

Research Progress Report

Botao Zhu

January 6, 2019

1 Reading and Research Activities

1.1 Summary

[1] proposed a deep learning based routing table construction method for a GPU-accelerated SDR. They adopted supervised Deep Belief Architectures (DBA) to compute the subsequent nodes with the traffic patterns of the edge routers as the input. And then, they presented unique characterizations of inputs and outputs based on the traffic patterns at the edge routers. They demonstrated how the trained DBA can predict the nodes, the benefits of the deep learning based routing strategy in terms of lower signaling overhead as well as fast convergence and effectiveness of proposed deep learning based solution compared to a benchmark routing method through both analysis and extensive.

[2] aims to solve the following problems: (1) mitigate the traffic overhead imposed on BSs in disaster situations, (2) extend BS coverage through collaboration with wireless ad-hoc networks, (3) establish the maximum number of communication connections in situations in which network resources are insufficient and communication links are unstable. In this paper, the node degree of wireless communication is classified by deep learning for disaster situations and virtual routes are set according to the predetermined node by employing the Viterbi algorithm.

1.2 Deep Learning Based Routing Strategy

1.2.1 Input and Output Design

The traffic pattern is served as the input to the deep learning structure and processed for routing path decision as the output. The traffic pattern at each router can be defined as the number of inbound packets of the router during each time interval, such as $\beta \Delta t$. Therefore, by assuming that a network comprises of N routers, we can use a matrix of β rows and N columns to represent the traffic patterns of all routers. The simulation results demonstrate that it is accurate enough to set the value of β to 1. The output layer can be designed to give the next node similar to the distributed routing strategy because of its lower complexity and higher tolerance. In the vector, only a single element has the value of 1, the order of which represents the next node. So, we can use two N dimensional vectors to represent the input and output.

1.2.2 Deep Learning Structure Design

The authors choose the DBA as the deep learning structures as shown in Figure 1.

DBA can be also seen as a stack of $(L - 2)$ Restricted Boltzmann Machines (RBM) and a logistic regression layer as the top layer. The structure of RBM consists of two layers, the visible layer, \mathbf{v} , and the hidden layer, \mathbf{h} , shown in Figure 2. While training an RBM, sets of unlabeled data are given to the visible layer, and the value of the weights and biases are repeatedly adjusted until the hidden layer can reconstruct the visible layer. To mathematically model the training process, they use a log-likelihood function of the visible layer given as follows.

$$l(\theta, a) = \sum_{t=1}^m \log p(v^t) \quad (1)$$

where θ denotes the vector consisting of all the values of the weights and biases of the hidden layer.

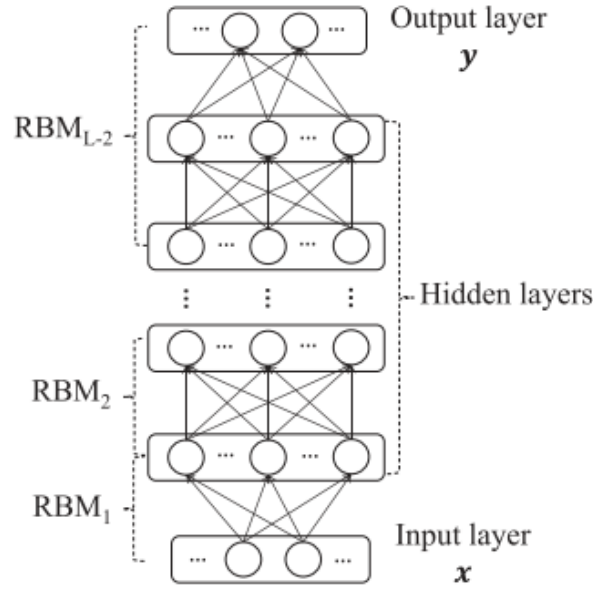


Figure 1: Considered L-layer DBA

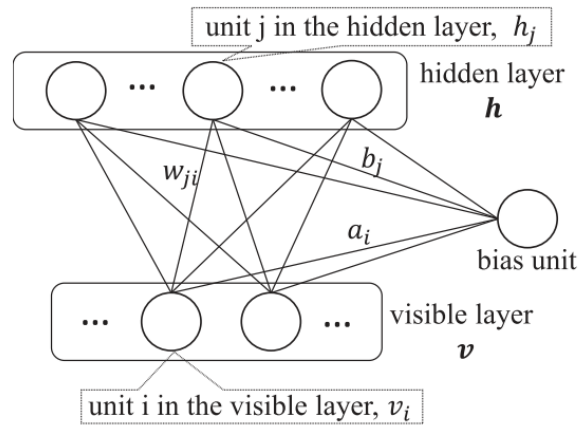


Figure 2: RBM

$$E(v, h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j h_j w_{ji} v_i \quad (2)$$

$$p(v, h) = \frac{e^{-E(v, h)}}{Z} \quad (3)$$

$$Z = \sum_v \sum_h e^{-E(v, h)} \quad (4)$$

where v_i and h_j are the unit i in the visible layer and the unit j in the hidden layer respectively. Z represents the normalizing constant partition function. Also, the relationship between $p(v)$ and $p(v, h)$ can be expressed as follows:

$$p(v) = \sum_h p(v, h) \quad (5)$$

However, the complexity of the calculation is extremely high. To solve this problem, they use the Gibbs Sampling method to sample the values of v and h to approximate the real values since the conditional distribution probability of one layer can be calculated. As each unit is independent on the other units in the same layer, when one layers is fixed, the conditional distribution probability of the other layer can be calculated as follows:

$$p(v|h; \theta, a) = \prod_i p(v_i|h; \theta, a) \quad (6)$$

$$p(h|v; \theta, a) = \prod_j p(h_j|v; \theta, a) \quad (7)$$

Since input units are continuous, they use Gaussian probability distribution to model the traffic patterns. Equations(2) and (6) should be revised as follows:

$$E(v, h) = -\sum_i \frac{(v_i - a_i)^2}{2\delta_i^2} - \sum_j b_j h_j - \sum_i \sum_j \frac{v_i}{\delta_i} h_j w_{ji} \quad (8)$$

$$p(v_i|h; \theta, a) = N\left(a_i + \delta_i \sum_j h_j w_{ji}, \delta_i^2\right) \quad (9)$$

where δ_i is the value of the variance for the unit v_i . However, in Figure 3, we can see that the visible layer of RBM_{L-2} consists of not only RBM_{L-3} 's hidden layer but also the output layer of the DBA, y . Its hidden layer is the top hidden layer of the DBA and energy function is expressed as follows.

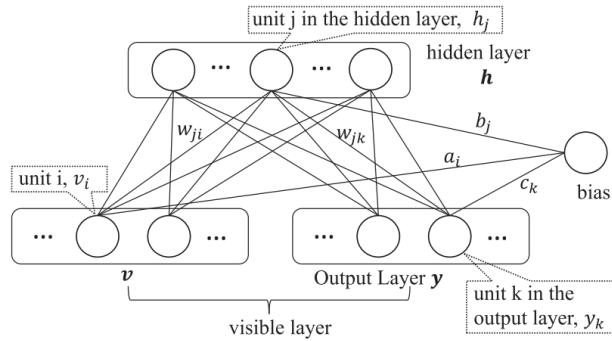


Figure 3: Last RBM

$$E(v, h, y) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_k c_k y_k - \sum_i \sum_j h_j w_{ji} v_i - \sum_j \sum_k h_j w_{jk} y_k \quad (10)$$

where y represents the vector in the output layer. c_k is the bias of the unit y_k . w_{jk} represents the weight of the link connecting the units h_j and y_k . The conditional distribution of the concatenated vector consisting of v and y is,

$$p(v, y|h; \theta, a) = p(v|h; \theta, a) p(y|h; \theta, a) \quad (11)$$

$$= \prod_i p(v_i|h; \theta, a) \prod_k p(y_k|h; \theta, a) \quad (12)$$

The purpose of supervised training is to minimize the difference between the output of the DBA and the labeled output y . They use the cross-entropy cost function to measure their difference given as bellow.

They use the cross-entropy cost function to measure the difference between the output of the DBA, denoted by $h_\theta(x)$, and the labeled output y .

$$C(\theta) = -\frac{1}{m} \sum_{t=1}^m (y^t \log(h_\theta(x^t)) + (1 - y^t) \log(1 - h_\theta(x^t))) + \frac{\lambda}{2} \sum_{l=2}^L \sum_{j=1}^{n_l} \sum_{i=1}^{n_{l-1}} (w_{ji}^l)^2 \quad (13)$$

(x^t, y^t) is the t th training data. $h_\theta(x^t)$ denotes the output of the DBA. $C(\theta)$ consists of two parts: the difference between the output of DBA and the labeled output, keeping the training process from overfitting. Using the gradient descent to update θ to minimize the value of $C(\theta)$. η_{bp} is the learning rate in the back-propagation process.

$$\theta := \theta - \eta_{bp} \frac{\partial C(\theta)}{\partial \theta} \quad (14)$$

1.2.3 The Procedures of the Proposed Deep Learning Based Routing Strategy

1.2.3.1 Initialization Phase The goal of the initialization phase is to obtain the labeled data which consist of the input vector and the corresponding output vector. They approach a number of available data set sources, such as the center for applied internet data analysis, and extract the traffic information and relevant routing paths.

1.2.3.2 Training Phase The training phase consists of two steps: the loop of the Greedy Layer-Wise training to train each RBM and the following backpropagation process to fine-tune the weights of links between the layer. Let N and I denote the total number of routers and the number of inner routers. So, every edge router needs to train $(N - I - 1)$ DBAs while each inner router needs to train $(N - I)$ DBAs. Then, every edge router needs to send its θ of $(N - I - 1)$ to other $(N - I - 1)$ edge routers. Every inner router needs to send its θ of $(N - I)$ DBAs to all the edge routers. Therefore, each edge router obtains θ of all the DBAs of all the routers in the network, and the number of sets of θ is $(N - I)(N - 1)$.

Algorithm 1. Supervised Train DBA

Input: $(x, y) = \{(x^t, y^t) | t = 1, \dots, m\}, \eta_{CD}, \eta_{bp}, L, n = (n_1, \dots, n_L)$

Output: θ

```

1: for  $i = 1, \dots, L - 2$  do
2:   TrainRBM( $u^i, \eta_{CD}, n_i, n_{i+1}$ )
3: end for
4: Fine-tuneDBA( $(x, y), \theta, \eta_{bp}$ )
5: return  $\theta$ 

```

1.2.3.3 Running Phase Every edge router can utilize the next node information to construct the whole paths from itself to all the other edge routers. They use an array of N elements, $\mathcal{TP}[N]$, to save the numbers of inbound packets of N routers in the network to represent the traffic patterns, and $\theta[N - I][N - 1]$ to save the parameters of all the DBAs in the network. Another array $\mathcal{ER}[N - I]$ is used to save the sequence numbers of the edge routers in the network since they are not continuous. Each edge router can obtain the outputs of DBAs to construct the paths to $(N - I - 1)$ edge routers. And then using a matrix, $\mathcal{NR}[N][N - I - 1]$, to save the results of these DBAs that can be used to build the whole paths to all the other edge routers. Figure 4 is the routing model and Table 1 is an example of the routing table.

Table 1: Routing table of R_3

Dest	Path
R_1	$R_3 \rightarrow R_2 \rightarrow R_1$
R_2	$R_3 \rightarrow R_2$
...	...
R_{16}	$R_3 \rightarrow R_7 \rightarrow R_{11} \rightarrow R_{15} \rightarrow R_{16}$

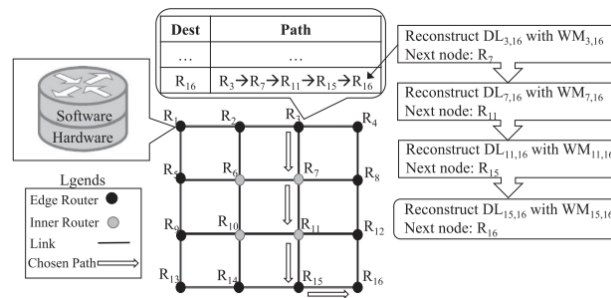


Figure 4: Routing model

2 Objectives for the Next 2 Weeks

3 Advisor's Comments

I'm very happy.cite [1]

Bibliography

- [1] (2015) The IEEE website. [Online]. Available: <http://www.ieee.org/>