Evolutionary Optimization of On-line Multilayer Perceptron for Similarity-Based Access Control

Andrii Shalaginov
Norwegian Information Security Laboratory
Center for Cyber- and Information Security
Norwegian University of Science and Technology
andrii.shalaginov@ccis.no

***Abstract*—Neural Networks have been successfully used in different fields of Information Security such that network intrusion detection and malware analysis because of ability to provide high level of abstraction for complex and incomplete data. Despite its successful application as off-line learning method, the on-line learning can be challenging when dealing with data streams. This paper presents an ongoing research on on-line Neural Network for Access Control. It can be used for similarity-based access to sensitive information. Conventional training is not efficient when dealing with data streams such that access patterns flow since the availability of the data samples is limited. Considering this obstacle we proposed to use Genetic Algorithm as meta-heuristic optimization in selection of individual training rates α for each weight. Similarity-based Access Control mechanism deals with a data stream that includes continuous flow of attributes characterizing user and resources, so the task is to estimate the likelihood of legitimacy of user accessing a particular resource in dynamic environment. This research contributes to the field of Information Security by overcoming the limitations of data stream mining in agile environment.**

I. Introduction

Artificial Neural Network (ANN) is one of the most powerful methods capable of modelling complex non-linear relations between the input data and output decision. It has been successfully applied for a number of fields in Information Security such that Intrusion Detection [1] and web attacks classification [2]. Access Control (AC) is one of the most important tasks of Information Security. Modern AC models and policies are based on a specifically predefined set of rules for a single user or group of users. The challenge comes when dealing with agile environment like in grid-systems or cloud environment according to Bedi et al. [3], making traditional access methods less efficient. In this work we target the on-line learning of AC mechanism based on the flow of access logs data. Despite the wide applicability of Multilayer Pecreptron (MLP), a type of ANN, usage in the data streams mining brings additional limitations like availability of data in a very short time period. However, MLP is capable of providing fast response [4]. So, we believe that optimization of MLP learning may facilitate similarity-based AC.

AC is intended to limit access using different policies and models and to prevent an unauthorized access. Sahafizadeh et al. [5] provided a comprehensive overview of modern AC models. It studied Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), etc. MAC relies on levels of authorization or class of a user when evaluating its ability to perform operations on the object. DAC provides a specific detail for each particular user or user role of where she is capable of going. RBAC uses an access matrix of objects and subjects in order to define an access rule. ABAC [6] is one of the initiatives by NIST for moving from Role-Based Access Control to more flexible evaluation of the asset's attributes. Vincent [7] in the NIST guide defined a scenario that includes access control policy, environment conditions, and Subject and Object with corresponding sets of attributes. Similarity-Based Access Control (SBAC) defined in the patent by Farber et al. [8] allowed to use similarities in access attributes. Considering this, we can say that the application of Soft Computing can facilitate AC and is capable of learning on-line since off-line may require significant resources when the organization size is large. There have also been some attempts to apply Machine Learning methods. The report [9] from NIST proposed a Risk-Adaptive Access Control model (RAdAC) that uses historical records to determine whether access should be granted or not. In addition, it was noted that Machine Learning, Evolutionary Computing (EC) in particular, can be included in RAdAC to improve the model. Furthermore, Bedi et al. [3] explored a way of applying ANN in the access of grid resources. In particular, requests for resources are classified via Radial Basis Function Neural Networks, because of non-linear ANNs superior generalization. However, to author's knowledge ANN has not been widely used in the AC models.

This research presents an improvement in the MLP learning process to be reliably used for data streams mining in AC. The proposed approach applies EC in order to train the MLP with help of EBP with respect to the need for fast data processing. The Genetic Algorithm (GA) [10] was chosen to provide a less complex solution during MLP learning rather than using the quadratic programming for constrained optimization as presented by Sheta et al. [11]. We also compare performance of a conventional Back-Propagation method with fixed-α, one optimized by Golden Section Search and a proposed method. Unlike Kanada [12], we target optimization of each individual α for weights update. The remainder of this paper is organized as following. Section II describes existing challenges in data streams mining by MLP and common ways of models learning. Section III presents proposed optimization of on-line learning. Section V provides an overview of the setup of the experiment, data utilized, and analysis of the results. Lastly, Section VI gives conclusions.

II. Background

As mentioned earlier, Similarity-Based AC is affected by velocity and veracity of the access traces appearing in access logs. It means that the data are available for a short time frame and should be processed in a fast *on-line* way rather than an iterative *off-line* [13]. *Data streams mining* is a special field that defines such on-line models [14]. From the perspective of Information Security, it be events monitoring, traffic processing, etc. in addition to access logs analysis [15]. So, the challenge with data stream mining can be formulated as following. At a time t_i there is happening some non-deterministic event T_i such that entering user credentials to access the resource, which also can be influence by some covert action or can be completely random. Each user X_i can be described by a set of M properties (features) $X_i = \{A \in R^M\}$, where features $a = \{a_0, \cdots, a_M\}$ can be either user- or resource-specific. So, the goal is to predict the class Y_i of this event, which defines that actions to be undertaken (*"allowed"* or *"blocked"*). This has to be done using previously collected logs and established access policies over some past time t as it is shown in the Figure 1.

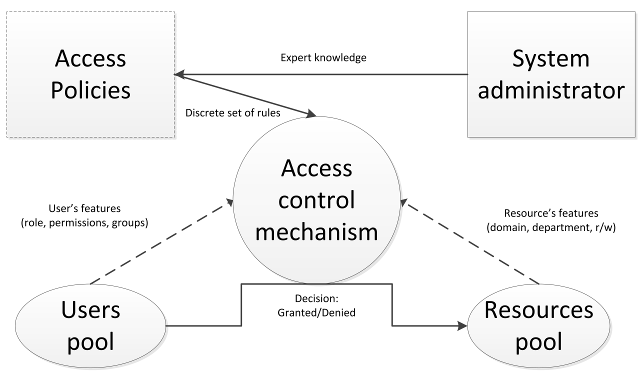


Fig. 1: A general way of how the Access Control mechanisms interact with objects and subjects according to ABAC [7]

In real world tasks the amount of statistics is huge and therefore it is not an option to re-learn the model each time when new event T_{i+1} arrives. At this point of view we concentrate on the data sample that have some predefined set of features A, where each feature is a numerical $\forall a_j \in A : a_j \in R$. The values of each feature a_j are unknown from before. Similarly the combination of the features in the given data sample $X_1, \cdots, X_n$ in this one from access control system sample that have not been appeared yet. Moreover, there is a need to determine a class Y_i of the given sample $X_i|_{i=1,\cdots,N}$, where N is a size of training data.

A. Multilayer Perceptron Learning

ANN is one of the most powerful ML methods capable of learning from erroneous, complex and incomplete data. As mentioned above, the training can be done either off-line or on-line. In this work we target *on-line* learning since the model should be capable of adjusting the parameters of the model from data when new sample comes.

Definition 1: MLP training is done via minimization of the objective function of the error signals $E(W)$:

$$E(W) = \frac{1}{2}(y - d)^2 \tag{1}$$

where d - desired output of the MLP and y - actual output and W is a set of all weights in MLP. The main obstacles in learning is that the method can stuck in a local optima unless the learning rate is optimal one. So, the primary optimization problem in the MLP is minimisation of the function:

$$min\ E(W)\ |_{W \in \mathbb{R}^M} \tag{2}$$

where each function in the $E(W)$ is an objective function of the neuron's weights w_i^j ($j-th$ hidden layer and $i-th$ hidden unit) that should be optimized with the following condition on the whole domain of the function $dom\ E(W) = \mathbb{R}$ and with respect to the learning rates α:

$$\forall\ w_i^j\ \in\ \mathbb{R}:\ E({w_i^j}^*)\ <\ E(w_i^j) \tag{3}$$

According to Heskes et al. [16], *on-line* learning introduces a new challenge in adapting to new data that arrive. Saad et al. [17] presented an analysis describing the on-line learning within MLP, where the authors focused on the dynamic evolution of the error function. Recently, there has been research on improvement of a supervised learning in back-propagation MLP as well as the creation of new methods for it according to Heskes et al [16] and Mandic et al. [18]. Researchers have proposed different learning schemes including complex algorithms like Widrow-Hoff LMS and Adaline as described by Widrow et al. [19]. The MLP learning process should be optimized as a differentiable error function, and then the Gradient Descent (GD) optimization of the function tunes the weight of the neurons:

$${w_i^j}^{new} = {w_i^j}^{current} - \alpha \cdot \nabla E(w_i^j) \tag{4}$$

where $E(w_i^j)$ is a multidimensional error function over a weight w_i^j. The principle of the learning then is to use so-called Delta Learning rule, comparing the output of the network against the labelled dataset. Delta Learning Rule makes a robust first-order approximation as stated by McClelland et al. [20] along the partial derivative direction only in case the learning rate is less than or equal to the optimal one, also described by Mandic et al. [18]. In real-world tasks however it is hard to predict how the optimal learning rate will change under the influence of an input concept drift.

The determination of the learning rate α brings with it the most challenge. There are several options for definition such as constant rate or iterative adjustment [10]. However, the data sample are available for a short period of time and data flow has unpredictable dynamic nature, and according to Wilson et al. [21], on-line training gives a faster convergence than batch training. So, the commonly used constant α is not suitable.

B. Error Back Propagation

The most commonly used method for ANN training is Error Back Propagation (EBP) that is based on the assumption that error function $E(W)$ can be reduced by using gradient measure to find optimal weights.

Definition 2: The generalized EBP learning rule for back propagation is defined as following referring to GD method:

$$w_i^{j^{new}} = w_i^{j^{current}} - \alpha \cdot \delta_i^j \cdot g(h_i^j) \tag{5}$$

where α - fixed learning rate, $h_i^j = \sum w \cdot h$ - sum of the weighted signals from the previous neurons layer, $g(h_i^j)$ - sigmoid activation function of the neuron or value of the x_i in case if the layer is initial one. Moreover, depends on the layer the value of the error signal will be calculated respectively for the intermediates layers:

$$\delta_i^j = \frac{1}{1+e^{-h_i^j}} \cdot (1 - \frac{1}{1+e^{-h_i^j}}) \cdot \sum w_i^j \cdot h_i^j \tag{6}$$

and corresponding output layer:

$$\delta_{output} = g'(h_i^j) \cdot (d-y) = \frac{1}{1+e^{-h_i^j}} \cdot (1 - \frac{1}{1+e^{-h_i^j}}) \cdot (d-y) \tag{7}$$

Originally, EBP makes it difficult to train each layer of the MLP since the exact values of the output of each hidden layers are not known since each layer learns more of the abstract meaning of the input data and the complexity of the model becomes higher with each new layer. So in general, the derivative of the error function $E(w_i^j)$ is calculated on each hidden layer.

The GD-based methods have been extensively studied recently. With respect to the optimization approach, there are several other valid ways of finding optimal weights in MLP. The stochastic GD originally used for optimization has a low speed of convergence and is therefore the baseline. In order to improve learning, one may use the Conjugate Gradient Descent and Newton-Raphson as studied by McAllester [22], where Hessian matrices need to be evaluated. On the downside, this approach requires additional algorithmic step to find an optimal direction and second-order derivatives, since the values of the derivatives are unknown and require multiple error functions evaluation during each epoch as result. On the other hand, conventional MLP learning principle is not suitable for the data streams mining since it provides an offline learning over a given finite sample G. The stand-alone EBP method is high resource-consuming in the tasks related to Big Data with respect to the speed of on-line processing.

C. Existing ways of α optimization for weights calculations

According to literature review, there can be named several approaches to weight optimization in the Eq. 5: (i) static α with gradient information, (ii) iterative adjustments of α using optimization, (iii) higher dimensions factorization for faster convergence. In this work we consider the 2^{nd} option since it is less computationally expensive than 3^{rd} and more efficient than 1^{st}. According to Luenberger et al. [23], line search along the gradient direction is the most promising method for iterative rate optimization since each weight can be optimized separately. In our view this is most promising approach in data streams mining eliminating a need for expensive computations. Visibly, the learning rate α must be either defined empirically or using some one-dimensional optimization. According to Roy [24] in most of the related studies the authors used as some predefined values or try several empirical values. Since α in most cases is a constant, it causes aggregation of the error and slow convergence when the value is not optimal. Generally speaking, the α can be optimized in different ways such as the *adaptation of a learning rate* or *alteration of the gradient values*. Kandil et al. [25] proposed to use time-varying learning through the linearization of the whole network, requiring additional expensive computation of matrices. This method uses a binary mapping scheme for computing the optimal learning rates for each of the layers. Yu et al. [26] described several approaches using first- and second-order derivatives for optimal α and momentum calculations, which are used to adjust the weights. At the same time Tielman et al. [27] suggested to use mini-batches and moving average of the squared gradient for each weight optimization. However, it requires keeping in memory a set of mini-batches with corresponding moving average parameters in addition to a very slow alteration of a learning rate. Kingma et al. [28] optimized α using similar weighting on a momentum estimations. Authors suggested to use decay parameters β, which requires additional empirical estimations. Further, Plagianakos et al. [29] suggested the use of pair-weights adjustments based on the previous epoch. On the other hand, the one-dimensional optimization like Golden Section Search (GSS) [23] can be considered one of the simplest and more effective approach than the application of Quadratic Programming. Therefore, GSS can be used in EBP to find an optimal rate α in an optimal region rather than some fixed starting point.

Definition 3: Golden Section Search is used to find the most promising value of α in the interval $[\alpha_{min}; \alpha_{max}]$ by means of following iterative process with an error threshold ϵ:

$$\alpha_1 = b - \frac{b-a}{\phi},\ \alpha_2 = a + \frac{b-a}{\phi} \tag{8}$$

where initially $a = \alpha_{min}$, $b = \alpha_{max}$ and $\phi = \frac{1+\sqrt{5}}{2}$. The Equation 8 defines an iterative procedure, which followed by the evaluation of $E(W - \alpha_1 \cdot \nabla E(W))$ and $E(W - \alpha_2 \cdot \nabla E(W))$. Each iteration ends with $a = \alpha_1$ or $b = \alpha_2$ until criteria $|a-b| > \epsilon$ is satisfied.

Another approach to optimization of α using EC proposed by Kim et al. [30]. Some authors target specifically generation of weights using EA rather than learning. Ding et al. [15] showed how the EC can be used for weights allocation. Jie et al. [31] stated the importance of GA since it is able to produce generalized model and improve learning. Further, Islam et al. [32] specifically mentioned applicability of GA to optimized a number of hidden neurons and layers towards improvement of classification accuracy. Finally, Kanada [12] emphasized the importance of GA in learning rate optimization, where learning rate is increased or decreased with respect to previous epoch according to geometric regression. Another work of interest is by Sajan et al. [33] where authors optimized number of nodes in addition to a single value of learning rate. In this work we will target global optimization of each individual learning rate independent from previous values.

III. A new method of on-line MLP training using Genetic Algorithm

Despite the successful application of EBP, there are multiple possibilities as to why this method fails to learn properly form the given data. (i) There may be several occurrences of local minima. (ii) The wrong placement of pre-defined set of initial parameters such as learning rate α will mislead the optimization procedure. Thus, one can highlight the following difficulties related to usage of EBP: convergence is not guaranteed, can be slow, and depends on the input data parameters. (iii) The challenge of training on-line model from the data within as low a number of iterations is apparent. From the literature we can see that EC has been applied for stochastic optimization in MLP before to reduce convergence time. It is also vital that the decision is made in a short period of time, because otherwise it can cause privacy breaches due to unreasonable delays in re-training an AC mechanism. In contrary to previous works [33], [12], [32], [31], [15] we suggest to apply optimization procedure for individual α.

A. Single-step on-line learning of MLP

The aforementioned optimization problem in a single-step online MLP learning is caused by non-linearity and a high level of abstraction.

Lemma: *The error function $E(W)$ in a single-step MLP is non-monotonic with multiple extreme points due to several layers of non-linearity introduced by nested hidden layers. As result conventional GD-based optimization methods may fail to find a global optimal set of weights W.*

The learning rate α in weights adjustments shall not be constant on every iteration, so that it will result in a decrease of the $E(W)$ on each learning iteration faster. Furthermore, each of these adjustments steps consist of an additional unconstrained optimization procedure that is aimed to find a corresponding optimal α by means of meta-heuristic real-valued GA. The GA enables multiple hypothesis evaluation at the same time since the problem is to find an appropriate value of α. This is done towards hardening the robustness against non-deterministic patterns in the access sequence. We consider MLP, which based on differentiable sigmoid activation function [10]. It can be seen that the error function $E(W)$ is a non-linear one that includes recursive additive composition of the neurons activation functions for each given labelled dataset:

$$E(W) = \frac{1}{2} \cdot (d-y)^2 = \frac{1}{2} \cdot (d - \frac{1}{1+e^{-h_i^j}})^2, \tag{9}$$

Remark. The complexity of the function $E(W)$ will grow with the number of hidden layers. So, the function will have a corresponding recursive form for each of the given labelled data samples in single-step approach as shown in the Eq. 10.

$$\begin{aligned} E(W) = \frac{1}{2} \cdot [d - \frac{1}{1+e^{-h_i^j}}]^2 = \\ \frac{1}{2} \cdot [d - g(\sum_{i=0}^{M-1} w_i^j \cdot \frac{1}{1+e^{-h_i^j}})]^2 = \cdots = \\ = \frac{1}{2} \cdot [d - g(\sum_{i=0}^{M-1} w_i^j \cdot g(\ldots w_i^{j0} \cdot g(\sum_{i=0}^{M-1} w_i^0 \cdot x_i)\ldots))]^2 \end{aligned} \tag{10}$$

The function in Eq. 10 represents an error surface $E(W)$ that has a non-linear dependency with respect to a set of neuron weights W. From this perspective, influence of the weights in the initial layer will have a bigger degree of non-linearity than the next higher layers. Rojas [13][Chapter 7] studied a similar example of the error function and depicted a possible local minima that affects the optimization. Therefore, there are multiple challenges for conventional GD optimization possibly resulting in local optima solutions [15]. Multiple plateaus and local minima make it unlikely to achieve the global minima, so GD will be stuck in a sub-optimal during weights update process. In fact, for an arbitrary weight w_i^j of the layer j the next function's $E(W)$ limit will have a place as shown in the Eq. 11 considering MLP error function described in the E. 10. It is based on the limit's property of the composite continuous functions $\lim f[g(x)] = f[\lim g(x)]$ according to Stein [34] since the sigmoid function $g(h_i^j)$ is a continuous and differentiable function. Moreover, the neighbourhood of the global minima of the derived limit is inside the tolerance interval $L \pm \epsilon$ for the given neighbourhood of the optimal value of the weight $w_i^{j\,optimal} \pm \delta$ as studied by Exner [35] while continuously changing the input data sample X. Also, we can see that the nested combination of multiple monotonically increasing sigmoid functions will result in number of local optima, rather than one global one. Such combinations of the sigmoid function will give a complex non-linear high level abstraction for the same input data pattern and different weight values. So, it is nearly impossible to define the exact value of the limit for the global extreme point neighbourhood since the resulting limit is a complex one and vector of the weights needs to be closer to global optima. However, the limit is not a constant value since there is a constant concept drift and each iteration in data stream mining. As result, the neighbourhood of the global optima $w_i^{j\,optimal}$ is changing stochastically with a new data sample coming, which means that the error function $E(W)$ has an inconsistent global optima region making usage of the fixed-α method less efficient.

$$\begin{aligned} \lim_{w_i^j -> w_{optimal}} E(W)|_{X=constant} = \\ \lim_{w_i^j -> w_{i\,optimal}^j} \frac{1}{2} \cdot [d - g(\sum_{j=0}^{M-1} w_i^j \cdot g(\ldots w_i^0 \cdot \\ g(\sum_{i=0}^{M-1} w_i^0 \cdot x_i)\ldots))]^2 \approx \\ \approx \frac{1}{2} \cdot [g(\ldots \lim_{w_i^j -> w_{i\,optimal}^j} g(\sum_{i=0}^{M-1} w_i^0 \cdot x_i)\ldots)\ldots]^2 \\ \in [L \pm \epsilon] \end{aligned} \tag{11}$$

So, there is a need to apply more advanced techniques for the weights optimization rather than conventional one like constant or increasing learning rate α. $E(W)$ is located within some ϵ of the L meaning that for different input X the global optima will be different. For our purpose we use sigmoid activation function for each neuron since it is differentiable [36] and most suitable for learning in case of two-class problems (*"denied"* or *"allowed"*). The main point of the optimization is to find an optimal set of weights W that gives the lowest possible value of the error function $E(W)$ during single-step learning. Therefore, it is unacceptable to apply purely unimodal optimization and line search because they are exposed to premature convergence according to Salomon [37]. Unimodal heuristic optimization has monotonicity as its necessary criteria according to Doerr et al. [38], yet this is not achieved. As result, found solution will be a local one.

B. An optimal individual learning rate α prediction using Genetic Algorithm

Our method targets usage of individual optimal α for each of the weights on each layer that makes MLP to converge faster than conventional unified fixed-α or deterministic decreasing/increasing α approaches. Additionally, the single-step on-line learning is applied since the availability of data samples for training is limited. For the on-line incremental learning the α has to be optimal on each step for eliminating accumulation of the errors residuals. So, meta-heuristic EC tends to solve the problem more reliably and quickly when classical unimodal search methods are slow.

Proposal: *Evolutionary Computing, Genetic Algorithm in particular, is applied as one-dimension optimization in a single-step MLP learning to facilitate a proper individual α-determination for each particular weight w_i^j optimal update.*

Non-monotonic function such as $E(W)$ has lower chances to be optimized due to multiple extreme points, and we therefore consider EC methods as the most promising for such tasks. MLP provides nested optimization problem. The *primary* problem includes seeking for optimal weights as shown in the Figure 12, which gives a value of the error function referring to the Equation 2.

$$min_{W \in \mathbb{R}^M} \ E[W - \alpha \cdot \nabla E(W)] \tag{12}$$

The *secondary* optimization problem is to find a set of optimal steps α as shown in the Figure 13, which will result in a global optimal solution for single-step algorithms to avoid long iterative learning. However, at this point, we need to consider each individual α_{ij} that employs mutation and crossover operations to be optimized by GA, which increases the chance to cover as much search space as possible while keeping the weights constant.

$$min_{\bar{\alpha} \in \mathbb{R}^M} \ E[W - D(\bar{\alpha}) \cdot \nabla E(W)] \mid_{W=const} \tag{13}$$

where D - is a diagonal matrix.

Remarks. The surface of the error function has a non-linear dependency on the weights values, which means that the weights update process has to be performed by meta-heuristic optimization methods in order to avoid premature convergence. Conventionally, GSS or similar line search methods are applied as a unimodal optimization. But it does not work well without reliable information about the borders of the optimal learning rate and is usually very slow. Naturally, GA will make it converge faster.

C. Proposed methodology

Under the aforementioned constraints in on-line incremental learning we proposed to use single-step MLP solving the nested optimization problem to find an optimal set of individual α for weights update. As result, the weights are only corrected based on the optimal learning rate. Additionally, it is important that toward privacy protection its application maintains the trade-off between speed and reliability of the answer. The algorithm of the proposed method is defined in the Listing 1. The conventional Error Back Propagation method was modified accordingly. For this algorithm we make the assumption that each hidden layer has the same dimensionality as input data vector. The computational complexity will be as following. For each neuron's weight there will be $pop\_size(\cdot p_{crossover} + p_{mutation}) \cdot N_{epochs}$ recalculation of the fitness function in a worth-case scenario. However, this can be decreased by introducing the memory of the fitness function values. Additionally, there is a chance of getting to the sub-optimal or optimal $\alpha_{optimal}$ within fewer amount of epochs in comparison to the Brent's Method (Golden Section Search) or Fibonacci method since they have rather linear convergence according to Press [39].

The real-valued does not requiring additional binary mapping schemes, and therefore the α values are used as chromosomes and the error function $E(W)$ is defined as a fitness one. The corresponding *Arithmetic Crossover* was performed for the crossover operation as discussed in the paper by Kksoy et al. [40]. Additionally, the mutation was done as a real-valued random uniform mutation of a chosen chromosome as shown by Adewuya [41]. Here we concentrated on deriving as closest to optimal learning rate α as possible in a short time frame. Information Security tasks often put speed and response time constraints on hard computational tasks rather than constraints on answer precision.

IV. Experimental design

We performed experiments having a data stream with limited availability of the data samples for training.

Dataset. The Kaggle Amazon Employee Access Challenge [42] was used as an access log to test the proposed method. It consists of train 32,769 data records characterized by 9 integer-valued feature as access requests: ACTION, RESOURCE, MGR_ID, ROLE_ROLLUP_1, ROLE_ROLLUP_2, ROLE_DEPTNAME, ROLE_TITLE, ROLE_FAMILY_DESC, ROLE_FAMILY, ROLE_CODE. ACTION is a binary class label: 0 - action against resource is denied (1,897), 1 - action is approved (30,871). Since the original labelled testing dataset of the challenge is not

Algorithm 1 Optimization of α-rate in single-step MLP training using real-valued GA

$x \leftarrow x_{new}$
$\overline{\overline{w}} \leftarrow \overline{\overline{w_{previous}}}$
$\delta_{output} = g(h) \cdot (d - y) \cdot (1 - g(h))$
for all hidden layer **do**
  for all neurons in a current hidden layer **do**
    $\delta_i^j \leftarrow g(h) \cdot (1 - g(h)) \cdot w_i^j \cdot \delta_{output}$
    initialization(α_{random},pop_size);
    while $N_{epochs} < N_{epochsMax}$ or $|Fit_k - Fit_{k-1}| < \epsilon$ or $|Fit^{'}| < \epsilon$ **do**
      MUTATION($p_{mutation}$,α)
      CROSSOVER($p_{crossover}$,$\alpha 1$,$\alpha 2$)
      SELECTION(α);
      $\alpha_{optimal} \leftarrow \alpha_{selected}$
    end while
    $w_{ij} \leftarrow w_i^j + \alpha_{optimal} \cdot \delta_i^j \cdot g(h_i^j)$
  end for
end for
return $\overline{\overline{w}}$
function MUTATION($prob_{mutation}$,α)
  $d \leftarrow random(0, 1)$ (generation of a real number)
  $\alpha_{mutated} \leftarrow \alpha_{mutate} \pm d$
  return $\alpha_{mutated}$
end function
function CROSSOVER($prob_{crossover}$,$\alpha 1$,$\alpha 2$)
  $d \leftarrow random(0, 1)$ (generation of a real number)
  $offspring1 \leftarrow d \cdot y_i + (1 - d) \cdot x_i$
  $offspring2 \leftarrow d \cdot x_i + (1 - d) \cdot y_i$
  return $\alpha_{off1}, \alpha_{off2}$
end function
function SELECTION(α)
  $Fit \leftarrow E(W)|_{\alpha}$
  return $\alpha_{optimal}$
end function

published, we decided to use the model by Duan et al. [43] with the performance of AUC = 0.92360 as a reference testing dataset that consists of 58,921 test data sample.

Performance Evaluation. For the comparison of the implemented model on the testing data, several metrics were used such as Mean Absolute Error (MAE), Root Relative Squared Error (RRSE) and Root Mean Square Error (RMSE) considering that data streams mining model is regressional according to Bifet [44].

$$MAE = \frac{1}{N} \cdot \sum_{i=0}^{N-1} |y_i - d_i|$$
$$RRSE = \sqrt{\frac{\sum_{i=0}^{N-1} (y_i - d_i)^2}{\sum_{i=0}^{N-1} (y_i - \bar{d})^2}} \tag{14}$$
$$RMSE = \sqrt{\frac{1}{N} \cdot \sum_{i=0}^{N-1} (y_i - d_i)^2}$$

MLP Configuration. For proof of concept demonstration we used only 3 layers out of 6 layers according to "Rule of thumb" the optimal $N_{hidden} = \frac{2}{3} \cdot (N_{In} + N_{out})$ and a generally accepted fixed learning rate $\alpha = 0.3$ was defined. Also the amount of epochs for the off-line learning was defined as 10 and 1 for the optimized single-step model. The Figure 2 shows the experimental design using the proposed method.

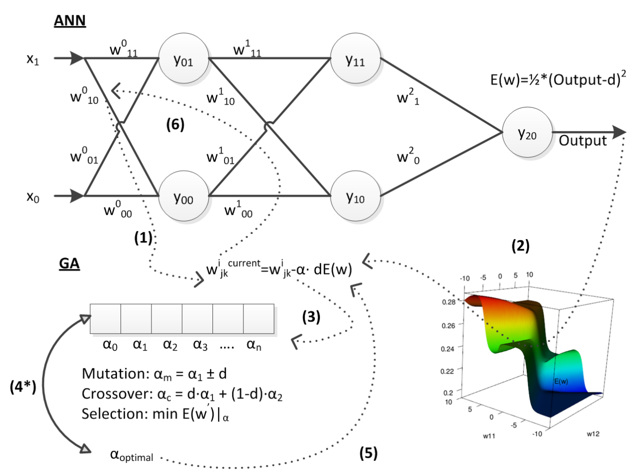


Fig. 2: Proposed method for single-step on-line learning

V. Results and Discussions

Our assumption was that there exist a set of pre-defined access policies and known access patterns by selecting a specific fraction of the data as training before data stream is fed to a model. The following experiments were conducted. (i) 100 samples are used for training followed by data stream of test data. (ii) 1,000 samples are used as training data and then testing is done for the new samples from the data stream. Also we performed standard batch learning and cross-validated the accuracy of implented method in C++ in comparison to community-accepted implementations in Weka [45] and RapidMiner [46] as shown in the Table I. Optimized model used 10 iterations of GA with 10 chromosomes in each population and 4 mutation and 7 crossover operations. Other MLPs used 100 epochs. The result look consistent. Generally, RRSE is a description of mean prediction. In this case we can see that the mean is biased towards approved action "1" since logically the majority of the access patterns in a system are legit. Therefore, all implementations show similarly high value of RRSE.

Method	MAE	RMSE	RRSE, %
MLP impl.	0.061	**0.140**	**100.849**
MLP impl. + GA	**0.054**	0.142	102.665
Weka MLP	0.061	0.149	107.554
RapidMiner MLP	0.059	0.151	108.700

TABLE I: Performance of implementations on static dataset

A. *Similarity-based Access Control performance*

We perform single-step on-line learning of the MLP as well as cross-validation of incoming data in the data stream S.

The experiment was built as following: the boundaries of α for GSS and GA were chosen to be $[0, 1]$. The MLP started from randomly initialized weights and trained with $l\%$ sample from the training dataset. The stream of $k\%$ samples from the classified test dataset is then fed to the model. Moreover, the train dataset constantly trains the model while new test data samples arrive. Finally, earlier-defined performance metrics are computed over the classified samples from the test dataset. The Table II represents the results for the on-line scenario. GSS method used the same number of iterations as number epochs in the GA. Original MLP without optimization trained 10 epochs, and MLP with optimization only one.

TABLE II: Performance comparison of MLP on test dataset in on-line incremental learning using optimized and non-optimized techniques in data stream scenario

Method	**MAE**	**RMSE**	**RRSE, %**
start with 100 pre-training samples			
MLP (1 epoch)	0.065	0.180	36.117
MLP (10 epochs)	0.070	0.179	35.933
MLP + GSS (1 epoch)	0.060	0.173	34.682
Proposed	**0.054**	**0.167**	**33.567**
start with 1000 pre-training samples			
MLP (1 epoch)	0.056	0.155	32.184
MLP (10 epochs)	0.057	0.155	32.183
MLP + GSS (1 epoch)	0.052	0.150	31.210
Proposed	**0.038**	**0.140**	**29.078**

The results show that proposed method gives better accuracy on all performance metrics for 100 and 1,000 access log samples that were available for the initial pre-training. However, such application of GA for on-line MLP optimization has a cost of higher computational complexity than simple utilization of constant α. Therefore, we believe that parallel optimization can help to achieve fast processing speed since sequential execution will be much slower than standard MLP. This can be done using modern CPU and GPU.

B. Error surface influence on MLP training

The fitness function used in GA is basically the same as error function in the MLP $E(W)$. The derivative on each step is estimated in the neighbourhood of the corresponding neuron weight w with precision $h = \pm 10^{-6}$ in error cost function $E(W)$ while keeping constant all other weights and the input data sample. To study dependency between the weights change and error function it was chose two arbitrary weights on the lower layers (w_3^1 and w_9^1) in the 3-layers perceptron. Figure 3 shows the surface of the error function for two arbitrary selected weights. Unless some sophisticated learning rate updating methods and stopping criteria are used, the EBP will converge to a sub-optimal solution.

The Figure 4, (a) represents an example of path of both weights along 1,000 iterations until it gets converged. The learning rate was constant, so it can be seen that the path is gradually following one direction. On the other hand, Figure 4, (b) shows that proposed method has a broader coverage of the search space bringing a higher chance of getting an optimal solution for the non-linear error function.

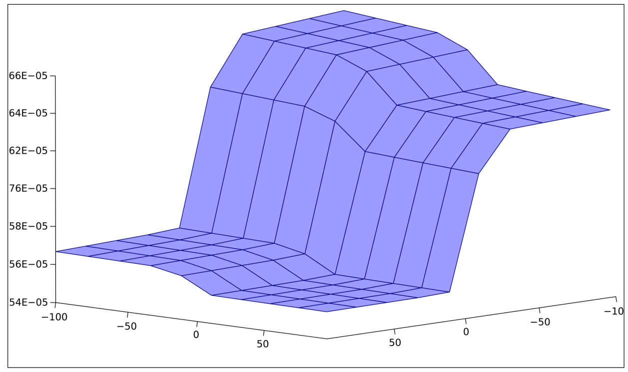
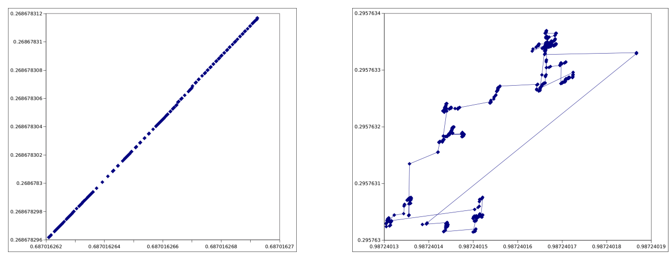


Fig. 3: Surface of the error function showing dependency of $E(W)$ on w_3^1 and w_9^1 as covariates in 3-layers MLP that was trained from the given dataset



(a) Conventional MLP training with a constant learning rate α

(b) Proposed method for individual learning rate α optimization

Fig. 4: Path traverse of the weights w_3^1 and w_9^1 in MLP

Considering smooth transition between minima and neighbourhood of the error function in the Figure 3 we can see that proposed method will likely result in a better solution rather than training ML with a constant step. Therefore MLP with the higher number of layers can be trained using adoptive rate α approach faster rather than fixed-rate iterative line search.

VI. Conclusions

Conventional off-line learning of Multilayer Perceptron is no more reliable for data streams mining since it requires more iterations to learn and does not converge quickly. As result, it cannot be successfully applied for on-line training of Similarity-Based Access Control models, where the attributes of resource and user can be utilized to evaluate whether similar users may access similar resources. To overcome this limitation, a single-step learning of Multilayer Perceptron can adjust the model once new access request come and does not require the constant access to that pattern. In this work Evolutionary Computing method was utilized, Genetic Algorithm in particular. This method is considered to be more robust in environment with a consistent drift of the data statistics and non-deterministic events. The influence of the layers weights on the error function was studied also and we can state that due to non-linearity it requires better optimization.

Therefore, we proposed how the appropriate learning rate α calculation can be done using Genetic Algorithm for each neuron in accordance to an optimal solution on each layer. This illuminates a need for constant iterative batch learning of the Multilayer Perceptron, allowing model to be a single step. One of the main benefits of the proposed method is that it scales and can be used for larger non-linear Neural Networks.

REFERENCES

[1] J. Planquart, "Application of neural networks to intrusion detection. sans institute," 2001.

[2] A. Shalaginov and K. Franke, "Towards improvement of multinomial classification accuracy of neuro-fuzzy for digital forensics applications," in *15th International Conference on Hybrid Intelligent Systems (HIS 2015)*, vol. 420, no. 1. Springer Publishing, 2015, pp. 199–210.

[3] P. Bedi, B. Gupta, and H. Kaur, "Access control on grid resources using radial basis function neural network," *Procedia Technology*, vol. 4, no. 0, pp. 336 – 341, 2012, 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012.

[4] Z. Salek, F. M. Madani, and R. Azmi, "Intrusion detection using neuarl networks trained by differential evaluation algorithm," in *Information Security and Cryptology (ISCISC), 2013 10th International ISC Conference on*, Aug 2013, pp. 1–6.

[5] E. Sahafizadeh and S. Parsa, "Survey on access control models," in *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, vol. 1, May 2010, pp. V1–1–V1–3.

[6] "Attribute based access control (abac) - overview," http://csrc.nist.gov/projects/abac/, accessed: 27.05.2016.

[7] V. C. Hu, D. Ferraiolo, Rick, Schnitzer, Adam, Sandlin, Kenneth, Miller, Robert, Scarfone, and K. Scarfone, "Guide to attribute based access control (abac) definition and considerations," National Institute of Standards and Technology, Guide, 2014.

[8] D. Farber and R. Lachman, "Similarity-based access control of data in a data processing system," May 17 2011, uS Patent 7,945,544.

[9] "A survey of access control models," NIST, Tech. Rep., 2009.

[10] I. Kononenko and M. Kukar, *Machine learning and data mining: introduction to principles and algorithms*. Horwood Publishing, 2007.

[11] T. H. Sheta A, "A comparision between genetic algorithms and sequential quadratic programming in solving constrained optimization problems," in *Artificial Intelligence Machine Learning (AIML)*, vol. 6, no. 1, 2006, pp. 67–74.

[12] Y. Kanada, "Optimizing neural-network learning rate by using a genetic algorithm with per-epoch mutations," in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 1472–1479.

[13] R. Rojas and J. Feldman, *Neural Networks: A Systematic Introduction*. Springer Berlin Heidelberg, 2013.

[14] Y. Yamamoto and K. Iwanuma, "Online pattern mining for high-dimensional data streams," in *Big Data (Big Data), 2015 IEEE International Conference on*, Oct 2015, pp. 2880–2882.

[15] S. Ding, H. Li, C. Su, J. Yu, and F. Jin, "Evolutionary artificial neural networks: a review," *Artificial Intelligence Review*, vol. 39, no. 3, pp. 251–260, 2013.

[16] T. M. Heskes and B. Kappen, "On-line learning processes in artificial neural networks," 1993.

[17] D. Saad and S. A. Solla, "On-line learning in multilayer neural networks," in *Mathematics of Neural Networks*. Springer, 1997, pp. 306–311.

[18] D. P. Mandic and J. A. Chambers, "Towards the optimal learning rate for backpropagation," *Neural Process. Lett.*, vol. 11, no. 1, pp. 1–5, Feb. 2000.

[19] B. Widrow and M. A. Lehr, "The handbook of brain theory and neural networks," M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1998, ch. Perceptrons, Adalines, and Backpropagation, pp. 719–724.

[20] J. L. McClelland and D. E. Rumelhart, *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*. Cambridge, MA, USA: MIT Press, 1988.

[21] D. R. Wilson and T. R. Martinez, "The general inefficiency of batch training for gradient descent learning," *Neural Networks*, vol. 16, no. 10, pp. 1429–1451, 2003.

[22] D. McAllester, "Neural networks: Backpropagation general gradient descent," http://ttic.uchicago.edu/~dmcallester/ttic101-07/lectures/neural/neural.pdf, university of Chicago.

[23] D. G. Luenberger and Y. Ye, *Linear and nonlinear programming*. Springer, 1984, vol. 2.

[24] S. Roy, "Factors influencing the choice of a learning rate for a backpropagation neural network," in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, vol. 1, Jun 1994, pp. 503–507 vol.1.

[25] N. Kandil, K. Khorasani, R. Patel, and V. Sood, "Optimum learning rate for backpropagation neural networks," in *Electrical and Computer Engineering, 1993. Canadian Conference on*, Sep 1993, pp. 465–468 vol.1.

[26] X.-H. Yu and G.-A. Chen, "Efficient backpropagation learning using optimal learning rate and momentum," *Neural Netw.*, vol. 10, no. 3, pp. 517–527, Apr. 1997.

[27] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, 2012.

[28] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] V. Plagianakos, D. Sotiropoulos, and M. Vrahatis, "Automatic adaptation of learning rate for backpropagation neural networks," *Recent Advances in Circuits and Systems*, no. 337, 2014.

[30] H. B. Kim, S. H. Jung, T. G. Kim, and K. H. Park, "Fast learning method for back-propagation neural network by evolutionary adaptation of learning rates," *Neurocomputing*, vol. 11, no. 1, pp. 101 – 106, may 1996.

[31] Z. Jie, H. Long, and R. Sijing, "Rbf neural network adaptive sliding mode control based on genetic algorithm optimization," in *2016 Chinese Control and Decision Conference (CCDC)*, May 2016, pp. 6772–6775.

[32] B. u. Islam, Z. Baharudin, M. Q. Raza, and P. Nallagownden, "Optimization of neural network architecture using genetic algorithm for load forecasting," in *Intelligent and Advanced Systems (ICIAS), 2014 5th International Conference on*, June 2014, pp. 1–6.

[33] K. S. Sajan, B. Tyagi, and V. Kumar, "Genetic algorithm based artificial neural network model for voltage stability monitoring," in *Power Systems Conference (NPSC), 2014 Eighteenth National*, Dec 2014, pp. 1–5.

[34] A. Stein, "Properties of limits," http://www.math.uconn.edu/~stein/math115/slides/math115-130notes.pdf.

[35] G. R. Exner, *Inside Calculus*, ser. Undergraduate Texts in Mathematics, S. Axler, F. W. Gehring, and K. A. Ribet, Eds. Springer-Verlag: Springer-Verlag, 2000.

[36] A. A. Minai and R. D. Williams, "On the derivatives of the sigmoid," *Neural Networks*, vol. 6, no. 6, pp. 845 – 853, 1993.

[37] R. Salomon, "The curse of high-dimensional search spaces: observing premature convergence in unimodal functions," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1, June 2004, pp. 918–923 Vol.1.

[38] B. Doerr, T. Jansen, D. Sudholt, C. Winzen, and C. Zarges, "Optimizingmonotonefunctionscanbedifficult," in *Parallel Problem Solving from Nature, PPSN XI*, ser. Lecture Notes in Computer Science, R. Schaefer, C. Cotta, J. Koodziej, and G. Rudolph, Eds. Springer Berlin Heidelberg, 2010, vol. 6238, pp. 42–51.

[39] W. Press, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[40] O. Kksoy and T. Yalcinoz, "Robust design using pareto type optimization: A genetic algorithm with arithmetic crossover," *Computers & Industrial Engineering*, vol. 55, no. 1, pp. 208 – 218, aug 2008.

[41] A. Adewuya, *New Methods in Genetic Search with Real-valued Chromosomes*. Massachusetts Institute of Technology, Department of Mechanical Engineering, 1996.

[42] Kaggle, "Amazon.com - employee access challenge," https://www.kaggle.com/c/amazon-employee-access-challenge, May 2013, accessed: 20.05.2016.

[43] P. Duan and B. Solecki, "solution for the amazon employee access challenge," https://github.com/pyduan/amazonaccess, accessed: 10.12.2015.

[44] A. Bifet, "Data stream mining - regression," May 2012, accessed: 14.02.2017.

[45] "Waikato environment for knowledge analysis (weka)," accessed: 20.05.2016.

[46] "Rapidminer," accessed: 20.05.2016.