Journal of Network and Computer Applications 76 (2016) 16–22

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Traffic matrix prediction and estimation based on deep learning in large-scale IP backbone networks

Laisen Nie[a], Dingde Jiang[a,*], Lei Guo[a], Shui Yu[b]

[a] School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
[b] School of Information Technology, Deakin University, Burwood, VIC 3125, Australia

ARTICLE INFO

Keywords:
Network traffic prediction
Network traffic estimation
Traffic matrix
Deep learning
Deep belief network

ABSTRACT

Network traffic analysis has been one of the most crucial techniques for preserving a large-scale IP backbone network. Despite its importance, large-scale network traffic monitoring techniques suffer from some technical and mercantile issues to obtain precise network traffic data. Though the network traffic estimation method has been the most prevalent technique for acquiring network traffic, it still has a great number of problems that need solving. With the development of the scale of our networks, the level of the ill-posed property of the network traffic estimation problem is more deteriorated. Besides, the statistical features of network traffic have changed greatly in terms of current network architectures and applications. Motivated by that, in this paper, we propose a network traffic prediction and estimation method respectively. We first use a deep learning architecture to explore the dynamic properties of network traffic, and then propose a novel network traffic prediction approach based on a deep belief network. We further propose a network traffic estimation method utilizing the deep belief network via link counts and routing information. We validate the effectiveness of our methodologies by real data sets from the Abilene and GÉANT backbone networks.

1. Introduction

Network traffic information is a crucial configuration input for network management and planning (Svigelj et al., 2015; Mardani and Giannakis, 2013) in a large-scale IP backbone network. It can be represented by a traffic matrix that reflects the volume of traffic flows between all possible pairs of original and destination (OD) nodes (Mardani and Giannakis, 2013). According to various network management tasks (i.e., load balancing and routing protocols configuration), kinds of granularities of traffic matrices are needed. For instance, the original and destination endpoints can be prefixes, links, routers, Point and Presence (Zhang et al., 2003). If we denote a traffic matrix by an N-by-T matrix X, each row of X represents the mean amount of the network traffic for an OD pair over a certain timeslot. Despite of its importance for a traffic matrix, network traffic monitoring techniques suffer from some technical and mercantile issues, thus we cannot obtain it easily and directly (Soule et al., 2005). Therefore, network operators try to estimate the traffic matrix from other readily available network information instead of measuring it directly.

The network tomography method is a state-of-the-art technique to obtain a traffic matrix, where link counts and routing information are used to infer it. The network tomography model in traffic matrix estimation problem has an explicit ill-posed feature (Polverini et al., 2015). Hence, lots of literatures refer to some additional information to accurately estimate the traffic matrix X (Soule et al., 2005). The most popular prior methods consist of the tomogravity method (Zhang et al., 2003), the principal component analysis (PCA) method, the route change method (Soule et al., 2005), etc. Though these methods have been the most current techniques for acquiring network traffic, a great number of problems still need to be solved. First, with the development of the scale of our networks, the level of the ill-posed feature of the network traffic estimation problem is much more deteriorated (Polverini et al., 2015; Hu et al., 2015; Xu et al., 2014). Besides, the statistical features of network traffic have changed greatly in terms of current network architectures and applications. For instance, the known statistical features of network traffic are self-similarity, long-range dependence, heavy-tailed distribution and so on (Xu et al., 2014; Niu and Tian, 2014). These new statistical features, which are insufficiently modeled by Poisson and Gaussian models, result in a poor estimator for current network traffic information (Soule et al., 2005). Motivated by that, we propose a network traffic prediction and estimation method respectively in this paper.

The main contributions of this paper are duple.

* Corresponding author.
E-mail address: jiangdd@mail.neu.edu.cn (D. Jiang).

http://dx.doi.org/10.1016/j.jnca.2016.10.006
Received 24 December 2015; Received in revised form 8 July 2016; Accepted 11 October 2016
Available online 13 October 2016
1084-8045/

- We first use a deep learning method to explore the dynamic features of network traffic. This architecture can deeply excavate mutual dependence among the traffic entries in various timeslots. Then we propose a novel network traffic prediction approach based on a deep belief network (DBN).
- We also try to use the deep belief network to solve the network tomography model. In our method, the deep belief network is used to learn the ill-posed inverse inference system. By training the deep belief network, we are able to estimate the traffic matrix accurately.

The rest of this paper is organized as follows. We first describe the related problems of the traffic matrix prediction and estimation in Section 2. After that, we introduce the related work about network traffic prediction and estimation fields and analyze the main challenges in Section 3. Section 4 introduces the deep learning theory and the deep belief network architecture. In Sections 5 and 6, we will propose the traffic matrix prediction and estimation methods, respectively. We will assess our methodologies in Section 7, and then we conclude our work in Section 8.

2. Problem statement

2.1. Traffic matrix prediction

Many network management operations rely on traffic flow prediction techniques, such as network planning and configuring network routing policies (Mardani and Giannakis, 2013; Soule et al., 2005). The essential destination of traffic matrix prediction is obtaining a predictor of the future traffic flow via its prior measurements (Soule et al., 2005; Kumlu and Hokelek, 2015). If we denote an entry of the N-by-T traffic matrix X by $x_{n,t}$, and then the traffic matrix prediction problem is defined as solving the estimator of $x_{n,t}$ (denoted by $\hat{x}_{n,t}$) via a series of historical and measured traffic data set ($x_{n,t-1}, x_{n,t-2}, x_{n,t-3}, \ldots$). The main challenge here is how to extract (or model) the inherent relationships among the traffic data set so that one can exactly predict $x_{n,t}$.

2.2. Traffic matrix estimation

The fundamental problem to estimating traffic matrices by network tomography model is solving an inverse problem that built by link counts, routing information and the traffic matrix (Zhao and Tan, 2015). If we denote link counts and the routing information by Y and A, then the network tomography model is

$$Y = AX. \tag{1}$$

The matrix A is the so-called routing matrix. We assume that all packets in an OD pair are transferred in the same path, and then the entries of the routing matrix A are 1 or 0. Each column of A represents the path of the relative OD pair (Zhang et al., 2003). The network tomography model exposes the relationship among link counts, routing matrix and traffic matrix. If a large-scale IP backbone network is built by K nodes and L links, the matrices Y, A and X are L-by-T, L-by-N ($N = K^2$) and N-by-T, respectively. Generally, link counts and routing information are obtainable for us. Link counts can be monitored by the simple network management protocol (SNMP) (Soule et al., 2005). SNMP is universally employed on many IP network devices. Each device employed the SNMP has a cyclic counter that is able to record the number of the bytes passing its interface. Then, without other devices, we can obtain the link count data using an SNMP poller that can circularly request the SNMP Management Information Base data by these devices (Zhang et al., 2003). The routing matrix A can be achieved from the configuration information of routers. As mentioned above, this inverse inference problem has a highly ill-posed feature, because the number of OD pairs in a large-scale IP backbone network is much larger than that of links. Hence, the main challenge to estimating traffic matrices is solving this inverse inference problem with highly ill-posed feature (Soule et al., 2005).

3. Related work

Statistical model techniques are usually performed in both traffic matrix prediction and estimation. Originally, researchers refer to some simple statistical models as add information to handle with the ill-posed feature of the network tomography system in traffic matrix estimation, typically, the Gaussian and Poisson models (Soule et al., 2005). The authors in Kumlu and Hokelek (2015) model traffic flows as a higher order Markov process, and join the Incremental Gaussian Mixture Model to estimate a traffic flow. Considering the various constructed features of a traffic matrix (e.g., spatio-temporal and low-rank features), many novel methods are proposed. For instance, in Soule et al. (2005), the authors use the PCA method to acquire an approximated expression of the traffic matrix for decreasing the level of the ill-posed feature. The authors in Roughan et al. (2012) consider the spatio-temporal feature of a traffic matrix, and propose a novel compressive sensing framework (termed as Sparsity Regularized Matrix Factorization) to estimate it. In Xu et al. (2014), the authors propose a probability model to estimate the Peer-to-Peer traffic matrix. Based on the PCA method, the authors in Zhao and Tan (2015) propose a novel traffic matrix estimation method by minimizing the Mahalanobis distance.

In fact, the methods of traffic matrix prediction and estimation have an inherent and implicit relationship. Most of traffic matrix estimation methods try to obtain a predictor of the traffic matrix at first, and then calculate an accurate estimator of the traffic matrix by recalibrating the initial predictor (Soule et al., 2005). Hence, many traffic matrix estimation algorithms can perform traffic matrix prediction, e.g., the PCA method (Soule et al., 2005) and the SRMF method (Roughan et al., 2012) mentioned in the above. The PCA method predicts a traffic matrix by learning the architecture feature of this traffic matrix. In SRMF estimation model, the authors define a temporal constraint matrix to describe the knowledge about the spatio-temporal structure of a traffic matrix. By choosing an appropriate architecture for the temporal constraint matrix, the SRMF model can carry out traffic matrix prediction (Roughan et al., 2012).

With the progress of current network applications, the network traffic flows are much more complex than before. The traffic flows show much more various and multiple statistical features (Polverini et al., 2015; Hu et al., 2015; Xu et al., 2014). Motivated by that, in this paper, we use the deep learning theory (Hinton et al., 2006; Bengio, 2009; Deng and Yu, 2014; Hinton and Salakhutdinov, 2006; Chen et al., 2015; Huang et al., 2014) to explore the connotative features of traffic flows, and propose a prediction method and an estimation method to the traffic matrix respectively.

4. Deep learning and deep belief networks

Deep learning is an expanded technique of the shallow learning (i.e., artificial neural networks), which is a machine learning paradigm used to learn deep hierarchical models of data (Hinton et al., 2006). The deep belief network is one of the most prominent deep learning primitives (Bengio, 2009; Deng and Yu, 2014). It is built by a series of Restricted Boltzmann Machines (RBMs) whose architecture is shown in Fig. 1. We see that the RBM is an undirected graphical model that consists of visible and hidden layers (denoted by v and h) (Bengio, 2009). Each visible unit (hidden unit) is connected to all hidden units (visible units). The peer units in the same layer are disconnected with each other. In an RBM, all values of v and h are stochastic variables, typically, Gaussian and Bernoulli distribution (Deng and Yu, 2014). A DBN is the stack of a number of RBMs (see Fig. 2), that is, the hidden layer of the low layer is the visible layer of the high layer. For an RBM model, a jointly probability distribution function over visible and hidden units is defined as

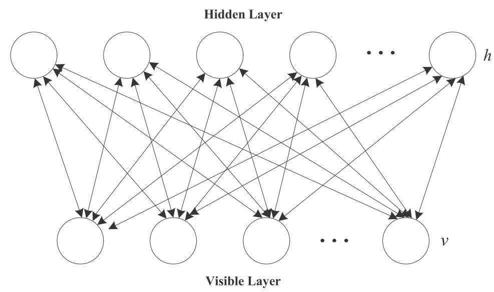

Fig. 1. Restricted Boltzmann machine.

$$p(v, h) = \frac{\exp(-E(v, h))}{\sum_{v,h} \exp(-E(v, h))}, \tag{2}$$

where $E(v, h)$ is termed as the energy function (Deng and Yu, 2014). According to various probability distributions of visible units, the definition of the energy function $E(v, h)$ is diverse. The common distributions of visible units are Gaussian and Bernoulli. When the probability distributions of visible units and hidden are Gaussian and Bernoulli, respectively, the energy function is

$$E(v, h) = -\frac{1}{2}\sum_{i=1}^{I} (b_i - v_i)^2 - \sum_{j=1}^{J} a_j h_j - \sum_{i=1}^{I}\sum_{j=1}^{J} w_{i,j} v_i h_j, \tag{3}$$

where I and J are the numbers of visible and hidden units, respectively (Deng and Yu, 2014). b_i and a_j are the biases of visible and hidden units. Relatively, when the probability distributions of visible units are Bernoulli (the hidden units are consistently Bernoulli), the energy function is

$$E(v, h) = -\sum_{i=1}^{I} b_i v_i - \sum_{j=1}^{J} a_j h_j - \sum_{i=1}^{I}\sum_{j=1}^{J} w_{i,j} v_i h_j. \tag{4}$$

Based on the above definitions, if the visible units are Gaussian or Bernoulli, then the conditional probabilities for hidden units is consistent, which can be defined as

$$P(h_j = 1|v) = sigm\left(a_j + \sum_{i=1}^{I} w_{i,j} v_i\right), \tag{5}$$

where $sigm(z) = \exp(z)/(1 + \exp(z))$ is the sigmoid function (Deng and Yu, 2014). Similarly, consider the conditional probabilities for visible units, if the visible units are Gaussian, it can be calculated by

$$P(v_i|h) = N\left(b_i + \sum_{j=1}^{J} w_{i,j} h_j, 1\right), \tag{6}$$

where $N\left(b_i + \sum_{j=1}^{J} w_{i,j} h_j, 1\right)$ denotes the Gaussian distribution whose mean and variance are $b_i + \sum_{j=1}^{J} w_{i,j} h_j$ and 1 (Deng and Yu, 2014). When the visible units are Bernoulli, the conditional probabilities for visible units are

$$P(v_i = 1|h) = sigm\left(b_i + \sum_{j=1}^{J} w_{i,j} h_j\right). \tag{7}$$

By contrast with a shallow learning algorithm, the main challenge in deep learning is tremendous computational complexity (Bengio, 2009; Deng and Yu, 2014). To train a deep model, a layer-wise greedy strategy should be employed. Considering a parameter (i.e., biases and weights of each RBM) update rule, the parameters are updated by computing the gradient of the negative log likelihood $\log P(v)$.

5. Traffic matrix prediction

Traffic matrix prediction is defined as the problem that estimates the future network traffic from the previous and achieved network traffic data. It is widely used in network planning. In this section, we will propose a method to predict network traffic from an achieved traffic matrix. We first introduce how to train the DBN from the achieved traffic matrix. Then a predictor of network traffic will be obtained via the trained DBN.

We assume that the achieved traffic matrix is X whose entry is $x_{n,t}$ ($n = 1, 2, \ldots, N$, $t = 1, 2, \ldots, T$). The DBN is a binary network, which causes an extensive error. Hence, we here first normalize the achieved traffic matrix X by dividing the maximum of X so that all entries of X are $[0, 1]$. For simplicity, during the rest of this section, the traffic matrix is regarded as the normalized one without special statement. Besides, we use the RBM with real-valued units for prediction (Salakhutdinov and Hinton, 2007). We take an OD pair as an instance to introduce our prediction method. In this case, we use x_t ($t = 1, 2, \ldots, T$) to denote an OD pair, and then $\hat{x}_{T+1}$ denotes the network traffic that need to be predicted.

Training data: In our prediction method, a slide window is employed to build the training data set. See Fig. 3, consider an OD pair with T timeslots, let the length of the slide window be W (denoted by the gray frame), then the number of the input layer is $W - 1$. The value on the right of the slide window is defined as the output of DBN. With sliding the window, for the OD pair with T timeslots, we can acquire $T - W + 1$ training sets. We denote the training set of the k-th window as $\mathrm{x}_{input}{}^{k}$ and $\mathrm{x}_{output}{}^{k}$ ($k = 1, 2, \ldots, T - W + 1$), obviously, $x_{input}^{k} = [x_k, x_{k+1}, \ldots, x_{k+W-2}]$ and $x_{output}^{k} = x_{k+W-1}$.

RBM learning: According to the above training data sets, the architecture of DBN in our method is shown in Fig. 4. It consists of an input layer, an output layer, and M hidden layers. Each hidden layer is made up of P units. The input layer has $W - 1$ units. The output layer has a single unit. We denote the p-th unit ($p = 1, 2, \ldots, P$) in the m-th hidden layer by $h_p^{(m)}$. The challenge of RBM learning is computing the negative log likelihood gradient function, since the gradient consists of a sum of all states of the model (Deng and Yu, 2014). We take advantage of the contrastive divergence algorithm proposed by Hinton et al. (2006) to approximately estimate the gradient instead of directly

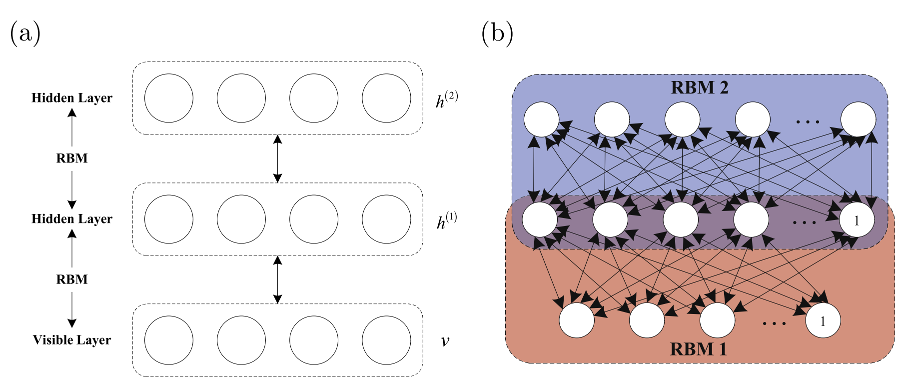

Fig. 2. (a) Stack of RBMs; (b) DBN architecture.

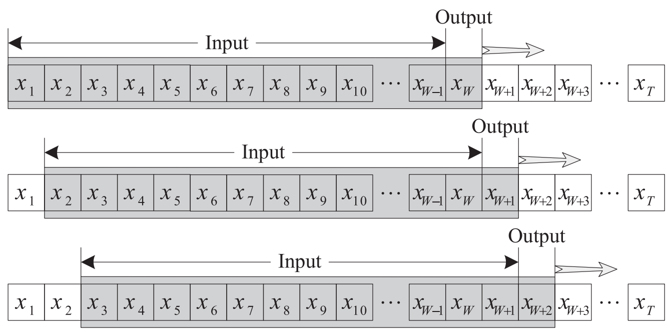

Fig. 3. Slide window for building training data set.

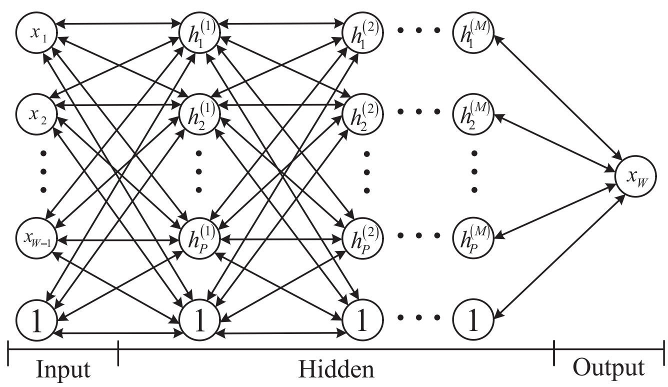

Fig. 4. DBN architecture for traffic matrix prediction.

computing it.

DBN learning is implemented by a greedy layer-by-layer strategy. By training DBN model, we can obtain the estimator $\hat{x}_{T+1}$ using $[x_{T-W+2}, \ldots, x_T]$ as the input of the trained model.

6. Traffic matrix estimation

Traffic matrix estimation is solving the inverse inference system with ill-posed feature shown in Eq. (1). In this paper, we will estimate the traffic matrix by the DBN introduced in the above section. In our estimation method, we first train the DBN by the priors of link counts and traffic matrix (denoted by $\tilde{Y}$ and $\tilde{X}$). We use $\tilde{T}$-timeslot data as the training data, and then the link counts $\tilde{Y}$ and the traffic matrix $\tilde{X}$ are L-by-$\tilde{T}$ and N-by-$\tilde{T}$. The architecture of the DBN used to estimate the traffic matrix X is shown in Fig. 5. It has L input units, N output units and M hidden layers. Each hidden layer has P units. For the training data, each corresponding column of $\tilde{Y}$ and $\tilde{X}$ is regarded as an input and output pair of the DBN. If we denote each entry of $\tilde{Y}$ and $\tilde{X}$ by $\tilde{y}_{l,t}$ ($l = 1, 2, \ldots, L$) and $\tilde{x}_{n,t}$ ($n = 1, 2, \ldots, N$), then the input and output pair is shown in Fig. 5. Notice that the training data is normalized by

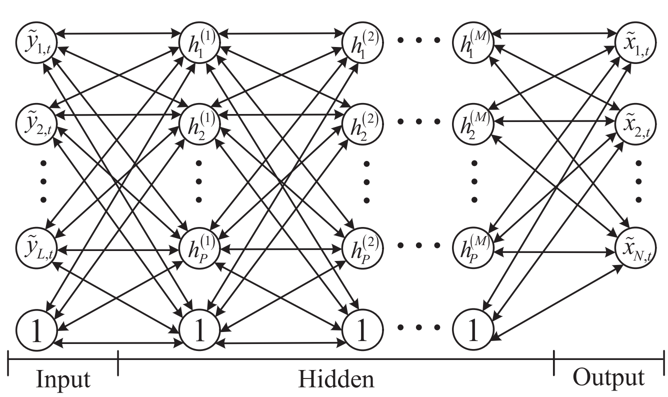

Fig. 5. DBN architecture for traffic matrix estimation.

dividing the maximum of the link counts $\tilde{Y}$. As mentioned in the above, the parameters of the DBN are updated according to the log likelihood $\log P(v)$, that is $\log P(\tilde{y}_t)$ in our estimation method where $\tilde{y}_t = [\tilde{y}_{1,t}, \tilde{y}_{2,t}, \ldots, \tilde{y}_{L,t}]$. In this case, from Eqs. (2), (3), (5) and (6), the log likelihood is calculated by

$$\frac{\partial \log P(\tilde{y}_t)}{\partial \theta} = -\sum_{\tilde{x}_t}\left(P(\tilde{x}_t|\tilde{y}_t)\frac{\partial E(\tilde{y}_t, \tilde{x}_t)}{\partial \theta}\right) + \sum_{\tilde{y}_t, \tilde{x}_t}\left(P(\tilde{y}_t, \tilde{x}_t)\frac{\partial E(\tilde{y}_t, \tilde{x}_t)}{\partial \theta}\right), \quad (8)$$

where $\tilde{x}_t = [\tilde{x}_{1,t}, \tilde{x}_{2,t}, \ldots, \tilde{x}_{N,t}]$, and θ denotes the parameters of DBN.

By training the model in Fig. 5, the DBN can learn the feature of the inverse inference system. Furthermore, the function $f\colon \tilde{y}_t \to \tilde{x}_t$ can be achieved from the trained DBN. Then we can estimate the traffic matrix X using Y as a variable of the function. Besides, the estimator of X should obey the constraint of the inverse inference system, thus the Iterative Proportional Fitting algorithm (Soule et al., 2005) is employed in our method to correct the estimator of the traffic matrix.

7. Simulation results and analysis

In this section, we will validate the prediction and estimation errors of our methods using real network traffic data sets from the Abilene and GÉANT backbone networks. The Abilene network consists of 12 peer nodes and 54 undirected links, which is used in education and research. The GÉANT network is made up of 23 peer nodes and 120 undirected links. We record 2010-timeslot traffic data from the Abilene network in our simulation, and each entry of the traffic matrix reveals the mean of the traffic flow over 5-min interval. Similarly, 2004-timeslot traffic data is sampled from the GÉANT network by 10-min interval. In our simulation, we also compare our prediction and estimation methods with a state-of-the-art method, that is, the PCA method introduced in the above section.

7.1. Prediction

This subsection will discuss the prediction effectiveness of our method. In our simulation for traffic matrix prediction, we set the parameters of DBN empirically for exact prediction, that is, the number of hidden layers is 8 and each hidden layer has 100 units. We first plot the prediction bias of our method for the Abilene and GÉANT networks in Figs. 6 and 7. The bias is defined as

$$bias_n = \frac{1}{T}\sum_{t=1}^{T}(\hat{x}_{n,t} - x_{n,t}), \quad (9)$$

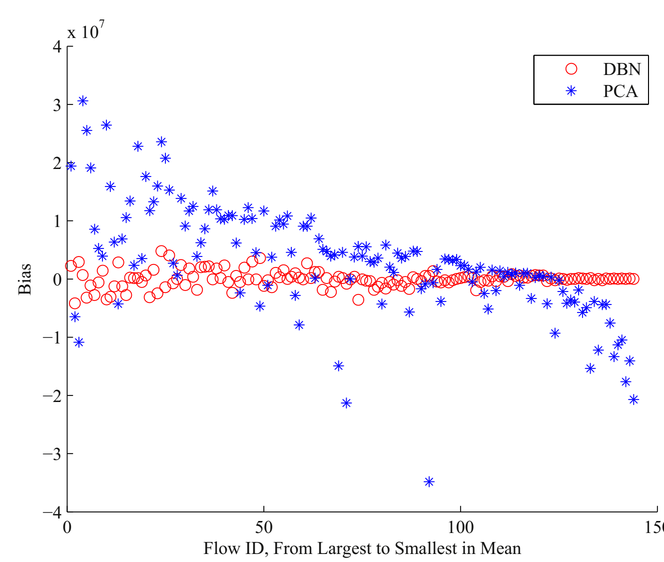

Fig. 6. Prediction Bias in Abilene.

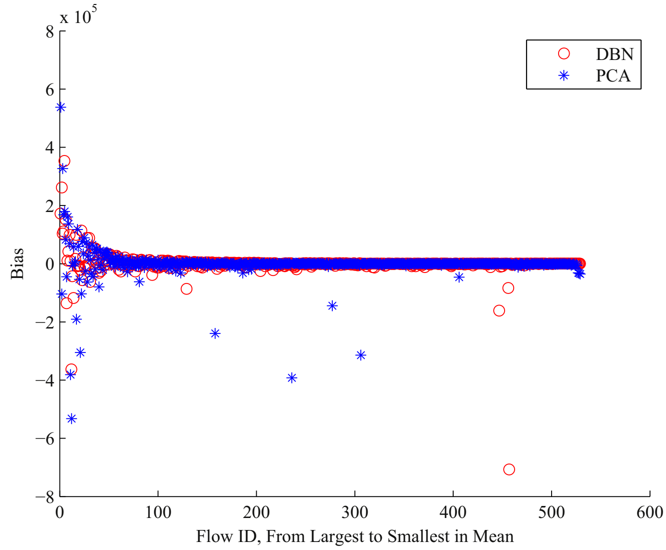

Fig. 7. Prediction Bias in GÉANT.

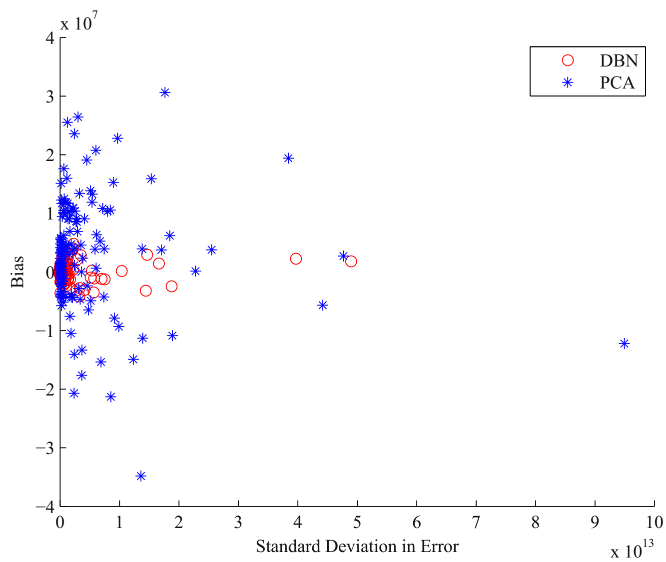

Fig. 8. Bias versus standard deviation in Abilene.

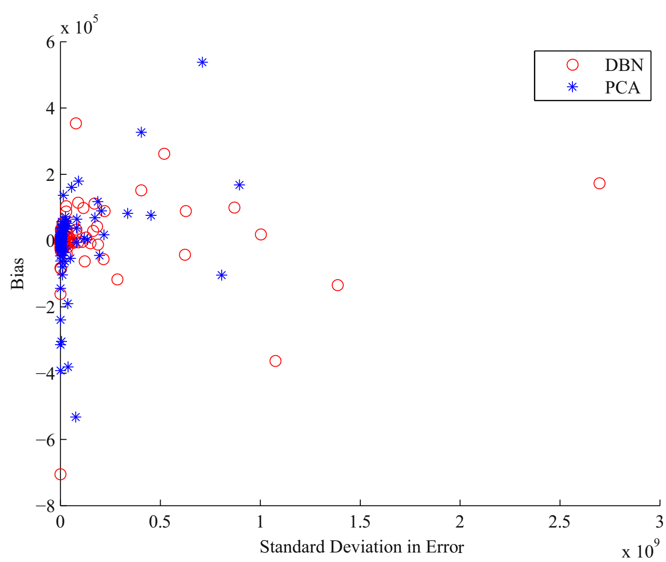

Fig. 9. Bias versus standard deviation in GÉANT.

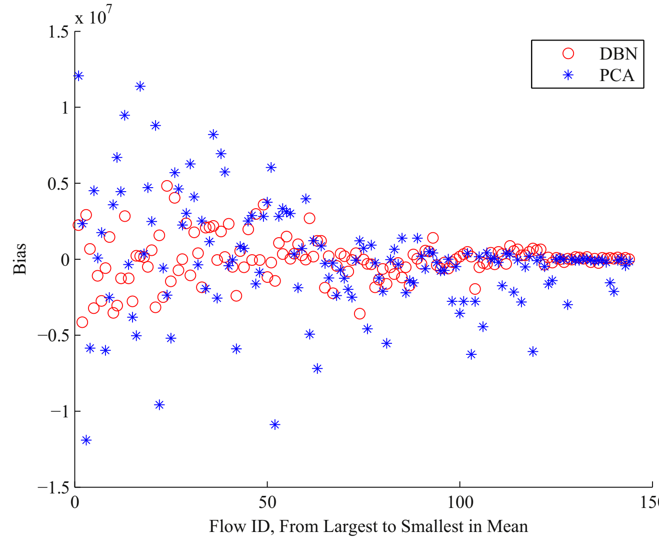

Fig. 10. Estimation Bias in Abilene.

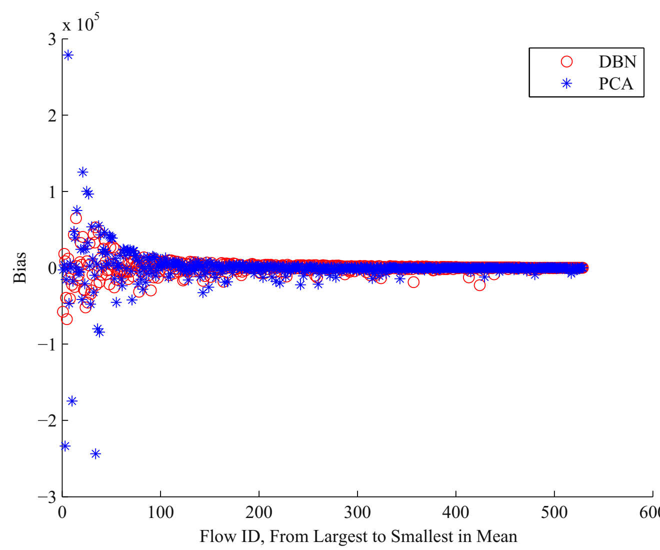

Fig. 11. Estimation Bias in GÉANT.

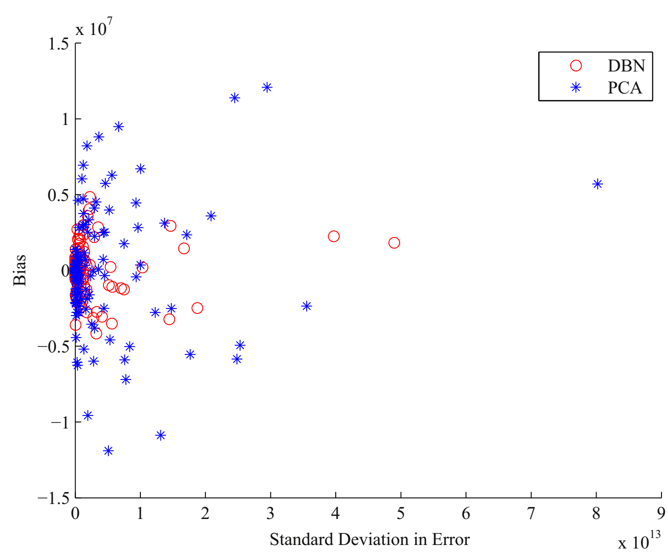

Fig. 12. Bias versus standard deviation in Abilene.

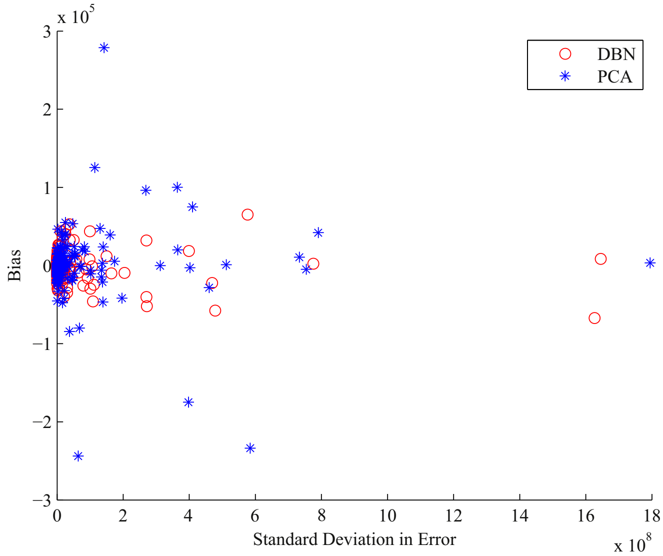

Fig. 13. Bias versus standard deviation in GÉANT.

where $x_{n,t}$ and $\hat{x}_{n,t}$ are real traffic entries of the traffic matrix and their predictors. The x-axis denotes the identity of each OD pair, which is arranged in descending order with respect to the mean of traffic flow. The y-axis illustrates prediction bias. Fig. 6 states that the biases of our method are decreased with the volume of traffic flows decreasing. By contrast, the biases of PCA are unordered, and some biases (i.e., the smallest OD pairs) are unsatisfactory. That is because the PCA method uses partial principal components to approximate a traffic matrix so that the network tomography model is an over-determined system. Hence, it performs poorly in bias for the smallest OD pairs. Especially, for the OD pairs with incisive jitters, the PCA method is not sensitive enough. We can acquire the same conclusion from the simulation results for the GÉANT network in Fig. 7. The trends of traffic flows in GÉANT are relatively steady. Hence, the frame of biases of PCA looks like that of our method, though there are individual biases for two methods.

A predictor with little bias is not equal to an effective one. Namely, once it has a high variance, an unbiased predictor also can be viewed as an unavailable one. In other words, an available predictor needs both well bias and variance. Thus, we refer to the standard deviation as a metric to assess the effectiveness of our prediction method. The standard deviation is defined as

$$sd_n = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}((\hat{x}_{n,t} - x_{n,t}) - bias_n(n))^2}\,. \tag{10}$$

Figs. 8 and 9 plot the bias versus the standard deviation for the Abilene and GÉANT networks. We find that, for the predictors with small biases, the standard deviation of our method is smaller than that of the PCA method. It declares that our method prefer to predict an OD pair with short timescale variation, i.e., it can provide faithful predictors for individual time points. Contrarily, the PCA method tends to predict an OD pair over a long time interval. Our method deeply learns the statistical features between the predictor and the training data. Therefore, it is outstanding in predicting a traffic entry individually. The PCA method extracts the principal components to approximately describe a traffic matrix. In this case, it is adept in capturing the mean of an OD pair.

7.2. Estimation

In this subsection, we will assess the estimation feature of our method. In these simulations, let the number of hidden layers be 12, and there are 100 hidden units in a hidden layer. Similarly, we also first explore the bias of our method dealing with traffic matrix estimation problems. According to Fig. 10, with the volume of traffic flows increasing, the biases of two methods are increased. We find that our method shows fewer estimation biases for the Abilene network. The PCA method has some high biases for the smallest OD pairs. In Fig. 11, we observe that the PCA method has a significantly negative bias for some largest OD pairs.

The standard deviation is plotted by Figs. 12 and 13. For the Abilene network, two methods perform significantly differently in standard deviation. PCA has the largest standard deviation for individual entries. For most entries of the traffic matrix, the standard deviation is low in our method. Hence, in traffic matrix estimation problem, we can make a similar conclusion that our method tends to estimate a short-term OD pair. According to the above simulation results, we see that our method is a powerful method not only for prediction but also for estimation. It remarkably outstands in prediction and estimation biases.

8. Conclusions

This paper focuses on the problems of traffic matrix prediction and estimation in large-scale IP backbone networks. Considering the various features of traffic flows, we use deep learning techniques to capture the features of a flow pair. Then we propose a traffic matrix prediction method based on the deep belief network. In terms of this deep belief network architecture, we further propose an effective traffic matrix estimation method. Finally, we assess the effectiveness of the proposed prediction and estimation methods by real network traffic data sets from the Abilene and GÉANT networks. We refer to a state-of-the-art method in our simulations, and compare our method with it. The results state that our method is able to accurately deal with the traffic matrix prediction and estimation problems.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (Nos. 61571104, 61071124), the General Project of Scientific Research of the Education Department of Liaoning Province (No. L20150174), the Program for New Century Excellent Talents in University (No. NCET-11-0075), the Fundamental Research Funds for the Central Universities (Nos. N150402003, N120804004, N130504003), and the State Scholarship Fund (201208210013). The authors wish to thank the reviewers for their helpful comments.

References

Bengio, Y., 2009. Learning deep architectures for AI. Found. Trends Mach. Learn. 2 (2), 1–127.
Chen, C.L.P., Zhang, C., Chen, L., Gan, M., 2015. Fuzzy restricted Boltzmann machine for the enhancement of deep learning. IEEE Trans. Fuzzy Syst. 23 (6), 2163–2173.
Deng, L., Yu, D., 2014. Deep learning: methods and applications. Found. Trends Signal Process. 7 (2–3), 197–389.
Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504C507.
Hinton, G.E., Osindero, S., The, Y., 2006. A fast learning algorithm for deep belief nets. Neural Comput. 18 (7), 1527–1554.
Hu, Z., Qiao, Y., Luo, J., 2015. Atme: accurate traffic matrix estimation in both public and private datacenter networks. IEEE Trans. Cloud Comput. 2015 (99), 1–14.
Huang, W., Song, G., Hong, H., Xie, K., 2014. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. IEEE Trans. Intell. Transport. Syst. 15 (5), 2191–2201.
Kumlu, D., Hokelek, I., 2015. Network traffic estimation using Markov chain and incremental gaussian mixture. In: Proceedings of 2015 23th Signal Processing and Communications Applications Conference, pp. 1187–1190.
Mardani, M., Giannakis, G., 2013. Robust network traffic estimation via sparsity and low rank. In: Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 4529–4533.
Niu, Y., Tian, H., 2014. Study on a new model for network traffic matrix estimation, In: Proceedings of 2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming, pp. 152–154.
Polverini, M., Iacovazzi, A., Cianfrani, A., Baiocchi, A., 2015. Traffic matrix estimation

enhanced by SDNS nodes in real network topology. In: Proceedings of 2015 IEEE Conference on Computer Communications Workshops, pp. 300–305.
Roughan, M., Zhang, Y., Willinger, W., Qiu, L., 2012. Spatio-temporal compressive sensing and internet traffic matrices (extended version). IEEE Trans. Netw. 20 (3), 662–676.
Salakhutdinov, R., Hinton, E., 2007. Using deep belief nets to learn covariance kernels for gaussian processes. In: Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, pp. 1249–1256.
Soule, A., Lakhina, A., Taft, N., Papagiannaki, K., Salamatian, K., Nucci, A., Crovella, M., Diot, C., 2005. Traffic matrices: balancing measurements, inference and modeling. In: Proceedings of SIGMETRICS 2005, pp. 362–373.
Svigelj, A., Sernec, R., Alic, K., 2015. Network traffic modeling for load prediction: a user-centric approach. IEEE Netw. 29 (4), 88–96.
Xu, K., Shen, M., Cui, Y., Ye, M., Zhong, Y., 2014. A model approach to the estimation of peer-to-peer traffic matrices. IEEE Trans. Parallel Distrib. Syst. 25 (5), 1101–1111.
Zhang, Y., Roughan, M., Duffield, N., Greenberg, A., 2003. Fast accurate computation of large-scale ip traffic matrices from link loads, ACM SIGMETRICS Performance Evaluation Review, vol. 31, 2003, pp. 206–207.
Zhao, E., Tan, L., 2015. A PCA based optimization approach for IP traffic matrix estimation. J. Netw. Comput. Appl. 57 (2015), 12–20.