Convergence Optimization of Backpropagation Artificial Neural Network Used for Dichotomous Classification of Intrusion Detection Dataset

Ivan Homoliak*, Dominik Breitenbacher, Petr Hanacek
Brno University Security Laboratory, Department of Intelligent Systems, Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic.

* Corresponding author. Tel.: +420541141185; email: ihomoliak@fit.vutbr.cz
Manuscript submitted October 20, 2015; accepted December 28, 2015.
doi: 10.17706/jcp.12.2.143-155

Abstract: There are distinguished two categories of intrusion detection approaches utilizing machine learning according to type of input data. The first one represents network intrusion detection techniques which consider only data captured in network traffic. The second one represents general intrusion detection techniques which intake all possible data sources including host-based features as well as network-based ones. The paper demonstrates various convergence optimization experiments of a backpropagation artificial neural network using well know NSL-KDD 1999 dataset, and thus, representing the general intrusion detection. Experiments evaluating usefulness of stratified sampling on input dataset and simulated annealing employed into the backpropagation learning algorithm are performed. Both techniques provide improvement of backpropagation's learning convergence as well as classification accuracy. After 50 training cycles, classification accuracy of 84.20% is achieved when utilizing stratified sampling and accuracy of 86.5% when both stratified sampling and simulated annealing are used. In contrast, the backpropagation by itself reaches only 76.63% accuracy. Comparing to state-of-the-art work introducing the NSL-KDD dataset, there is achieved accuracy higher about 4.5%.

Key words: Artificial neural network, backpropagation, data mining, intrusion detection.

1. Introduction

Artificial neural networks (ANN) are in general considered as an effective approach of handling and classifying patterns. However, soft precision, long training cycles as well as overstimulation hinder their real world application. This is due to the fact that an ANN often intake a large number of inputs and utilize many computational neurons in one or more hidden layers which makes them very challenging in accurate and fast training. The ANNs are expected to carry out by themselves the majority of the computational work leading to a decision which makes them dependent on powerful computational units.

The main objective of our work is to contribute on convergence of an ANN especially in the area of intrusion detection where its deployment is very challenging because every misclassification can cause serious implications [1]. There is selected the backpropagation ANN because it is designed for supervised machine learning and is able to resolve nonlinear separable problems. Lot of papers aimed on intrusion detection proclaim the backpropagation as effective and powerful method [2]-[4] which also prompted us to use it.

The paper is organized as follows. Section 2 states the related work. Section 3 presents analysis of the testing dataset. Section 4 describes the design and the topology of utilized ANN. Section 5 proposes convergence optimization techniques of the ANN. Section 6 presents the experimental results using proposed modifications. Section 7 discusses real world deployment of machine learning in intrusion detection. Section 8 concludes the paper and outlines the future work.

2. Related Work

There is divided related work into two categories, both presenting concepts of Intrusion Detection Systems (IDS) employing ANN. The first one contains papers presenting ANNs which assume only features extracted from network traffic. The second part mentions papers presenting ANNs which utilize features extracted from various sources, including network and host ones. This separation can be generalized on all machine learning approaches in IDS. The latter category is more relevant to the experiments described in the current paper because of used input dataset. The former category is more challenging as its data are limited to network traffic features only.

2.1. ANN for Network Intrusion Detection

Artificial neural networks are examined in a lot of works aimed on network intrusion detection. One of these works presents a system called HIDE and is described in the papers [5], [6]. HIDE detects network-based attacks as anomalies using statistical preprocessing and ANN classification. The authors tested five different types of ANN and they achieved the best results by the backpropagation ANN and perceptron-backpropagation-hybrid ANN which has directly connected input neurons to neurons in an output layer.

The authors Labib *et al*. describes the Network Intrusion Detection System (NIDS) using Self-Organizing Maps (SOM) called NSOM [7]. NSOM uses a structured SOM to perform unsupervised classification of real-time Ethernet network data. The paper also presents a graphical tool continuously displaying the clustered data which reflects network activities.

Following paper [8] demonstrates efficient analysis of network data from perspective of intrusion detection. The authors of the paper perform data clustering by SOM and detection by a multilayer perceptron ANN. The DARPA 1999 network dataset [9] is used for experiments and authors aim on denial of service attacks, distributed denial of service attacks and port scans. The authors achieved correct predictions of normal traffic in 100% of the cases but simultaneously they achieved only 24% of correctly predicted attacks.

2.2. ANN for General Intrusion Detection

In the context of this work, the general intrusion detection is referred to a technique which utilize features extracted from a network and a host machine.

Hawkins *et al*. use Replicator Neural Networks (RNNs) to provide a measure of the outlyingness of data records [10]. The effectiveness of the RNNs for outlier detection is demonstrated on two publicly available datasets including a reduced subset of the KDD Cup 1999 [11] considering only logged in attacks and relevant legitimate traffic [12].

The paper [4] gathers a survey of various ANN approaches used for intrusion detection. Almost all approaches use the KDD Cup 1999 dataset. The authors mention the backpropagation ANN as very accurate supervised machine learning method for the purpose of intrusion detection and classification. On the other hand, the authors mention, that it suffers from local minimum problem and slow convergence.

An anomaly-based intelligent intrusion detection system OCTOPUS is designed in [13]. The authors utilize two layers' approach. The first layer contains Kohonen ANN (representing SOM) which performs unsupervised classification and aids in reducing of the false negative rate. The second layer consists in a

Support Vector Machines (SVM) classifier which classify traffic into two classes: normal and malicious traffic. SVM are very efficient in identification of the anomalies [14] and can bear a certain amount of noise in the input. OCTOPUS achieved accuracy of *97.40%* with maximal deviation of *8.57%* using the KDD Cup 1999 dataset.

An intrusion detection model employing the backpropagation ANN based on dynamic change learning rate strategy combined with the simulated annealing algorithm was proposed in the paper [3]. It changes the learning rate value according to the change in system error between the previous iteration and the current one. The experiments with the KDD Cup 1999 show that the approach is effective and improves the detection efficiency and the real-time property of intrusion detection system.

The paper [15] presents hierarchical anomaly IDS employing SOM ANN which performs unsupervised machine learning. The results show that SOM recognized malicious traffic from normal one in 92.37% of the cases when testing on the KDD Cup 1999, while it achieved only 75.49% accuracy in the case of using the NSL-KDD dataset [16].

The authors Naoum *et al*. propose usage of a multilayer perceptron ANN trained using an enhanced resilient backpropagation training algorithm for intrusion detection [2]. In order to increase the speed of convergence, an optimal learning factor was added to the weight update equation. The experiments demonstrate promising results in terms of accuracy, storage and time using a subset of the NSL-KDD dataset containing circa one quarter of the original dataset. The designed system was capable to classify records with a detection rate about 94.7%.

An ensemble cluster classification technique using SOM ANN for intrusion detection is proposed in the paper [17]. The authors presents promising results achieving high classification accuracy with regards to precision.

3. Analysis of the Testing Dataset

The most popular dataset for evaluating the performance of network intrusion detection methods is the KDD Cup 1999 [11] which was collected during simulated attacks in military testing environment. The drawbacks of the dataset were discussed in [16]. The original dataset KDD Cup 1999 contains about five millions of records and more than half of them are duplicates – considering both the testing and the training datasets. Another drawback of the dataset is that it does not contain uniform class distribution of the records which resulted in poor performance of classification obtained by various methods including Neural Networks, SVM [18], etc.

3.1. The NSL-KDD Dataset

The shortcomings of the KDD Cup 1999 dataset prompted the authors of the paper [16] to preprocess the original KDD Cup dataset. The training and testing sets of the NSL-KDD dataset contain *125 973* and *22 544* records, respectively. The records of the original KDD Cup dataset are labeled by *22* classes – each class represents a different type of attack. The NSL-KDD dataset itself [19] contains records labeled by *22* classes and 2 classes, respectively. In the two classes' expert knowledge case, the first class represents legitimate traffic records and the second class malicious one. Experiments performed in this paper utilize classification into two classes using the NSL-KDD dataset.

3.2. Dataset Reduction

The reduced training dataset of the NSL-KDD [19] was selected for some of our experiments. It contains *20%* of the original NSL-KDD training dataset with the same class distribution like the original one has. The selection of the reduced training NSL-KDD dataset was prompted by high time complexity of the ANN training. On the other hand, the testing of trained ANN is performed on the complete testing dataset.

3.3. Data Preprocessing

ANN works on numeric variables, therefore the first preprocessing step is substitution of all nominal variables for numerical ones. NSL-KDD dataset contains three nominal attributes which are transformed into sequences of doubles starting with *0.0* and increasing by *1.0* for the purpose of our work. Normalization of attributes into specified range is proposed to use as well. Another data preprocessing step is stratified sampling which will be discussed as convergence optimization technique later. The correct functionality of implemented solution was validated by AND and XOR problems.

4. Design of Backpropagation ANN and Its Topology

As mentioned before, the backpropagation ANN was chosen because it is designed for supervised machine learning and is able to resolve nonlinear separable problems which general intrusion detection is supposed to be. The learning mechanism of the backpropagation ANN is based on gradient descent technique [20]. A common method for measuring the difference between the expected output t and the actual output y of a neuron is the squared error measure [21]:

$$E = \frac{1}{2}(t - y)^2, \tag{1}$$

where E is the squared error. For each neuron j its output o_j is defined as

$$o_j = \varphi(net_j) = \varphi\left(\sum_{k=1}^{n} w_{kj} o_k\right), \tag{2}$$

where net_jrepresents weighted sum of outputs o_k of neurons in the previous layer. The number of inputs of the current neuron is n. The activation function φ has to be nonlinear and differentiable. A commonly used activation function is the sigmoidal function

$$\varphi(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

having differential equation

$$\frac{d}{dz}\varphi(z) = c_1\varphi(z)(c_2 - \varphi(z)). \tag{4}$$

In order to follow gradient descent method for computing of weights' change, partial derivative of the error with respect to a weight w_{ij} is needed to be calculated by formula

$$\frac{\partial E}{\partial w_{ij}} = \delta_j o_i \tag{5}$$

which represents a gradient. The δ_j is defined by formula

$$\delta_j = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net}_j} = \begin{cases} (o_j - t_j)\varphi(\mathrm{net}_j)\left(1 - \varphi(\mathrm{net}_j)\right) & \text{if } j \text{ is an output neuron,} \\ \left(\sum_{i \in L} \delta_l w_{jl}\right)\varphi(\mathrm{net}_j)\left(1 - \varphi(\mathrm{net}_j)\right) & \text{otherwise,} \end{cases} \tag{6}$$

where L represents next layer to neuron j. A learning rate α is needed to be chosen in order to update the weight w_{ij}. The change of weight which is used for additive update of old weight, is equal to the product of

the learning rate and the gradient, multiplied by -1 [21]:

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}. \tag{7}$$

If the backpropagation is provided with an appropriate number of hidden layers, then it can minimize the error of nonlinear functions with high complexity [22]. There is considered two main aspects when designing the topology of the ANN: the dichotomous classification and the number of attributes contained in the NSL-KDD dataset which is equal to *41*. Therefore, the most appropriate and intuitive topology of the ANN should contain *41* and *2* neurons in the input and output layers, respectively. According to [4], one hidden layer of an ANN is sufficient for the problem of intrusion detection. Considering that and requirement of low computational costs, the only one hidden layer is selected. A number of neurons in the hidden layer is one of the subjects in our experiments' stage. There is utilized logistic activation function for output and hidden layers in the ANN which was designed. The input layer utilizes linear activation function which is the same like one in (2). All of our experiments utilize fully connected ANN's layers with adjacent ones.

5. Proposed Experimental Modifications

This section describes all proposed techniques which are supposed to influence speedup of ANN's training convergence, thus influence classification accuracy in the validation process. The outcome of experiments with these techniques will be analyzed in Section 6.

5.1. Initialization of Weights

The initial setting of weights is performed by a pseudo-random generator. The same seed is chosen because of requirement on accurate comparison of experiments each other and reproducing them as well. When normalization of input values is used, then the initial values of weights can be set by the uniform distribution assuming range

$$\left\langle -\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}} \right\rangle, \tag{8}$$

where d represents the number of inputs of a neuron. Note that this range only makes sense when input dataset is normalized to have mean *0* and standard deviation *1*.

5.2. Normalization of Attributes

Some sources state the best performance of the backpropagation ANN can be achieved by normalization of input attributes into range $\langle 0.1, 0.9 \rangle$ [23]. The normalization described in the paper [24] is utilized in this work. The authors define normalization by formula

$$\tilde{x}_t = \frac{(x_t - x_{t,\min})(b - a)}{(x_{t,\max} - x_{t,\min})} + a. \tag{9}$$

The a and b represent the required interval of normalization. The $x_{t,\min}$ and $x_{t,\max}$ represent minimal and maximal value of an attribute x_t observed in dataset.

5.3. Training with Random Sample Selection

While training an ANN, the samples are iteratively processed in the same order as they occur in the dataset. The proposed modification performs a swap of $n/2$ randomly selected pairs of samples after every cycle of backpropagation's training. The n denotes the size of a training dataset. In other words, this technique represents stochastic model of learning, therefore it reduces chances of getting stuck in local minima.

5.4. Training with Momentum

The objective of employing this technique is to prevent deadlock in local extremes – extreme solutions. Originally, a delta – back propagated error of neuron – is computed using deltas and weights of neurons in the next layer. When a momentum is employed, then actual delta is computed considering previous value of delta, too. Therefore, weight change of training iteration t is computed by

$$\Delta w_{kj}(t) = \eta \frac{\partial E_s}{\partial w_{kj}} + \alpha \Delta w_{kj}(t-1), \tag{1}$$

where $\alpha > 0$ and $\eta < 1$ [20]. Effective learning rate for each weight should be able to adapt as needed and helps speedup convergence. The momentum was first time introduced and discussed in [21] as convergence improvement technique.

5.5. Stratified Sampling with Equal Number of Samples

The stratified sampling [25] is a general data mining preprocessing method serving for partitioning objects into mutually exclusive groups by some criteria. In the case of this work there are two classes representing the legitimate and malicious records. The stratified sampling with equal number of samples, moreover, performs selection of the same samples' number in each group. This is due to the fact that learning algorithms are often biased towards the more frequent records and prevent them from learning infrequent records which are usually more harmful. Therefore, this technique aims on prevention of inaccurate learning when classes have different sizes.

5.6. Simulated Annealing

The simulated annealing (SA) is proposed as the last convergence optimization technique aimed on overcoming of local extremes. SA was also applied in the paper [26] as an independent optimization technique of ANN as well as other ones, including genetic algorithms and the backpropagation. In contrast to it, there is applied SA as a convergence improvement technique of training by the backpropagation algorithm.

Two options how to apply SA on backpropagation training are recognized – decrease temperature and accept a state with some probability: A) after going-over a whole training dataset or B) after going-over each training sample. After a few experiments, the second option was selected. Therefore, the perturbation of an ANN (meaning the weights' modification) was executed until a new state was not accepted with some probability. The pseudocode of backpropagation's training procedure which applies simulated annealing algorithm is following:

Epsilon represents desired minimal error. If it is reached, training of ANN stops. The algorithm is stated in its final form and utilize all proposed modifications from this section except the normalization of input attributes. Note that the stratified sampling method is applied before the ANN training process because it is data preprocessing step. The best convergence optimization results are achieved by this algorithm and are described in the next section.

```
procedure trainANN(temperature, tempStep, trainData, epsilon, momentum, learningRate, layers) {
  trainCycles = 0
  actMin = MAX
  do {
        globalError = 0.0
        prevError = 0.0
        for (tEntry in trainData) {
            do {// perturbation
                out = classify(tEntry)
                backpropag(tEntry, trainCycles, momentum)
                modifyWeights(learningRate)
                // computation of actual error from current and expected output
                actError = 0.0d
                lastLayer = layers[layers.size() - 1]
                for (i = 0 ; i < lastLayer.getNeurons.size() ; i++) {
                    actError += pow(out[i] - tEntry.targetOutputs.get(i), 2.0d)
                }
            } while (nextRandomFloat() > exp((prevError - actError) / temperature))
            prevError = actError
            globalError += 0.5 * sqrt(actError)
        }
        randomizeCollection(trainData) // application of random sample selection
        if (globalError < actMin) {
            actMin = globalError
            saveWeights(layers) // store best configuration of weights
        }
        trainCycles++
        temperature = (temperature - tempStep <= 0) ? 0.1 : temperature - tempStep
    } while (globalError > epsilon)
}
```

6. Experiments and Results

This section is aimed on analysis of convergence behavior and accuracy of classification depending on previously described optimization techniques which will be tested. Many experiments were performed, but only the three most significant will be discussed. These experiments have several points in common. The common applied techniques/settings are following:

- Usage of momentum because it was successful and there was achieved faster convergence of utilized ANNs in all preliminary experiments.
- Initial weights' setting was performed by linear congruential pseudo-random generator [27] with the same initial seed in order to make results of experiments comparable.
- It was disabled normalization of input attributes' values because after series of experiments it showed itself to be inaccurate and had negative impact on convergence of a training process.
- Stochastic model of learning is employed, meaning that each propagation of sample is followed immediately by a weight update. It refers to training with random sample selection technique.
- Finally, the interval for the weights' initialization was chosen in range $\langle -0.05, 0.05 \rangle$ according to several preliminary experiments.

6.1. Experiment 1: Various Topologies of ANN

This experiment examines classification accuracy of the backpropagation ANN depending on the number of neurons in the only hidden layer. There is considered the assumption: *3* layers of the ANN, including input and output ones, are enough to achieve accurate results. The ANN was trained on *500* samples of the reduced training dataset and validated on all samples of the complete testing dataset. They were executed *1 000* traversals through the training dataset for each topology.

Results: The results of this experiment are depicted in Fig. 1. There are two series in the graph: blue (darker) and orange (lighter). The orange series refers to the case when the training dataset was used for

training as well as validation purposes. Considering the orange series, there can be analyzed dependence of learning accuracy of the ANN on a number of neurons in the hidden layer. The blue series represents the case when the complete NSL-KDD testing dataset was used for validation purposes. This case shows how the ANN was able to generalize during classification.

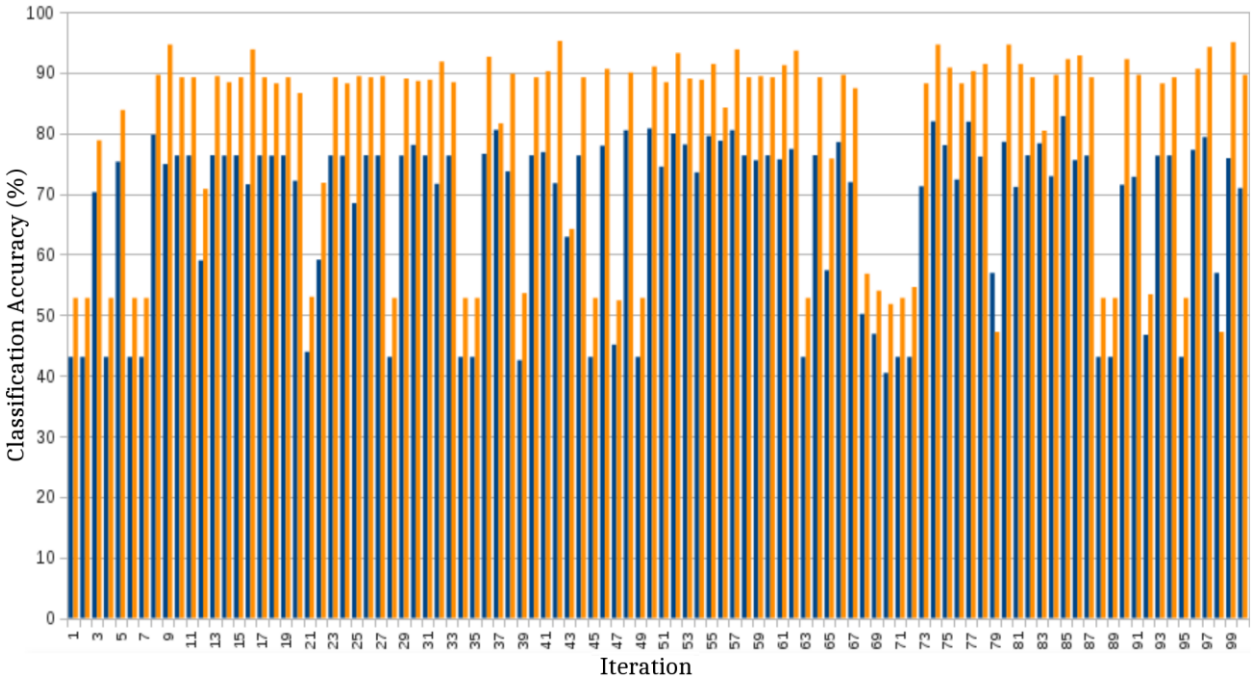

Fig. 1. Classification accuracy dependence on the number of neurons.

Looking on the number of neurons equal to *37*, the minimal difference of accuracy between training and validation experiment can be observed and is equal to *1.07%*. Next, the number of neurons equal to *42* reflects the best achieved learning accuracy of *95.2%*. When testing dataset is used for classification purposes, then it is achieved the best classification accuracy of *82.81%* in the topology containing *85* neurons in the hidden layer. Note that also topology with *74* and *77* neurons achieved interesting results. The interesting configurations of ANN's topology will be considered in the next experiments.

6.2. Experiment 2: Convergence of Learning with Stratified Sampling

This experiment compares the learning convergence of the backpropagation ANN trained with the stratified sampling method described in Section 5 and the ANN trained without it. The topology instance of the ANN contained three layers consisting of: *41* neurons in the input layer; *1* neuron in the output layer; and *42* neurons in the hidden layer. More instances of a topology were experimented as well, but only one of the best is presented. The ANN was trained on the whole NSL-KDD training dataset. *50* traversals through the training dataset were executed and the best found configuration of weights was stored. The reason why only *50* traversals through the training dataset were chosen, is the fact that it contained circa *126 000* samples, making the training process very slow regarding to the time complexity of it.

Results: When this experiment was executed without stratified sampling method, then the best global error was equal to 4 898.73. If the stratified sampling method was enabled, then the best global error equal to 4 780.04 was achieved. In the first case, classification accuracy of 76.63% was achieved when testing on the complete NSL-KDD testing dataset. In the second case, classification accuracy of 84.20% was achieved – meaning significant improvement of it. The comparison of these two cases is depicted in Fig. 2.

There is obvious decreasing trend of global error when stratified sampling was enabled – meaning optimization of extremes (fluent decreasing) and overcoming of extremes (sudden jump). But, when it was

disabled, then deadlock in local extreme occurred and solution of local extreme was not optimized well. Therefore, we consider the stratified sampling with equal number of samples as method that is able to improve learning performance of an ANN and is able to overcome deadlocks in local extremes and optimize all extremes.

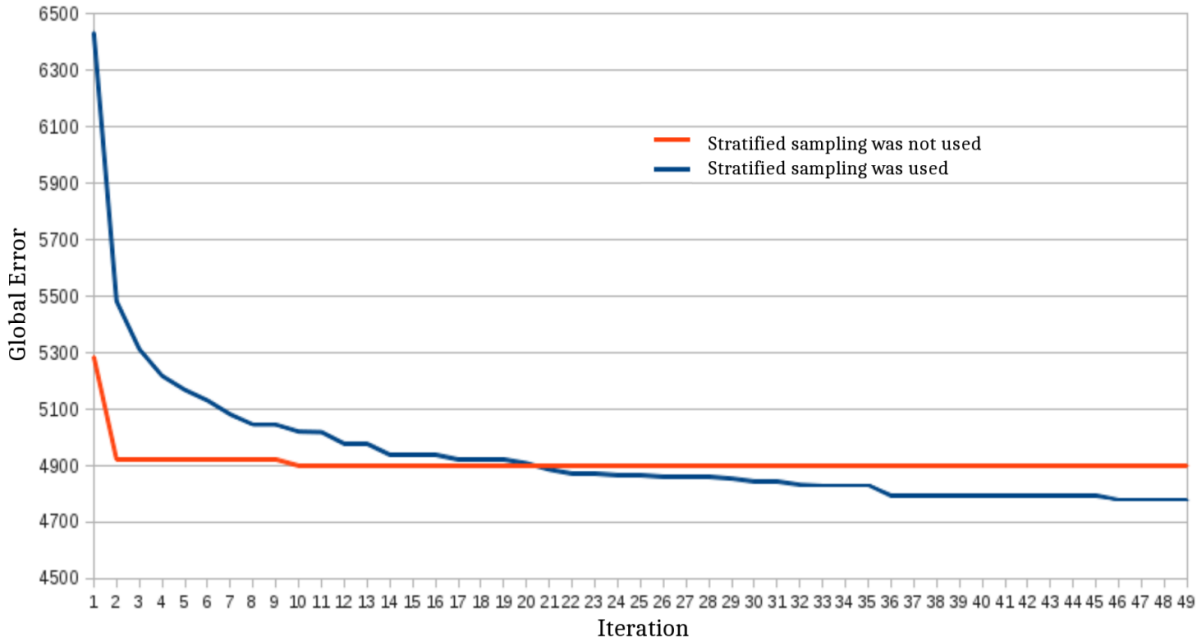

Fig. 2. Comparison of the ANN trained with stratified sampling and without it.

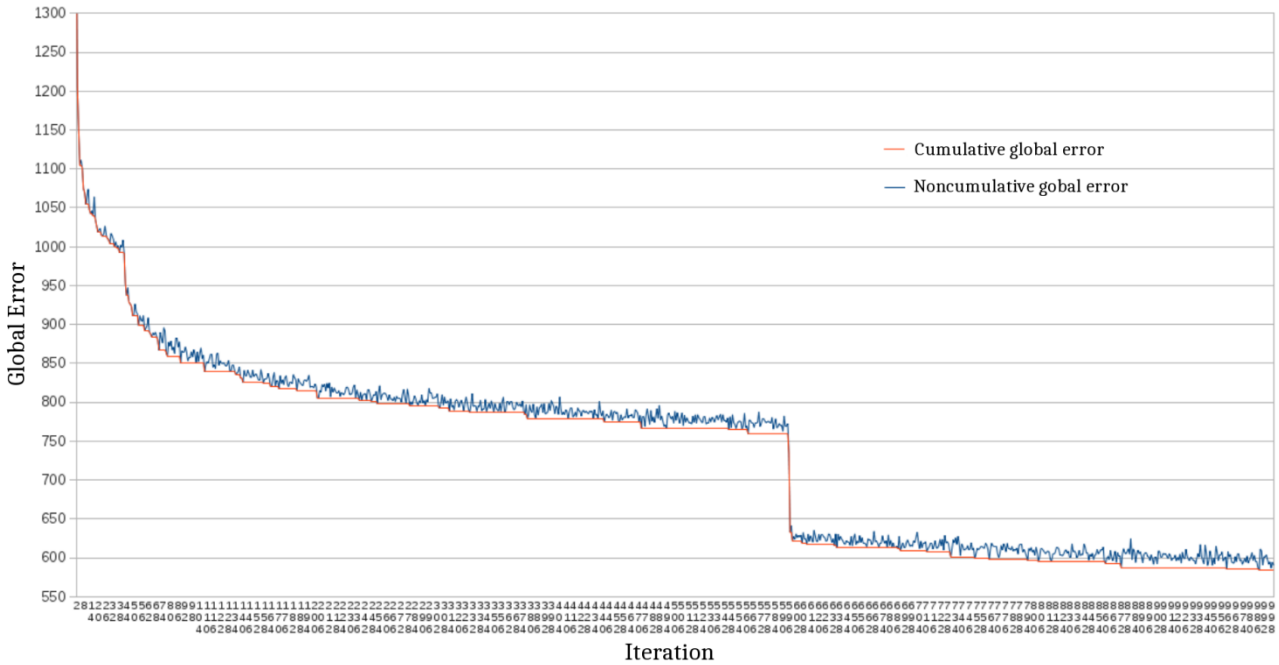

Fig. 3. Convergence dependence on stratified sampling.

Additional Remarks: The current experiment was also executed with enabled stratified sampling and normalization of attributes described in Section 5. Classification accuracy of *43.07%* with global error equal to *28 099.42* were achieved, therefore normalization did not bring any improvement.

Next, stratified sampling experiment was also performed on the reduced training dataset, but containing

1 000 iterations of training. Classification accuracy of *85.85%* and minimal global error equal to *584.63* were achieved using the reduced testing dataset. The convergence of the ANN is depicted in Fig. 3 which contains two series – the blue series (darker one) and red series (lighter one). The blue series represents noncumulative global error and the red one represents cumulative global error. A decreasing trend of global error in both cases can be observed – this fact confirms the technique has the ability to optimize extremes when using higher number of iterations as well. Sudden jumps of global error represent the ability of the technique to overcome local extremes. Two such jumps of global error can be seen around iteration *40* and *590*.

6.3. Experiment 3: Convergence of Learning with Simulated Annealing

This experiment compares learning convergence of the backpropagation ANN trained with SA method described in Section 5 and the ANN trained without it. The experiment also applies stratified sampling with equal number of samples which showed to be accurate in previous experiment. As in the previous experiment, the training of the ANN was performed on the whole NSL-KDD training dataset. The training process consisted of a 50 traversals through the training dataset and the best found configuration of weights was stored. This experiment presents results obtained using the same instance of topology like previous one.

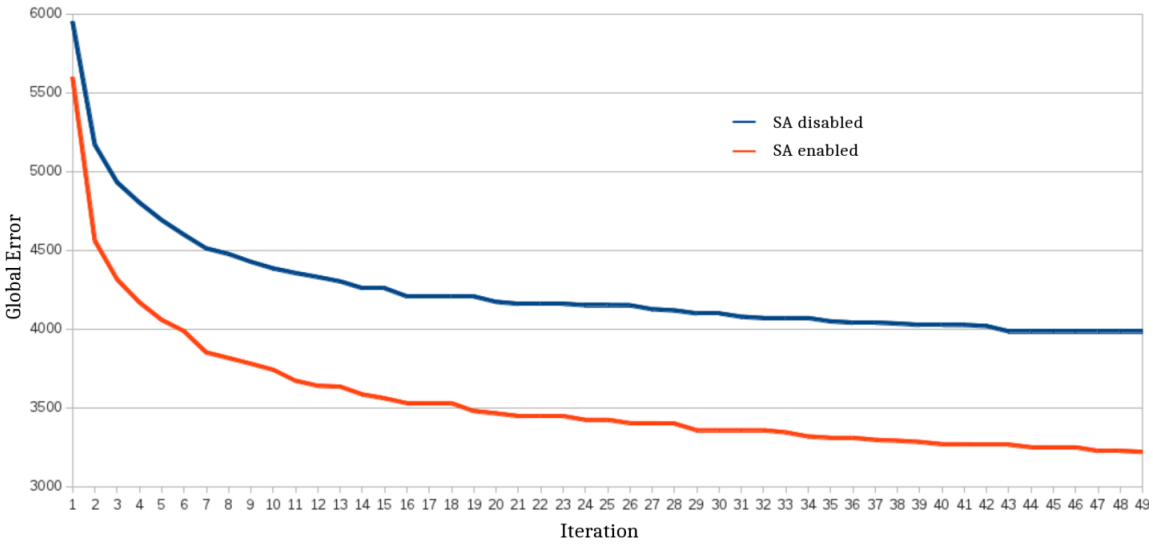

Fig. 4. Comparison of the ANN trained with simulated annealing and without it.

The difference between this experiment and previous one is new initial seed of pseudo-random generator. The seed was preserved the same among runs of the current one.

Various temperatures and various ways of their decreasing were experimented, but finally, there were selected settings according to a size of training dataset. The restoration of temperature after each training iteration was also utilized.

Results: Fig. 4 illustrates learning convergence of the ANN trained with simulated annealing (orange/light series) and without it, respectively (blue/dark series). It can be seen that simulated annealing had positive impact on the convergence of global error. Similar deductions as in the case of stratified sampling are concluded – the learning process is able to overcome local extreme and optimize any kind of them.

The best classification results were achieved using the simulated annealing in combination with the stratified sampling method. Classification accuracy of *86.5%* is achieved using the complete NSL-KDD

dataset. This is better than the classification accuracy achieved by any machine learning technique evaluated in the work [16]. In the paper [16], there was achieved classification accuracy of *82.02%* using a NB trees and *77.41%* using a multilayer perceptron ANN.

7. ANN in Real World Network Intrusion Detection

There exists a lot of works aiming on network intrusion detection ([5]-[8]) and intrusion detection in general ([2]-[4], [10], [13], [15], [17]) using various machine learning approaches including ANN.

Lots of these papers present very accurate and promising results. But in fact, there are lacking operational deployments of such systems for network intrusion detection [1]. It should be realized that costs of misclassified data can be very high. A legitimate communication is classified as an attack one in false positive case, and therefore a denial of service occurs. A malicious communication is classified as legitimate one in false negative case, and therefore an evasion occurs. Both cases are very critical. The paper [1] also explains machine learning in intrusion detection as very challenging and completely different than in other areas, whereas, the consequence of misclassified data is not so expensive.

Therefore, we stand for the idea of applying machine-learning-based network anomaly detectors for network monitoring and forensic purposes, rather than intrusion detection.

8. Conclusion

Two categories of intrusion detection approaches were identified: network intrusion detection and general intrusion detection. The typical exemplar dataset of the first category is the DARPA 1999. The representative datasets of the second category are KDD CUP 1999 and NSL-KDD.

This paper covers the second category of general intrusion detection approaches and employs the backpropagation ANN whose learning convergence and classification accuracy are improved by several proposed techniques. The experiments utilizing the stratified sampling and the simulated annealing methods confirm the convergence improvement of the training process of the backpropagation ANN. The best achieved classification accuracy is about 4.5% higher in comparison with the state-of-the-art work presenting the NSL-KDD dataset [16].

Our future work will be aimed on experiments with newer datasets than NSL-KDD 1999. We also plan to apply optimization techniques on datasets containing pure network data in order to perform analysis on the field of network intrusion detection as well.

Acknowledgment

The work was supported by the IT4Innovations Excellence in Science project LQ1602 and the internal BUT project FIT-S-14-2486. We would like to thank our friend Xavier Carpent for discussions and feedback.

References

[1] Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *Proceedings of IEEE Symposium on Security and Privacy* (pp. 305–316).

[2] Naoum, R. S., Abid, N. A., & Al-Sultani, Z. N. (2012). An enhanced resilient backpropagation artificial neural network for intrusion detection system. *International Journal of Computer Science and Network Security, 12(3)*, 11–16.

[3] Guangjun, S., Jialin, Z., & Zhenlong, S. (2008). The research of dynamic change learning rate strategy in bp neural network and application in network intrusion detection. *Proceedings of the 3rd International Conference on Innovative Computing, Information and Control* (pp. 513–513).

[4] Shah, B., & Trivedi, B. H. (2012). Artificial neural network based intrusion detection system: A survey. *International Journal of Computer Applications*, *39(6)*, 13-18.

[5] Zhang, Z., Li, J., Manikopoulos, C., Jorgenson, J., & Ucles, J. (2001). Hide: A hierarchical network intrusion detection system using statistical preprocessing and neural network classification. *Proceedings of IEEE Workshop on Information Assurance and Security* (pp. 85–90).

[6] Manikopoulos, C., & Papavassiliou, S. (2002). Network intrusion and fault detection: A statistical anomaly approach. *Communications Magazine, 40(10),* 76–82.

[7] Labib, K., & Vemuri, R. (2002). Nsom: A real-time network-based intrusion detection system using self organizing maps. *Networks and Security*, 1–6. Retrieved October 23, 2015, from http://citeseerx.ist.psu.edu/ viewdoc/download?doi=10.1.1.135.9256&rep=rep1&type=pdf

[8] Bivens, A., Palagiri, C., Smith, R., Szymanski, B., & Embrechts, M. (2002). Network-based intrusion detection using neural networks. *Intelligent Engineering Systems through Artificial Neural Networks*, *12(1)*, 579–584.

[9] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., & Das, K. (2000). The 1999 darpa offline intrusion detection evaluation. *Computer Networks, 34(4)*, 579–595.

[10] Hawkins, S., He, H., Williams, G., & Baxter, R. (2002). Outlier detection using replicator neural networks. In Y. Kambayashi, W. Winiwarter, & M. Arikawa (Eds.), *Data Warehousing and Knowledge Discovery*, (pp. 170–180). Berlin: Springer.

[11] Hettich, S., & Bay, S. (1999). *Kdd Cup 1999 Data*, *The UCI KD Archive*, University of California, Department of Information and Computer Science. Irvine, CA.

[12] Yamanishi, K., Takeuchi, J. I., Williams, G., & Milne, P. (2000). On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 320–324).

[13] Mafra, P. M., Moll, V., da Silva, F. J., & Santin, A. O. (2010). Octopus-iids: An anomaly based intelligent intrusion detection system. *Proceedings of the International Symposium on Computers and Communications* (pp. 405–410).

[14] Haijun, X., Fang, P., Ling, W., & Hongwei, L. (2007). Ad hoc-based feature selection and support vector machine classifier for intrusion detection. *Proceedings of Grey Systems and Intelligent* (pp. 1117–1121). Nanjing, China.

[15] Ibrahim, L. M., Basheer, D. T., & Mahmod, M. S. (2013). A comparison study for intrusion database (kdd99, nsl-kdd) based on self organization map (som) artificial neural network. *Journal of Engineering Science and Technology, 8(1),* 107–119.

[16] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. *Proceedings of the 2nd IEEE Symposium on Computational Intelligence for Security and Defence Applications.*

[17] Rathore, D., & Jain, A. (2012). Design hybrid method for intrusion detection using ensemble cluster classification and som network*. International Journal of Advanced Computer Research, 2(3).*

[18] Steinwart, I., & Christmann, A. (2008). *Support Vector Machines*, New York: Springer Science & Business Media.

[19] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2012). Nsl-kdd dataset. Retrieved October 23, 2015, from https://web.archive.org/web/ 20150205070216/http://nsl.cs.unb.ca/NSL-KDD

[20] Wang, S. C. (2003). Artificial neural network. *Interdisciplinary Computing in Java Programming*, 81–100, New York: Springer Science & Business Media.

[21] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by backpropagating errors. In T. A. Polk, & C. M. Seifert (Eds.), *Cognitive Modeling* (pp. 213-220). Cambridge: MIT Press,

533-536.
[22] Buscema, M. (1998). Back propagation neural networks. *Substance Use & Misuse, 33(2), 233–270.*
[23] Buckland, M., & Collins, M. (2002). *AI Techniques for Game Programmin*g, Premier press.
[24] Klevecka, I., & Lelis, J. (2008). Pre-processing of input data of neural networks: The case of forecasting telecommunication network traffic. *Telektronikk, 3(4),* 168–178.
[25] Penn State University. *Sampling Theory and Methods.* Retrieved October 19, 2015, from: https://onlinecourses.science.psu.edu/stat506/node/27
[26] Sexton, R. S., Dorsey, R. E., & Johnson, J. D. (1999). Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing. *European Journal of Operational Research*, *114(3)*, 589–601.
[27] Eichenauer-Herrmann, J., & Grothe, H. (1989). A remark on long-range correlations in multiplicative congruential pseudo random number generators. *Numerische Mathematik, 56(6)*, 609–611.

Ivan Homoliak was born in Rimavska Sobota, Slovak Republic, in 1987. He received his B.S. in information technology and M.S. in the field of network security from Brno University of Technology (BUT), Czech Republic, in 2007 and 2012, respectively. Currently, he is a Ph.D. student at Faculty of Information Technology (FIT), BUT. His current research interests include intrusion detection in obfuscated network traffic, network security, data mining and soft computing.

Dominik Breitenbacher was born in Ostrava, Czech Republic, in 1988. He received his B.S. in information technology and M.S. in information technology security from BUT, Czech Republic, in 2012 and 2015, respectively. Currently, he is a Ph.D. student at FIT, BUT. His dissertation theme is on intrusion detection systems for embedded and mobile devices.

Petr Hanacek was born in Uherske Hradiste, Czech Republic, in 1964. He obtained his M.S. in computer engineering, Ph.D. in computer science and habilitation at BUT in 1988, 1997 and 2003, respectively. He leads the local security research group Security@FIT which focuses on research in the field of computer and network security.

He currently heads the Department of Intelligent Systems and works as associate professor on FIT, BUT, Czech Republic. Since 1987 to 2001, he worked at the Department of Computer Science at Faculty of Electrical Engineering and Computer Science, BUT. Since 2002, he works at FIT, BUT.

Dr. Hanacek is a member of Czech and Slovak Information Society (CIS), Czech & Slovak Simulation Society (CSSS) and member of Special Interest Group on Security, Audit and Control (ACM – SIGSAC).