Network disruption prediction based on neural networks

D.S. Taimanov[1]

[1]*Samara National Research University, 34 Moskovskoe Shosse, 443086, Samara, Russia*

Abstract

Network disruptions cause significant financial losses and discomfort of customers. However, communication systems provide various data about equipment condition. This information can be used to predict network disruptions. Purpose of research is applying neural networks to network disruption prediction. Introduced approach has good feature extraction ability and data corruption resilience. Representation for network disruption dataset has been developed. Chosen network type is deep belief networks. Net structure variations have been proposed for chosen data representation and neural network type. Experimental research of proposed methods gives meager results for selected dataset. Results can be improved by net structure complication and by increasing dataset volume.

Keywords: data mining; neural networks; prediction; network disruptions; big data

1. Introduction

Network disruptions cause financial losses and inconveniences. They are increasing with the growth of telecommunication influence on all spheres of life. Communication systems amplification and data science advancement allow predicting network disruptions. Communication systems provide various data about equipment condition. Data science gives different methods of prediction. Network disruption prediction involves continuous network condition monitoring and disruption hazard evaluation.

It is difficult to find present task in previously published works. So tasks with similar data types have been observed. Network disruptions data values mostly belong to enumerated unordered type. Similar data types appear in work [1]. Authors used deep belief networks to identify risk factors and predict bone disease progression.

There is an interesting approach to use neural networks for telecommunication disruptions prediction. “Telstra Network Disruptions” dataset [2] is destined for disruption prediction.

2. Network equipment condition data

“Telstra Network Disruptions” dataset consists of records about events. Each event described by attributes: *location*, *event_type*, *resource_type*, *severity_type*, *log_feature*. The goal of competition [2] is distinguish each event between different kinds of *fault_severity*. There are three kinds (values) of *fault_severity*: 0 – normal operation, 1- momentary glitch, 2 – total interruption of connectivity. Each attribute described in table 1.

Table 1. “Telstra Network Disruptions” dataset structure.

Attribute name	Number of different values, N_i	Type	Repetition, values/event	Description
id	7381	integer, unordered	-	Identifier, unique. Intended for data union.
fault_severity	3	integer, ordered	-	Class of event, **forecast objective**.
location	929	enumerated, unordered	-	Network equipment location.
event_type	49	enumerated, unordered	12468/7381	Type of event.
resource_type	10	enumerated, unordered	8460/7381	Type of resource that caused event.
severity_type	5	enumerated, unordered	-	Type of log message
log_feature	331	tuple: *type – volume* *type*: enumeration, unordered *volume*: integer, ordered	23851/7381	Features extracted from logs with its volume

As we can see, all significant attributes belong to enumerated (categorical) unordered type. This attributes shows record affiliation with some of class by some of parameter. For example, “location 1”, “event_type 15”, “resource_type 8”, “severity_type 2”. *log_feature* attribute is a tuple “type of value, volume of value” (e.g. “feature 35, 17”).

Attributes are kept in separate tables distributed by different CSV-files. This manner of data storage is used because of attribute values repetition. For example, single event can be of “event_type 15” and “event_type 11” at the same time. *id* attribute is used for table matching. Table structure is represented in figure 1 a.

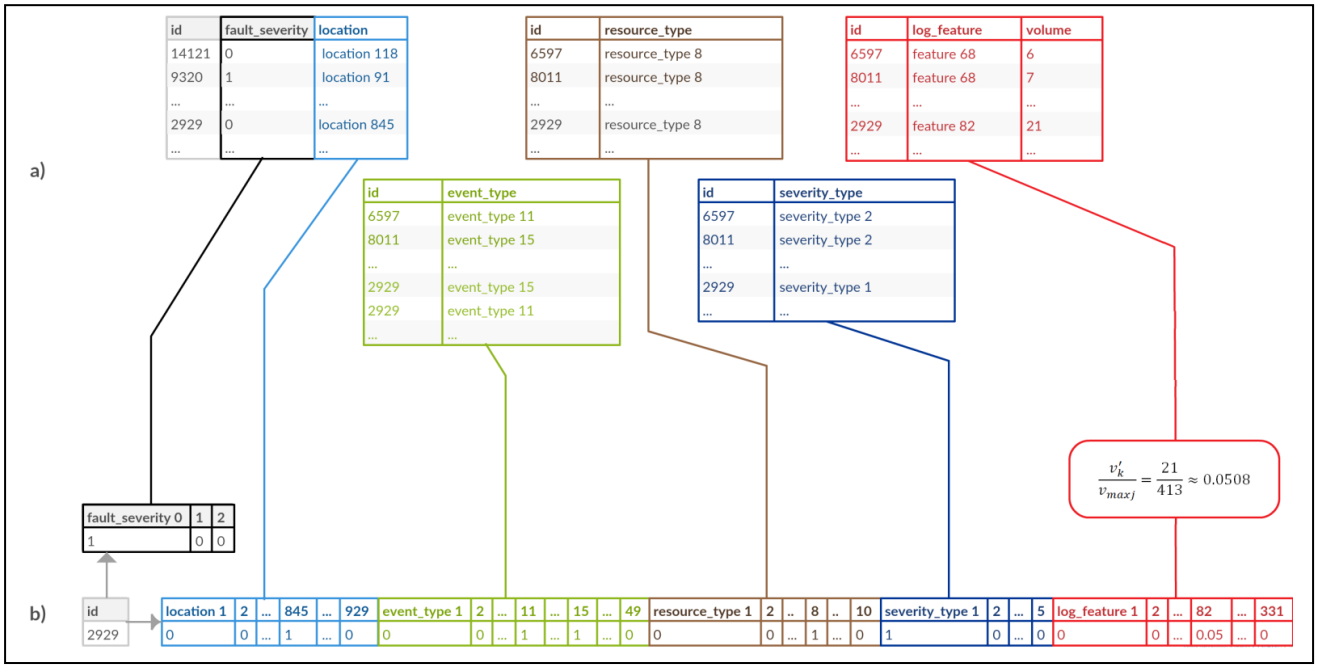

Fig. 1. Dataset structure (a) and input vector representation (b).

We have to make representation of event to construct input vector. Each attribute translates into a sparse vector. This vector has a corresponding index for each possible attribute value. Sparse vector has "1" at corresponding index if attribute possess this value for current event and "0" otherwise (figure 1 b). This is the simplest way to construct input vector for unordered categorical types.

Thus, input vector parts for *location*, *event_type*, *resource_type* and *severity_type* are as follows:

$$\boldsymbol{v_i} = (x_1, .., x_{N_i}).$$

Here N_i is the amount of *i*-th attribute values; $i = \overline{1, M}$, M – amount of categorical attributes; x_j is as follows:

$$x_j = \begin{cases} 1, if\ X_{ij}\ \in \boldsymbol{f}_i \\ 0, otherwise \end{cases}.$$

Here $\boldsymbol{f}_i$ are values of *i*-th attribute for current event; X_{ij} – value of *i*-th attribute corresponding to *j*-th index.

fault_severity input vector part constructs in the same way.

Values for *log_feature* are different from values for other attributes. They not only appeared or not appeared for current event but have volume. Corresponding elements of input vector possess this volume (normed) instead of "1" if present. Thus, last equation is as follows for *log_feature* attribute:

$$x_j = \begin{cases} \dfrac{v'_k}{v_{maxj}}, & if\ Y_j = f_k \\ 0, & otherwise \end{cases}.$$

Here v'_k is volume of *k*-th appeared feature; f_k – *k*-th appeared feature («type» in tuple); Y_j – value of *log_feature* corresponding to *j*-th index, $j = \overline{1, N_Y}$; N_Y – possible *log_feature* values amount; где v_{maxj} – maximum value for *j*-th feature.

Thus, input vector *V* is a sequence of attribute vector parts $\boldsymbol{v_i}$, normed and has length *N*:

$$N = \sum_{i=1}^{M} N_i + N_Y.$$

N is 1324 for "Telstra Network Disruptions" dataset. This is a large value. So, deep neural networks have been used.

3. Deep neural networks

Dee belief networks [3] have been chosen. This type of deep neural networks has good hidden feature extraction ability and can be fast trained and fine-tuned for high-dimensional datasets. Deep belief networks (DBN) is a composition of restricted Boltzmann machines (RBM) [4]. RBMs stacks layer by layer to construct DBN (figure 2).

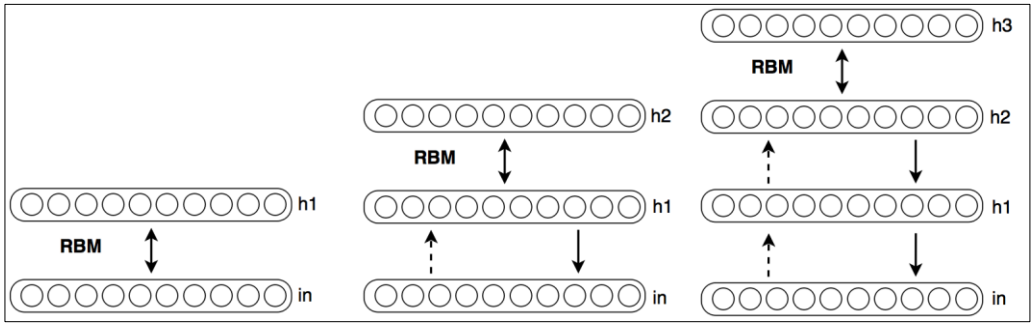

Fig. 2. Deep belief network construction.

Boltzmann machine (figure 3 a) is a stochastic machine formed by stochastic neurons [5]. This net fully connected. Neurons are divided into visible and hidden. Visible neurons are clumped onto values from dataset during the training. Hidden neurons always operate freely. These neurons can capture high order statistical correlations in the clumped values [5].

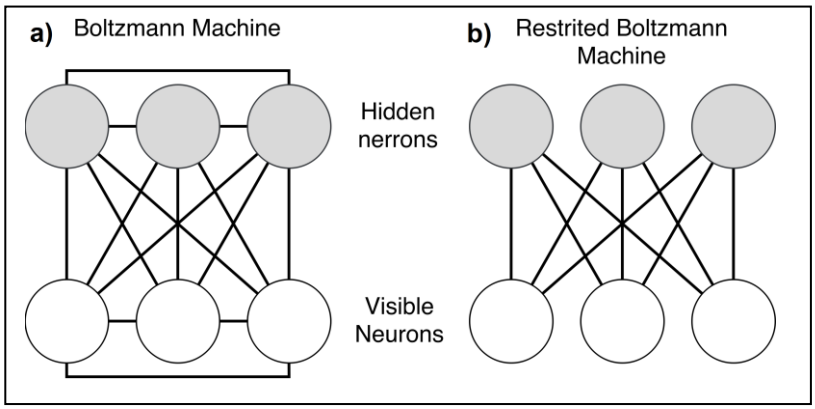

Fig. 3. Example of a figure caption.

Boltzmann machine has one significant disadvantage: too long time of training. It caused by multiple neuron activation. Restricted Boltzmann machine avoid this problem by removing connections between neurons with the same type (figure 3 b). RBMs can be fast trained at practice [4].

Algorithm [6] of training DBNs are shown below (figure 2):

1) Train first two layers (h1, h2) as RBM using dataset.
2) Pass dataset through trained RBM and get values from layer “h2”. Values got from “h2”layer are a new dataset.
3) Train next pair of layers (h2, h3) using new dataset, then repeat step 2 and 3 for next layers and so on until last hidden layer trained. Previous steps are called “pretraining phase”.
4) Train whole net using backpropogation or another common algorithm to fine-tune weights. This step is called “training phase”

Set of possible architecture variations forms during research:

- Various types of nets (DBN, Perceptron)
- Different layers count
- Various layer sizes

Neural nets have 1324 neurons at input layer and 3 neurons at output layer for “Telstra Network Disruptions” dataset.

This dataset can be divided into train and test parts using three different ways:

1) Random division.
2) Division for each *location*. Events for each location are divided into train and test parts separately.
3) Division by *location.* All events for certain location occur in one of sets entirely.

4. Results and Discussion

Regular accuracy ranking function has been used. It is a right-to-total ratio:

$$A = \frac{P}{N}.$$

Here *A* is accuracy; *P* is a count of correct predictions; N – total predictions count.

Two loss functions have been tried: mean square error and weighted mean square error.

$$f(\boldsymbol{y}) = \frac{1}{I}\sum_{i=1}^{I}(\hat{y_i} - y_i)^2.$$

Here $f(\boldsymbol{y})$ – mean square loss function; $\mathbf{y}$ – vector of predictions; y_i – likelihood of i-th class, $y_i \in \overline{0,1}$; $\hat{y_i}$ – observed value for *i*-th class; I – number of classes.

Mean square function has one significant disadvantage for “Telstra Network Disruptions” dataset: considerable class imbalance often causes degenerate models construction. These models classify any input vector into one and the same class, the largest class.

Described problem can be solved via using a weighted mean square loss function:

$$f_w(\boldsymbol{y}) = \frac{1}{I}\sum_{i=1}^{I}\left((\widehat{y_i} - y_i)\right)^2 \times c_i.$$

Here $f_w(\boldsymbol{y})$ – weighted mean square loss function, c_i – significance of prediction error for i-th class.

Setting c_i inversely proportional to class occurrences number helps to solve the problem.

Best accuracy values for different networks consist of two and three hidden layers are listed in table 2.

Table 2. Best accuracy for different neural networks.

Network type	Error measure dataset	Pretraining	Hidden layers count	
			2	3
Deep Belief Network	Train	Usual	0.901	0.913
	Test	Usual	0.745	0.756
	Test	Extended	0.749	0.757
Perceptron	Test	-	0.749	0.751

Two types of networks have been considered: deep belief networks and perceptron (as a comparison). Accuracy and loss function have been measured via train or test selection. Additional data from test part of source dataset has been used in one of experiments.

Deep belief networks reach better results in comparison with perceptron for selected dataset and train vector representation.

Accuracy is increasing with the growth of hidden layers count. Therefore, it is reasonable to try networks that are more complex.

Additional data usage causes accuracy growth. It shows that better results can be reached for bigger datasets.

5. Conclusion

There is an interesting approach to use neural networks for telecommunication disruptions prediction. It is reasonable because of high dimension of input data. Deep belief networks have been selected. This type of deep neural networks has good hidden feature extraction ability and can be fast trained and fine-tuned for high- dimensional datasets.

Best reached accuracy is 75.7 % of correct predictions. This result can be improved by net structure complication. Bigger dataset usage also can help.

References

[1] Li H, Li XY, Ramanathan M. Identifying informative risk factors and predicting bone disease progression via deep belief networks. Methods 2014; 69(3): 257–265.

[2] “Telstra Network Disruptions” competition. URL: https://www.kaggle.com/c/telstra-recruiting-network (01.02.2017).

[3] Hinton GE. Deep belief networks. URL: http://www.scholarpedia.org/article/Deep_belief_networks (01.02.2017).

[4] Hinton GE. Boltzmann_machine. URL: http://www.scholarpedia.org/article/Boltzmann_machine (01.02.2017).

[5] Haykin SS. Boltzmann machine. Neural Networks: A Comprehensive Foundation 1999; 11: 584–491.

[6] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Computation 2006; 18: 1527–1554.