simulated with a time span of 480 minutes. The time window $\Delta T$ is set to 2 minutes and the predicting phase time span is set to 10 minutes.

In order to decide the time window number $T_N$ of the model in fig. 2, different $T_N$ values are evaluated and the result is shown in fig. 6. The accuracy increases with the increment of $T_N$ until a threshold is reached. The reason is that, when $T_N$ is too small, some nodes that should affect the current node are missed in the model. When $T_N$ is too large, nodes that are too old will not affect the current node and they will be ignored in the learning process. In the following experiments, the authors set $T_N = 8$ for PEMS data and $T_N = 12$ for SUMO data.

The accuracy evaluation result for real data and simulated data is shown in fig. 7 and fig. 8. The curve of traffic flow change with time and it can roughly be found that PCEP-EBN outperforms other methods. In order to display the result more clearly, the MAPE of the 3 methods for real data and simulated data is shown in fig. 9 and fig. 10. The total percentage error is shown in table 2.

<table>
<thead>
<tr>
<th></th>
<th>ADBN</th>
<th>DBN</th>
<th>PCEP-EBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEMS data</td>
<td>15.95%</td>
<td>1.79%</td>
<td>8.12%</td>
</tr>
<tr>
<td>SUMO data</td>
<td>8.95%</td>
<td>4.59%</td>
<td>7.78%</td>
</tr>
</tbody>
</table>

Based on the total MAPE value and MAPE curve it can be found PCEP-EBN outperforms other methods obviously. The reason is that PCEP-EBN uses an evolving BN model which can adjust itself according to the change of data distribution. The ADBN model is an adaptive model which can cluster historical data, learn models from data clusters, and select appropriate model at run time. But this model cannot adjust it structure and parameters according to new data distribution. That is why the ADBN get worse accuracy than PCEP-EBN. Although the deep belief network method uses deep architecture to get better accuracy, it still cannot address the problem of distribution drift well. It can also be found that the advantage of PCEP-EBN is more obvious in SUMO data than in PEMS data. The reason is that more data distribution drift is simulated in SUMO data. From all experiments it can be found that PCEP-EBN is an effective method and it outperforms other popular methods when processing data with distribution drift. The percentage error is high at the begin of experiment for all methods because the denominator of MAPE is small at the begin.
Fig. 9. Percentage error of the three methods on PEMS data

Fig. 10. Percentage error of the three methods on SUMO data

6. Conclusion and Future Work

In this paper the authors propose a predictive complex event processing method based on evolving Bayesian networks. The Bayesian model has two dimensions: event type and time. The Gaussian mixture model and EM algorithm are used as approximate method to infer the Bayesian model. This method supports calculating score metric incrementally. Bayesian network structure and parameters evolving algorithms are proposed. The evaluation results in road traffic domain with both real application data and simulated data shows that this method is effective for predictive complex event processing and it outperforms other popular methods when processing traffic prediction in intelligent transportation systems. This method can also be used in many other areas, especially in proactive event processing systems which support executing some actions to avoid unwanted states.

The performance of the BN structure evolving method still needs to be improved. In the future, the authors will try to use some heuristic methods in the search algorithm. The authors will also consider using parallel searching to improve the performance. The performance of the parameter updating algorithm is also need to be improved.

Acknowledgments

This project is supported by the “Study of Proactive Complex Event Processing for Large-scale Internet of Things” project of National Natural Science Foundation of China (61371116).

References


