A Streaming Data Prediction Method based on Evolving Bayesian Network

Yongheng Wang[1(✉)], Guidan Chen[1], Zengwang Wang[1]

[1] College of Information Science and Electronic Engineering, Hunan University, Changsha, China 410082

wyh,chengd2015,wangzw@hnu.edu.cn

Abstract. In the Big Data era, large volumes of data are continuously and rapidly generated from sensor networks, social network, the Internet, etc. Learning knowledge from streaming Big Data is an important task since it can support online decision making. Prediction is one of the useful learning task but a fixed model usually does not work well because of the data distribution change over time. In this paper, we propose a streaming data prediction method based on evolving Bayesian network. The Bayesian network model is inferred based on Gaussian mixture model and EM algorithm. To support evolving model structure and parameters based on streaming data, an evolving hill-climbing algorithm is proposed which is based on incremental calculation of score metric when new data is arrived. The experimental evaluations show that this method is effective and it outperforms other popular methods for streaming data prediction.

Keywords: Streaming Data · Prediction · Evolving Bayesian Network

1 Introduction

In the Big Data era, large volumes of data are continuously and rapidly generated from sensor networks, social network, the Internet, etc. Recently streaming big data processing attracts more attention. The less time we use to make decision, the more value we can get from the whole processing. Therefore, learning knowledge from Big Data and making decision on line is a very important task. Predictive analytics is an important learning method since it can identify events before they have happened, so that some unwanted events can be eliminated or their effects mitigated. Many models and algorithms have been used in predictive analytics. In this paper, we use Bayesian networks model since it has rich mathematical basis and it can explain to the user how the system came to its conclusions.

Currently predictive analytics technology for streaming data has some challenges. First, in streaming data prediction the system can only process data on single-pass and cannot control over the order of samples that arrive over time. The second challenge is call “concept drift” which reflects a situation that the input and/or output concepts do not follow a fixed and predictable data distribution. A fixed model learned from

adfa, p. 1, 2011.

historical data may not handle the concept drift issue well. The last challenge is that, event streams often have high incoming rate. Streaming data prediction is usually used to support online decision support system which need high performance even for large scale distributed event streams.

To address these challenges, in this paper we propose a Streaming Data Processing method based on Evolving Bayesian Networks (SDP-EBN), which supports single-pass processing of streaming data and uses evolving hill-climbing algorithm to support model structure evolving.

2 Related work

Predictive analytics has been studied for many years and a series of models and algorithms are proposed. Among these models, deep neural networks and Bayesian network are more popular for streaming data. Huang et al. proposed a predictive analytics method based on deep belief networks [1]. 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. Zhu et al. proposed a Bayesian network model for traffic prediction using prior link flows [2]. Evolving systems are developed to address the concept drifts in data stream. Angelov et al. proposed an evolving intelligent system which supports learning and adjusting model from data stream [3]. Recently many incremental machine learning methods [4] are proposed which are suitable for learning from data stream with concept drifts. Another related work is evolving neural network which uses evolutionary algorithms to optimize the structure and parameters of neural networks [5].

Learning Bayesian network 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. The commonly used score metrics include Bayesian Information Criterion (BIC) [6] and Bayesian Dirichlet equivalent (BDe) [7], etc.

To address the concept drifts in streaming data, evolving Bayesian network is needed. For abrupt drifts, a common idea for evolving Bayesian network is to learn different models and switch among models when data distribution is changed [8]. Incremental drifts are more difficult to be handled. Some researchers use sampling-based method for evolving Bayesian network [9]. However, since we only need best network structures in streaming data prediction, heuristic search-based methods [10, 11] will typically find them more quickly. Compared to the related work, our work considers how to calculate the score metric incrementally so that the model does not need to be trained from the scratch. Furthermore, the current hill-climbing algorithms that starts from only the highest local peak is prone to miss best result near other local peaks. Our algorithm searches top-k peaks parallel which can get better accuracy without losing much performance.

3 The SDP-EBN method

3.1 Bayesian network model for streaming data prediction

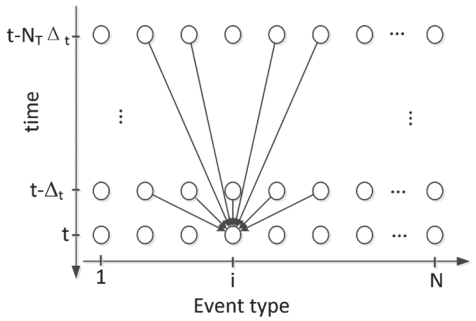

Fig. 1. The streaming data and Bayesian network structure

The BN model for streaming data prediction is shown in fig. 1 which has two dimensions: event type and time. The event type is related to some attribute of objects, e.g., the position of objects, or some state of the system, e.g., the traffic flow of roads. Assume the event type number is N, in a prediction task the state of node (i,t) is related to a set of states before time t (parent nodes). Let $x_{i,t}$ represent the state variable of node (i,t) and $pa(i,t)$ represent the set of parent nodes of (i,t). N_P denotes the number of nodes in $pa(i,t)$. $N_T\Delta_t$ is the time window which means we only consider nodes from time $t\text{-}N_T\Delta_t$ to $t\text{-}\Delta_t$ as the parents of nodes in time t. The set of state variables for $pa(i,t)$ is $X_{pa(i,t)}=\{x_{j,s} \mid (j,s) \in pa(i,t)\}$. According to the BN theory, the joint distribution of all nodes in the network can be represented as:

$$p(X)=\prod_{i,t} p(X_{i,t}\mid X_{pa(i,t)}) \tag{1}$$

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

$$p(X_{i,t}\mid X_{pa(i,t)})=p(X_{i,t},X_{pa(i,t)})/X_{pa(i,t)} \tag{2}$$

Since it is not easy to calculate the joint distribution $p(x_{i,t},X_{pa(i,t)})$, we use Gaussian Mixture Model (GMM) [12] to approximate it like the following:

$$p(X_{i,t},\ X_{pa(i,t)}) = \sum_{m=1}^{M} \pi_m g_m(X_{i,t},X_{pa(i,t)}\mid \mu_m,\Sigma_m) \tag{3}$$

where M is the number of Gaussian models and $g_m(\cdot|\mu_m,\Sigma_m)$ is the m-th Gaussian distribution with $(N_P + 1) \times 1$ vector of mean values μ_m and $(N_P + 1) \times (N_P + 1)$ covariance matrix Σ_m. Using EM algorithm we can infer the parameters $\{\pi_m,\mu_m,\Sigma_m\}_{m=1}^{M}$ from historical data [12]. Then the conditional distribution $p(x_{i,t}|\ X_{pa(i,t)})$ can be derived from $p(x_{i,t},\ X_{pa(i,t)})$ and the prediction $\hat{x}_{i,t}$ can be calculated from $X_{pa(i,t)}$ with minimum mean square error (MMSE) method. In SDP-EBN, the GMM parameters can be updated based on EM algorithm. Therefore, we mainly consider how to evolve the structure of the Bayesian networks.

3.2 Bayesian network structure learning

Since search-based methods are more suitable for evolving than constraint-based methods, we use a search-and-score method with BIC metric [6] score function. The BIC metric is defined as following:

$$g_{BIC}(G:D) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} m_{ijk}\log\theta_{ijk} - \sum_{i=1}^{n}\frac{q_i(r_i-1)}{2}\log m \qquad (4)$$

where D is the samples, m=|D|, q_i=| $X_{pa(i)}$|, r_i is the number of discrete states of x_i, m_{ijk} represents the times in D that x_i is k when the parent of x_i is j, $m_{ij} = \sum_{k=1}^{r_i} m_{ijk}$, $\theta_{ijk}=m_{ijk}/m_{ij}$, satisfying constraints $0\le\theta_{ijk}\le 1$, $\sum_{k=1}^{r_i}\theta_{ijk} = 1$, n is the number of variables.

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 graph space for the optimal structure is a NP-hard problem. Therefore, greedy local search method is used in this paper. To calculate the BIC metric incrementally, an algorithm is proposed as shown in algorithm 1.

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

Input: BN structure G, old data set D, new data get ΔD
Output: new metric BIC_new

```
Δedge ←{e | edge e is affected by ΔD}
if |Δedge|/EdgeNumber(G) > θ) then
    calculate BIC_new directly from skeleton
    Return BIC_new
Δp ← {i,j,k | m_ijk is changed by ΔD}
Δli ← 0 //li means the likelihood part of BIC
for each pair (i,j) in Δp
|   m_ij_new ← 0
|   for each k in (i,j,k) triple of Δp
|       m_ij_new ←m_ij_new + m_ijk
|   for each k in (i,j,k) triple of Δp
|       Δli ← m_ijk · log(m_ijk/m_ij_new) - m_ijk_old · log(m_ijk_old/m_ij_old)
|   update m_ij with m_ij_new
end for
li_new ← li_old + Δli
BIC_new ← li_new – penalty_old
return BIC_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. In our work the objective quality function is the BIC metric. Based on the property of the BN model in fig. 1, we define a simple but effective one edge neighborhood (OEN). An OEN 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, OEN(*M*)={*M'*|*M'*=op(*M*) ∧ op∈ {add an edge, remove an edge}}.

In algorithm 1 we assume the BN structure is static. In order to calculate the BIC metric incrementally with one edge change when data is not changed, we use the following equation:

$$BIC_{new} = BIC_{old} + \sum_{k=1}^{r_v} m_{uvk} log\theta_{uvk} - \frac{q_v(r_v-1)}{2} logm \quad (5a)$$

$$BIC_{new} = BIC_{old} - \sum_{k=1}^{r_v} m_{uvk\_old} log\theta_{uvk\_old} + \frac{q_{v\_old}(r_{v\_old}-1)}{2} logm \quad (5b)$$

where (5a) is for adding an edge and (5b) is for removing an edge.

To generate BN structure from data D without considering new incoming data, we use a max-min hill-climbing (MMHC) algorithm.

3.3 Evolve Bayesian network structure

We use an evolving MMHC algorithm (EMMHC) to evolve BN structure as shown in algorithm 2. The current BN structure G_{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 BIC score 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 give an incremental data set ΔD. If the change of score when applying ΔD is smaller than a threshold value δ, then the current BN structure need not be changed (lines 3 – 4). The BIC(G_{new}, ΔD) 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 localMinList (lines 5 – 6). The subtasks are executed parallel and the result with lowest score is selected.

Algorithm 2: EMMHC

input: current BN structure G_{cur}, statistic information in skeleton I_{stat}, incremental data set ΔD, localMinList
output: new BN structure G_{new}

```
1   currentScore ←BIC(Gcur)
2   Gnew ← Gcur
3   if (|BIC(Gnew, ΔD) - currentScore |/currentScore < δ) then
4      return Gnew
5   for each G in top-k of localMinList
6      create a new subtask by calling LocalSearch with parameters G, Istat,
         ΔD, currentScore
7   run all sub tasks parallel and wait the result from all sub tasks
8   Gmin ← the BN structure with the lowest score in the sub tasks
9   if (BIC(Gmin) < BIC(Gnew)) then
10     Gnew ← Gmin
11  return Gnew
```

The subtask LocalSearch is shown in algorithm 3. The idea behind this algorithm is that, since only part of the graph is affected by the new data, we only need to add or remove edges inside the affected part to minimum the score. This algorithm has two stages: expansion stage and contraction stage. During the expansion stage, an edge is repeatedly added that can minimum the score function until the score cannot be decreased. During the contraction stage, an edge is repeatedly removed if the score function is not increased. 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 getCandidateEdgesToRemove function gets the edges that are in current BN structure and affected by incremental data set ΔD.

Algorithm 3: LocalSearch

input: current BN structure G_{cur}, statistic information in skeleton I_{stat}, incremental data set ΔD, current score value currentScore
output: new BN structure G_{new}

```
1   affectedNodes ← getAffectedNodes(ΔD)
2   candidateEdgesToAdd ← getCandidateEdgesToAdd(affectedNodes, Gcur)
3   candidateEdgesToRemove ← getCandidateEdgesToRemove (affectedNodes, Gcur)
4   expansionStage, contractionStage ← true
5   while(expansionStage)
6   |   minScore ← the min score of the BN by adding an edge in
    |       candidateEdgesToAdd
7   |   if (minScore < currentScore)
8   |       Gnew ← the BN structure that get main score
9   |   else
10  |       expansionStage ← false
11  end while
12  while(contractionStage)
13  |   minScore ← the min score of the BN by removinging an edge in
14  |       candidateEdgesToRemove
15  |   if  minScore ⊁ currentScore then
16  |       Gnew ← the BN structure that get min score
17  |   else
18  |       contractionStage ← false
19  end while
20  return Gnew
```

4 Experimental Evaluations

We evaluated the EBN method separately since it is the key part of this paper. We draw samples from a known BN, apply structure learning on the synthetic data, and compare the learned structure with the original one. Our original BN structure contains 783 event types (nodes) and 1752 edges. The time window N_T is set to 15. In order to evaluate the accuracy of BN structure learning and updating, we use the Structural Hamming Distance (SHD) [13]. Servers with 2 Xeon E3 processors and 16GB memory is used in the experiments for EBN.

We first evaluate the parameters δ and θ. The sample number for phase p_0 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 δ and θ are shown in table 1. We can see when δ increases, the accuracy is decreased but the running time is also decreased. From table 1 we can also find the accuracy is almost not affected by θ.

In the next experiment, we compare the accuracy and performance of our EBN method with a typical sampling-based method nsDBN [9] and a recent search-based

method iHCMC [11]. We evaluated the accuracy and performance of the three methods for different window size and total data size. The accuracy evaluation result of the 3 methods with different window size and data size is shown in Fig. 2. We also evaluated the performance of the 3 methods with different window size and the result is shown in Fig. 3.

Table 1. Average SHD values and total running time values for different δ and θ. The values inside the parenthesis are total running time values (seconds)

θ \ δ	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
0.1	157 (83.4)	161 (79.3)	167 (77.6)	178 (75.4)	195 (72.6)	213 (68.7)	226 (64.4)	245 (52.5)
0.2	159 (80.6)	157 (77.5)	174 (76.3)	187 (74.4)	192 (70.2)	205 (66.6)	230 (63.1)	239 (49.7)
0.3	162 (78.4)	154 (74.4)	178 (73.3)	188 (72.4)	188 (69.1)	214 (63.9)	232 (61.2)	238 (47.2)
0.4	148 (77.7)	169 (72.3)	162 (71.9)	172 (70.0)	205 (67.7)	218 (62.8)	219 (61.8)	244 (46.3)
0.5	151 (77.5)	155 (75.7)	168 (74.5)	171 (72.2)	191 (68.6)	203 (64.4)	218 (63.9)	251 (47.8)
0.6	161 (81.8)	170 (79.9)	158 (77.7)	168 (72.8)	206 (69.9)	208 (67.3)	223 (66.1)	248 (52.7)
0.7	149 (86.2)	165 (84.9)	155 (82.4)	182 (78.4)	191 (73.2)	213 (69.7)	225 (67.4)	239 (54.9)
0.8	153 (89.2)	158 (87.6)	163 (84.3)	175 (80.3)	187 (75.3)	215 (73.3)	233 (69.8)	245 (58.9)

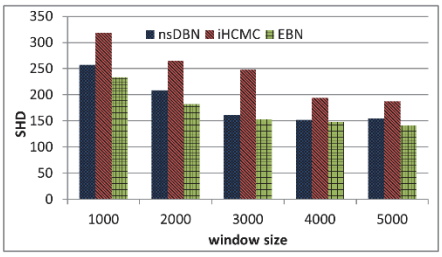

Fig. 2. The accuracy of 3 methods with different window size and data size. Dynamic phase number n is fixed at 30.

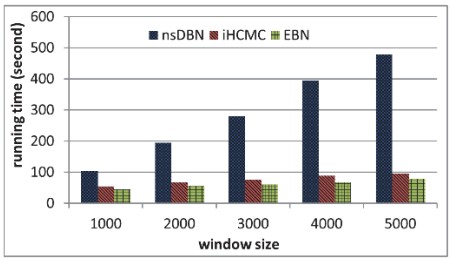

Fig. 3. The performance of 3 methods with different window size and data size

We evaluated the complete SDP-EBN method using both real and simulated data. The real application data comes from the PEMS traffic monitoring network*. We also developed a traffic simulation system based on the road traffic simulation package SUMO [14]. Our method is compared with Adaptive Dynamic Bayesian Network (ADBN) [8] and Deep Belief Network (DBN) [1]. The accuracy evaluation result is shown in fig.4 and fig. 5. From all experiments, we can see SDP-EBN outperforms other popular methods when processing data with distribution drift.

* PeMS project, https://pems.eecs.berkeley.edu/

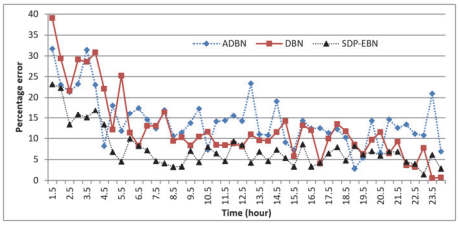

Fig. 4. Percentage error of the three methods on PEMS data

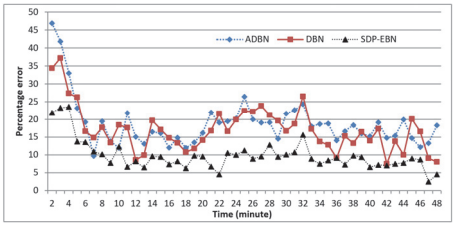

Fig. 5. Percentage error of the three methods on SUMO data

Acknowledgments. This work is supported by the National Natural Science Foundation of China (No.61371116).

References

1. W. Huang, G. Song, H. Hong, et al. Deep Architecture for Traffic Flow Prediction - Deep Belief Networks With Multitask Learning. IEEE Transactions on Intelligent Transportation Systems, 2014, 15 (5), pp. 2191 – 2201.
2. S. Zhu, L. Cheng, Z. Chu. A Bayesian network model for traffic flow estimation using prior link flows. Journal of Southeast University (English Edition), Vol.29, No.3, 2013, pp.322-327.
3. P. Angelov. Autonomous Learning Systems: From Data Streams to Knowledge in Real-time. Wiley Press,2013.1.
4. T. Li. PICKT: A solution for big data analysis. Proc. of the 10th International Conference on Rough Sets and Knowledge Technology, Tianjin, China, 2015,pp.15-25.
5. B. Yevgeniy, V. Olena, P. Iryna, et al. Fast learning algorithm for deep evolving GMDH-SVM neural network in data stream mining tasks. Proc. of the IEEE First International Conference on Data Stream Mining & Processing, Lviv, Ukraine, 2016, pp. 257-262.
6. Y. M. Shtarkov. Universal sequential coding of single messages. Problems of Information Transmission, 1987, 23(3), pp. 3–17.
7. D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. Machine Learning, 1995, 20(3), pp. 197–243.
8. A. Pascale and M. Nicoli. Adaptive Bayesian network for traffic flow prediction. In Proc. of the Statistical Signal Processing Workshop (SSP), 2011 IEEE, pp.177-180.
9. J. W. Robinson, A. J. Hartemink. Learning Non-Stationary Dynamic Bayesian Networks. Journal of Machine Learning Research, 2010, vol 11, pp.3647-3680.
10. K. Yue, Q. Fang, X. Wang, et al. A parallel and incremental approach for data-intensive learning of Bayesian networks. IEEE Transactions on Cybernetics, 45(12), 2015, pp.2890-2904.
11. S. Acharya, B. Lee. Causal network construction over event streams. Information Sciences, Vol 261, issue 2014, pp.32-51.
12. C. Bishop. Pattern recognition and machine learning (2nd edition). Springer 2010.
13. I. Tsamardinos, L. E. Brown, C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning, 65(1), 2006, pp.31-78.
14. M. Behrisch, L. Bieker, J. Erdmann, et al. Sumo - simulation of urban mobility: An overview. In Proc. of the third international conference on advances in system simulation (SIMUL 2011), Barcelona, Spain, October 23, 2011, pp.63–68.