RESEARCH ARTICLE

DyHAP: Dynamic Hybrid ANFIS-PSO Approach for Predicting Mobile Malware

Firdaus Afifi[1], Nor Badrul Anuar[1]*, Shahaboddin Shamshirband[1], Kim-Kwang Raymond Choo[2,3,4]*

1 Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia, **2** Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, Texas. United States of America, **3** School of Information Technology & Mathematical Sciences, University of South Australia, Adelaide, South Australia, Australia, **4** School of Computer Science, China University of Geosciences, Wuhan, China

* badrul@um.edu.my (NBA); raymond.choo@fulbrightmail.org (KKRC)

Abstract

To deal with the large number of malicious mobile applications (e.g. mobile malware), a number of malware detection systems have been proposed in the literature. In this paper, we propose a hybrid method to find the optimum parameters that can be used to facilitate mobile malware identification. We also present a multi agent system architecture comprising three system agents (i.e. sniffer, extraction and selection agent) to capture and manage the pcap file for data preparation phase. In our hybrid approach, we combine an adaptive neuro fuzzy inference system (ANFIS) and particle swarm optimization (PSO). Evaluations using data captured on a real-world Android device and the MalGenome dataset demonstrate the effectiveness of our approach, in comparison to two hybrid optimization methods which are differential evolution (ANFIS-DE) and ant colony optimization (ANFIS-ACO).

Citation: Afifi F, Anuar NB, Shamshirband S, Choo K-KR (2016) DyHAP: Dynamic Hybrid ANFIS-PSO Approach for Predicting Mobile Malware. PLoS ONE 11(9): e0162627. doi:10.1371/journal.pone.0162627

Editor: Wen-Bo Du, Beihang University, CHINA

Received: June 19, 2016

Accepted: August 25, 2016

Published: September 9, 2016

Data Availability Statement: All relevant data are within the paper.

Funding: This work was supported by the Ministry of Science, Technology and Innovation, under Grant eScienceFund 01-01-03-SF0914.

Competing Interests: The authors have declared that no competing interests exist.

Introduction

The ubiquity and popularity of mobile devices is likely to increase in the foreseeable future. For example, according to the Global Web Index, 80% of Internet users own at least a smartphone and the online mobile shopping showed 150% increase in 2015 compared to 2014 [1]. Due to the widespread use of mobile devices and the amount of personal information stored on these devices, they have become the targets of cybercriminals such as malware authors and hackers [2–5]. Android devices are one of the most targeted platforms due to its market share, and open nature of the operating system [6–9]. One popular mitigation strategy used by mobile device users is anti-malware app. However, a recent systematic evaluation of popular free Android cloud-based anti-malware apps concluded:

> *that no single cloud anti-malware app can be solely relied upon to mitigate known malware. The findings were also concerning, particularly that malware threats are becoming more sophisticated and targeted, using various attack vectors to escalate permissions and exfiltrate data* [10]

Not surprisingly, mobile security and malware detection has been the subject of recent research. In order to detect malware, one could deploy an intrusion detection system (IDS) which can be either anomaly-based or signature-based (also called behavioral-based). The signature-based approach relies on a predefined pattern or malware signature. While such approach is popular, they are ineffective in detecting unknown malware [11,12]. Unlike signature-based detection, the anomaly-based approach seeks to differentiate between normal and abnormal conditions. For example, abnormal conditions include specific malware characteristics (e.g. malware code and its logical structure) or behaviors, identified during the analysis of such applications (i.e. malware analysis).

Malware analysis, a process of understanding how a particular piece of malware functions by dissecting and studying the code and its behavior with the aims of mitigating the threat [13], can be broadly categorized into static or dynamic analysis. Techniques such as machine learning have been utilized to differentiate normal and abnormal patterns in suspicious applications. For example, Dimitrios *et al.* [14] evaluated the suitability of five machine learning classifiers, namely: Radial Basis Function (RBF), Bayesian Networks, K-Nearest Neighbors (KNN) and Random Forest in detecting anomalies on mobile devices. Similarly, Feizollah *et al.* [15] analyzed the performance of machine learning classifiers in detecting Android malware and findings as high as 99.94% detection rate for KNN. Despite the amount of efforts on the topic, mobile malware detection remains a topic of active research (and the focus of this paper).

In recent times, a number of studies have seek to evaluate the effectiveness of computational intelligence-based solution for improved performance [16]. For example, FUGE [17] uses fuzzy theory and genetic method in cloud job scheduling algorithm. Findings from the authors' evaluations suggested the approach is efficient in terms of execution time, execution cost, and average degree of imbalance. Similarly, FR-TRUST [18] uses fuzzy theory to compute a peer trust level, and has been demonstrated to provide a high—ranking accuracy. In the study of malware, however, there are relatively few research works that use the fuzzy inference system because this is a complex NP problem. It is unlikely that efficient algorithms for solving this problem are deterministic; hence, the interest in heuristic algorithms.

We introduce an integrated method to detect mobile malware in this paper. Specifically, we use neural network function and regression to generalize the relationships between inputs based on the Adaptive Neuro-Fuzzy Inference System (ANFIS). The particle swarm optimization was also combined in our approach in order to optimize the malware prediction model. We then seek to evaluate the effectiveness of our approach and compare its performance with other hi-tech soft computing hybrid approaches.

The rest of this paper is structured in sections as follows: review of the related work (see Section 2), research methodology (see Section 3), proposed ANFIS framework (see Section 4), and evaluation of findings (see Section 5). Finally, the last section concludes this study.

Related Work

Malware detection approaches are categorized into anomaly-based and signature-based detection [19]. The signature-based method finds malware by comparing collected information from monitored users and system activities to an existing list of known malicious files database (i.e. malware signatures) [20]. While this approach has worked in the past, it is largely ineffective against new malware whose signature does not yet exist, or malware that uses "oligomorphic", "polymorphic" and "metamorphic" to avoid detection by encrypting or modifying parts of the code [21]. Such an approach also requires user to constantly update the signature database. Anomaly-based approach monitors and analyzes network traffic, system, user activity levels, etc. for a particular pattern of behavior. An intrusion is flagged when there is a deviation from the normal behavior patterns [22]. Machine learning classifiers, such as support vector machine, neural network, genetic algorithm, fuzzy logic, and decision tree, have been used in malware detection models [23]. To optimize malware detection model, selection of particular features is important in the machine learning classification process.

Malware analysis can be static or dynamic [24]. In static analysis, a program is examined by inspection without execution the actual application. Such process is normally performed manually by malware analysts to understand the logical structure, flow and data content stored within the binary itself, behavior of the suspicious application, etc. [25,26], [27]. For instance using Android Application Package (APK) file, [28] and [29] used file permissions as key reference points to detect malware on Android devices. However, Android malware such as DroidKungFuUpdate can avoid from being detected by not requesting access to suspicious permissions [30]. Do, Martini and Choo [31] in a recent work demonstrated how data can be exfiltrated from an Android device using inaudible sound waves via the device's speaker, which requires no permission. With the constant evolution of (mobile) malware and significant increase in the number of applications, it would be impossible to manually analyze all suspicious applications. In dynamic analysis, application activities such as network traffic and system calls are analyzed while the application is running. For example, Crowdroid [32] collected the device's kernel system calls to determine the application patterns. However, collecting system calls is a complicated task which requires device to be rooted. This can result in devices being more vulnerable to malicious exploitation. Yerima *et. al* [33] analyzed requested permissions of 2000 applications and determined that more than 93% of malware applications request for network connectivity (e.g. to communicate with the command and control server and to exfiltrate data). This indicated that, malicious applications tend to use network more than normal applications. Hence, we focus on analyzing mobile network traffic.

A variable selection process through the ANFIS was used to find the most significant parameters in malware detection. The aim was to find a subset of the logged variables that shows good prognostic abilities [34–36], and one can filter irrelevant variables by use of former knowledge. Donald A. [37] proposed genetic algorithm (GA) based variable selection for optimization, which aim to decrease the error between true values and prediction model by choosing the suitable explanatory variables (input). the ANFIS [38,39] was employed as a powerful tool for the variable selection in this paper. ANFIS has also been used in several engineering fields for modelling [40–43], predictions [44–46] and control [47–50]. The main idea of neuro-adaptive learning methods is to perform the fuzzy modelling procedure for data learning [51,52]. The ANFIS forms the fuzzy inference system with pairs (input/output) of data [53]. This approach enables fuzzy logic to adapt the membership function parameters to best track the given input/output data by the fuzzy inference system.

Metaheuristic optimization algorithms have become popular choice for solving complex problem [54]. As pointed out by one of the reviewers that combining ANFIS and Particle Swarm Optimization (PSO) for prediction problems has been widely studied and understood [55–57]. Pooranian and Shojafar [58], for example, proposed combining PSO with the gravitational emulation local search (GELS) to solve the independent task scheduling problem in grid computing. Jiang also proposed a PSO based ANFIS approach to improve accuracy in modelling customer satisfaction, and demonstrated that such an approach achieves better performance than fuzzy regression (FR), ANFIS and Genetic Algorithm (GA) based ANFIS approaches [59]. Other combinations of PSO and ANFIS have been proposed in forecasting such as in short-term wind power [60], and spur dike's parameters [61]. In order to increase its accuracy and performance, this paper applied three optimization techniques to ANFIS which are ANFIS-PSO (ANFIS-particle swarm optimization), ANFIS-DE (ANFIS-differential evolutionary) and ANFIS-ACO (ANFIS-ant colony optimization). These hybrid algorithms help

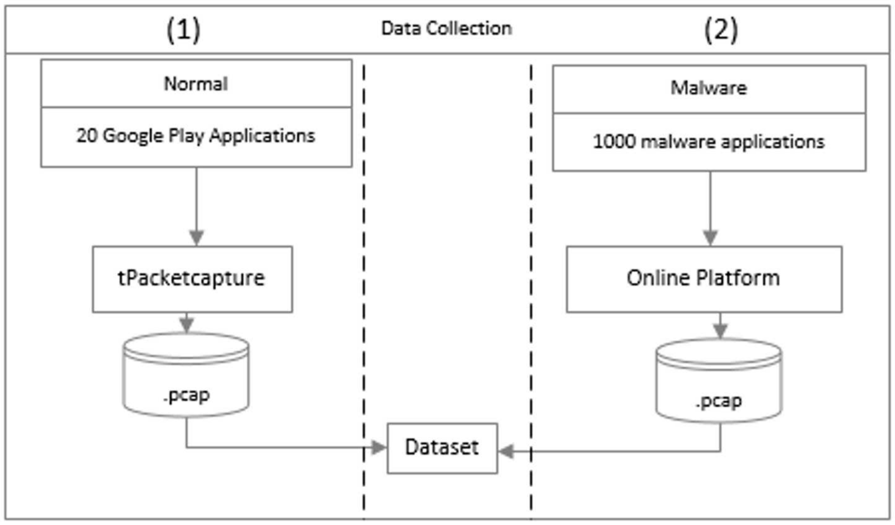


Fig 1. Data collection phase.

doi:10.1371/journal.pone.0162627.g001

improve the ANFIS performance by tuning the membership function towards zero error analysis.

Research Methodology

This section describes our experiment setup, which consists of two phases, namely: data collection, and feature selection, extraction and labelling phase.

Data Collection

We gathered and analyzed network traffic developed by Android apps. In this phase, different approaches were used to capture malware and normal network traffic (Fig 1).

Twenty popular (and reputable) apps from four different app categories were downloaded from Google Play and installed on a mobile device running Android operating system Jelly Bean version 4.1.2 (see Table 1). Prior to installation, we checked the authenticity of the apps. The network traffic from running these apps was captured in a real-time network environment, where each app was run for 30 minutes.

Of the 1260 malware data samples from 49 families in the Malgenome [30] dataset, we captured the network patterns of 1,000 samples. The samples were analyzed in real-time with public malware-detection sandbox, namely: Anubis Iseclab [62] and automatic Android program analysis, SanDroid [63], since the malware data samples in the dataset were generated by these platforms.

Table 1. Normal application categorization.

App	Total	Description
Social	3	Enables user to interact with other users or find people who share common interest such as hobbies, religion, politics, and alternative lifestyles
Communication	6	Enables user to make (free) phone call, video call, send multimedia message, attach file using network connection
Game	10	Enables user to play for enjoyment with certain situation either for educational or amusement purpose. It can be grouped with network connection or connected with social website.
Tool	1	Enables user to customize phone feature

doi:10.1371/journal.pone.0162627.t001

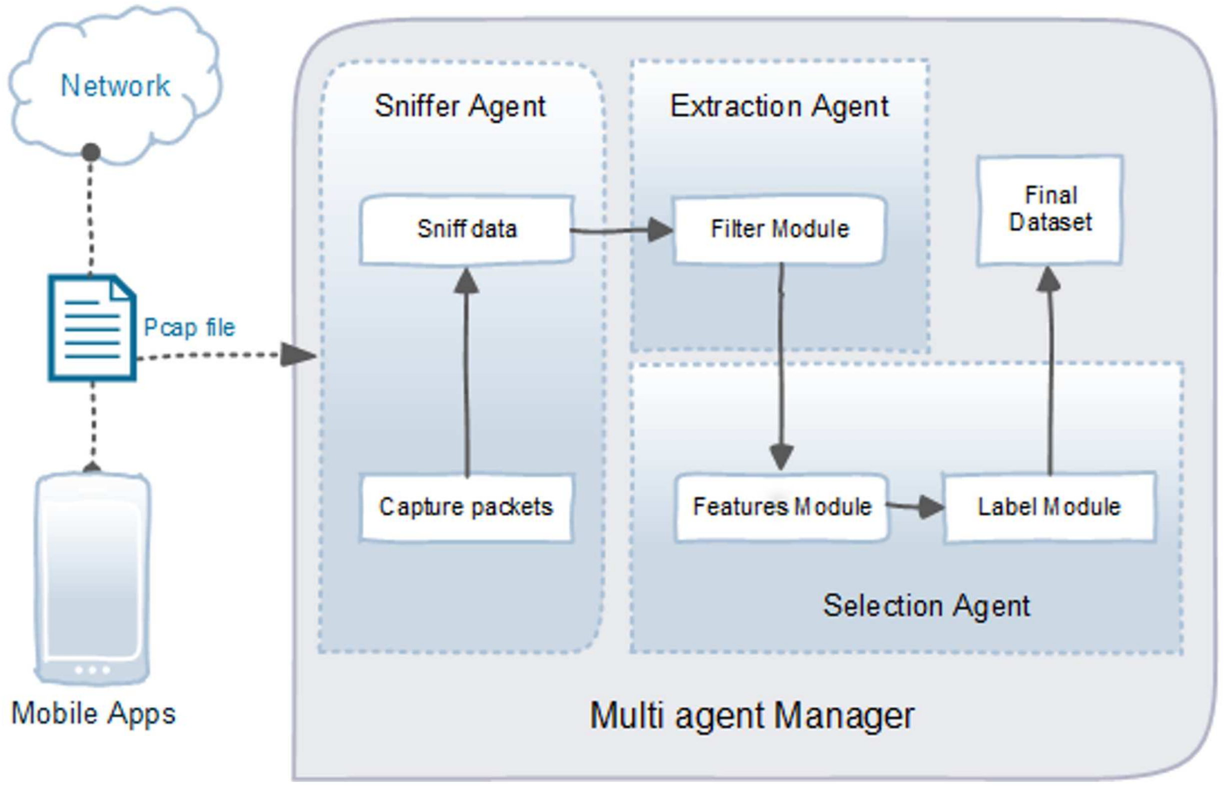


Fig 2. Multi agent.

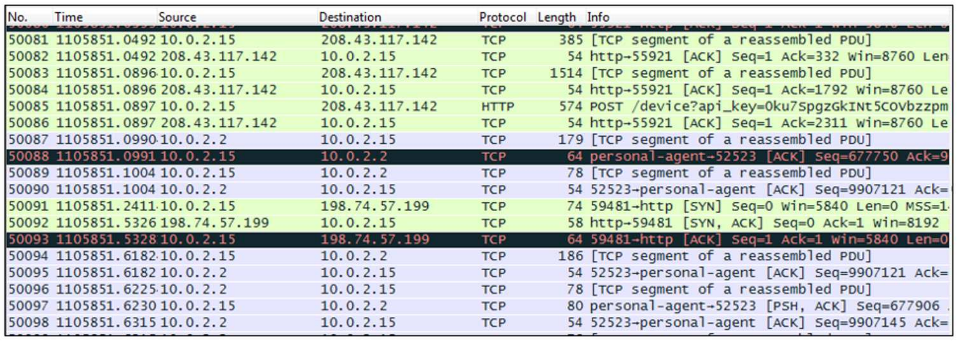
doi:10.1371/journal.pone.0162627.g002

Feature Selection, Extraction and Labelling

In this multi agent system architecture (Fig 2), three system agent is proposed to capture and manage the pcap file for data preparation phase. The details of each agent and its role are given in the next section.

Sniffer Agent. In the sniffer agent, the pcap file is capture from the network connection between mobile apps and internet. Sniffer module using tShark (a network protocol analyzer) [64], to retrieved all related information (see Fig 3). Later the sniffed data is fed to extraction agent.

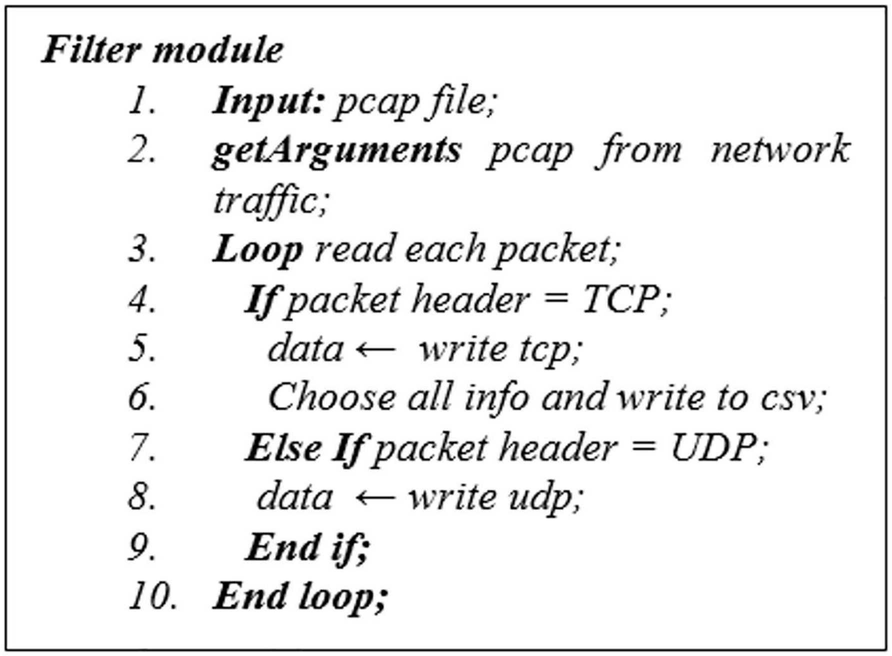
Extraction Agent. Extraction agent consist of filter module which filter the collected network traffic using Java and Wireshark routine to clear the captured packets from unwanted



```
No.   Time         Source          Destination     Protocol Length Info
50081 1105851.0492 10.0.2.15       208.43.117.142  TCP        385 [TCP segment of a reassembled PDU]
50082 1105851.0492 208.43.117.142  10.0.2.15       TCP         54 http→55921 [ACK] Seq=1 Ack=332 Win=8760 Len
50083 1105851.0896 10.0.2.15       208.43.117.142  TCP       1514 [TCP segment of a reassembled PDU]
50084 1105851.0896 208.43.117.142  10.0.2.15       TCP         54 http→55921 [ACK] Seq=1 Ack=1792 Win=8760 Le
50085 1105851.0897 10.0.2.15       208.43.117.142  HTTP       574 POST /device?api_key=0ku7SpgzGkINt5COVbzzpm
50086 1105851.0897 208.43.117.142  10.0.2.15       TCP         54 http→55921 [ACK] Seq=1 Ack=2311 Win=8760 Le
50087 1105851.0990 10.0.2.2        10.0.2.15       TCP        179 [TCP segment of a reassembled PDU]
50088 1105851.0991 10.0.2.15       10.0.2.2        TCP         64 personal-agent→52523 [ACK] Seq=677750 Ack=9
50089 1105851.1004 10.0.2.15       10.0.2.2        TCP         78 [TCP segment of a reassembled PDU]
50090 1105851.1004 10.0.2.2        10.0.2.15       TCP         54 52523→personal-agent [ACK] Seq=9907121 Ack=
50091 1105851.2411 10.0.2.15       198.74.57.199   TCP         74 59481→http [SYN] Seq=0 Win=5840 Len=0 MSS=1
50092 1105851.5326 198.74.57.199   10.0.2.15       TCP         58 http→59481 [SYN, ACK] Seq=0 Ack=1 Win=8192
50093 1105851.5328 10.0.2.15       198.74.57.199   TCP         64 59481→http [ACK] Seq=1 Ack=1 Win=5840 Len=0
50094 1105851.6182 10.0.2.15       10.0.2.2        TCP        186 [TCP segment of a reassembled PDU]
50095 1105851.6182 10.0.2.2        10.0.2.15       TCP         54 52523→personal-agent [ACK] Seq=9907121 Ack=
50096 1105851.6225 10.0.2.2        10.0.2.15       TCP         78 [TCP segment of a reassembled PDU]
50097 1105851.6230 10.0.2.15       10.0.2.2        TCP         80 personal-agent→52523 [PSH, ACK] Seq=677906
50098 1105851.6315 10.0.2.2        10.0.2.15       TCP         54 52523→personal-agent [ACK] Seq=9907145 Ack=
```

Fig 3. Sample of captured packets.

doi:10.1371/journal.pone.0162627.g003



```
Filter module
1.  Input: pcap file;
2.  getArguments pcap from network traffic;
3.  Loop read each packet;
4.    If packet header = TCP;
5.      data ← write tcp;
6.      Choose all info and write to csv;
7.    Else If packet header = UDP;
8.      data ← write udp;
9.    End if;
10. End loop;
```

Fig 4. Pseudocode of filter module.

doi:10.1371/journal.pone.0162627.g004

data. For example, we only use TCP packets in the network traffic data and remove UDP and *Domain name system* (DNS) packets from it. The pseudocode of this module is shown in Fig 4.

Selection Agent. This agent is one of the most important agents in this model. This agent has two modules which is Feature Module and Label Module. Feature Module choose a number of features to be used as main attributes to classify mobile malware. The features were chosen from a wide range of features in unbiased packet level features of the TUIDS intrusion dataset. The main challenge in this phase was to identify the best applicable features in particular, which result in higher detection accuracy and avoiding an overfitting model. The dataset needs to be filtered and refined from numerous excessive features. Some of the features are linked, which can complicate the process of malware detection. Furthermore, features with redundant information from other features may reduce the detection model accuracy and increase computational time and complexity of the model. In this study, a specific method to select the best attributes from machine learning tool Weka [65] called ClassifierSubsetEval was applied.

We choose seven connection-based features to analyze, as shown in Table 2. The extracted features were stored as a sequence of comma separated values (CSV) file. Next, after dataset was extracted with selected features, it was passed to Label module which labeled the dataset according to Fig 5. This phase remove noise in dataset and to ensure experiment validity. The final dataset from a combination of normal and infected data consists of three hundred thousand rows of data with seven features, prior to splitting into 70% training and 30% testing dataset. In order to avoid overfitting issue, we train our model with a wide range of examples and split datasets.

Proposed Approach

We introduce an approach in this paper that combines adaptive neuro fuzzy inference system (ANFIS) and particle swarm optimization (PSO). We used PSO to improve performance of

Table 2. Input and output parameters.

Inputs/Output	Parameters	Description
input 1	Maximum_Frame	The maximum number of frame in last P packets.
input 2	Frame_STD	Standard Deviation for frame in P packets
input 3	Count_ACK	The number of Acknowledge packet in the last P packets.
input 4	Minimum_Frame	The minimum number of frame in last P packets.
input 5	Average_Dest_Port	Average number of unique destination port in the last P packets.
input 6	Average_Frame	The average frame flowing in the last P packets.
input 7	Average_Source_Port	Average number of unique source port in the last P packets.
output 1	0,1	Uninfected = 0, Infected = 1

doi:10.1371/journal.pone.0162627.t002

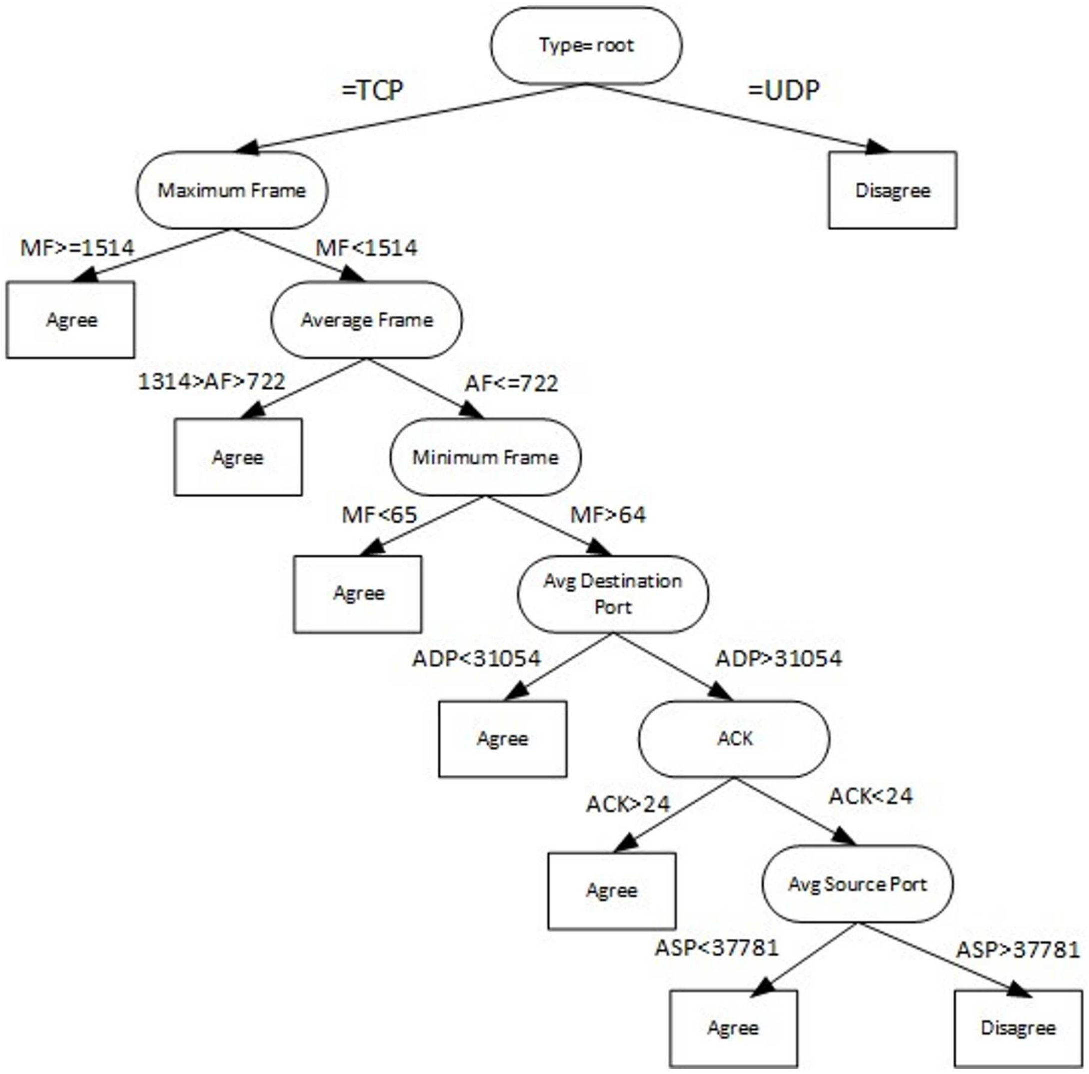


Fig 5. Decision tree for data labeling.

doi:10.1371/journal.pone.0162627.g005

ANFIS by adjusting the membership functions and minimizing the error. Forecasts from ANFIS can be used to reconstruct future behavior of the malware.

Particle swarm optimization (PSO)

PSO is an approach for optimizing "continue" and "discontinue" decision making functions, which develop by Dr. Kennedy and Dr. Eberhart in 1995 [55]. PSO has been used to model animals' sociological and biological behavior (like groups of birds searching for food) [66]. The PSO has also been employed in population-based search approach, in which a particle of a population is present for each individual potential solution or swarm. In this method, the position of particle is changed constantly in a search space until reaching to the optimum solutions and computational restrictions are reached.

Former experiential research shows the efficiency and advantages of the mentioned method for optimization [67], [68].

For example, in an optimization issue with D variables, a swarm of N particles is established in a way that every particle will be allotted to an arbitrary position in the hyperspace with D measurements. Position of each particle for this situation is associated with a possible answer for the optimization matter. Both v and x are flight speed of a particle over a solution space and its position (direction). A scoring capacity is allocated to every individual x in the swarm, which gains a wellness value. The latter is an indication of its competence to address the issues.

A particle's best prior position is represented by Pbest, and Gbest signifies the best swarm particle. Each particle can log its own Pbest and find its Gbest. Subsequently, all particles that move over the D-dimensional solution space should follow the rules updated for new positions until they achieve optimum position. The subsequent deterministic and stochastic update rules show how a particle's position and velocity are updated (Eq 1):

$$v_i(t) = \omega v_i(t-1) + \rho_1(x_{Pbest_i} - x_i(t)) + \rho_2(x_{Gbest} - x_i(t)) \tag{1}$$

$$x_i(t) = x_i(t-1) + v_i(t) \tag{2}$$

In the above equation, random variables are shown by q_1 and q_2 and x represents an inertia weight.

Positive acceleration constants are represented by C_1 and C_2 and the random variables are outlined as $q_1 = r_1c_1$ and $q_2 = r_2c_2$, with (r_1, r_2, $U(0,1)$). The stochastic and weights of growing speed terms that lead to a particle reaching to the Gbest and Pbest have speeding up constants of C_1 and C_2. A particle can move a long distance from the target locales when the qualities are few, while huge qualities cause the abrupt particles development to target locales. In line with the average practice in [69], both C_1 and C_2 constants are equal to 2.0 in this study. In Eq 2, the best likely amendment of dormancy x provides a harmony between the nearby and worldwide examinations, which reduces the amount of emphases on finding an ideal arrangement. A latency rectification capacity called the IWA or "idleness weight approach" was used in this exploration work [69,70]. The x (latency weight) is changed amid the IWA according to the associated relationship:

$$\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{\mathrm{Itr}_{\max}} Itr \tag{3}$$

In Eq 3, x_{max} and x_{min} represent the primary and ultimate inertia weights, the current number of iteration is represented by Itr and the maximum number of iteration is represented by Itr_{max}.

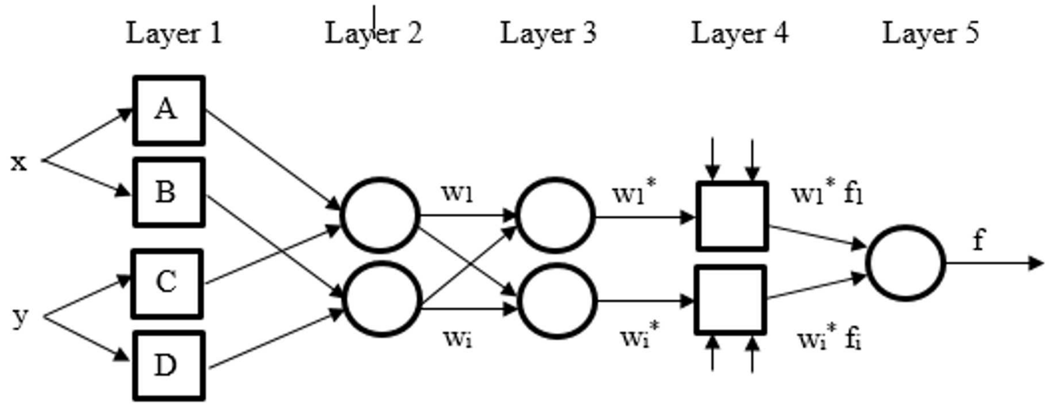


Fig 6. ANFIS structure.

doi:10.1371/journal.pone.0162627.g006

ANFIS

The term adaptive neuro-fuzzy inference system was introduced by Jang, 1993 refer to combination of Fuzzy Logic and Artificial Neural Network to produce a powerful processing tool [71]. For every input, two fuzzy if-then rule were generate in this study with maximum equal to 1 and minimum equal to 0. Fig 6 shows the ANFIS arrangement and inputs.

Assume two inputs fuzzy if-then rules of Takagi and Sugeno's type [72] were adopted:

$$if\ i\ is\ A\ and\ j\ is\ C\ and\ k\ is\ E\ and\ l\ is\ G\ then\ f_1 = p_1 i + q_1 j + r_1 k + s_1 l + t \tag{4}$$

Layer 1 contains membership functions (MFs) of input variables and feed input values for the next layer. Each node in 1st layer is adaptive as: $o = \mu(i)$, where $\mu(i)_i$ are membership functions.

The bell-shaped membership functions (Fig 7) is presented in Eq 5 for which the lowest and highest amounts are 0 and 1, respectively.

$$f(x; a, b, c) = \frac{1}{1 + \left(\frac{x-c}{a}\right)^{2b}} \tag{5}$$

The function is subject to the following parameters, namely a, b and c. Each of these parameters define as follows: a is half width of the curve; b defines the gradient together with a; and c is the midpoint of the membership function as shown in Fig 7.

In the 2nd layer (the membership layer), the weight of MFs is considered. The first layer provides the input values for layer 2. The nodes in the second layer are fixed node. The output is the product from all incoming signals and be described as,

$$w_i = \mu(i)_i \cdot \mu(i)_{i+1} \tag{6}$$

Output of every node indicates the weight strength of a rule.

In layer 3 which is the rule layer, every node does the pre-condition matching of the fuzzy rules, that calculate each rule's activation level as well as the normalized firing strength. This is a fixed layer as well, and each node computes the proportion of ith rule of the firing strength to

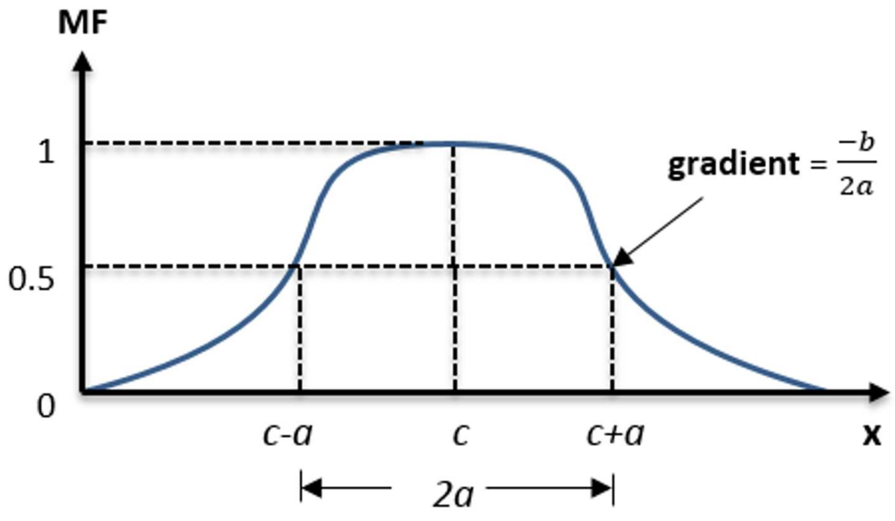


Fig 7. Three parameters in bell membership function; (a, b and c).

doi:10.1371/journal.pone.0162627.g007

the sum of *i*th firing strengths of all rules as:

$$w_i^* = \frac{w_i}{w_1 + w_2}, \ for \ \ i = 1, 2 \tag{7}$$

The outputs of this layer are named as normalized weights or firing strengths.

In layer 4 or defuzzification layer, all the adaptive nodes provide the resulting output values from the inference of rules.

$$O_i^4 = w_i^* \cdot f = w_i^* p_1 i + q_1 j + r_1 k + s_1 l + t \tag{8}$$

Here, the parameters set is shown as $\{p_i, q_i, r_i, s_i, t\}$.

Layer 5 or the output layer summarizes the inputs output from layer 4. This layer also transforms the results of fuzzy classification into a crisp. Here, the single node is fixed node and the whole incoming signals is sum up to produce overall output as below,

$$O_i^5 = \sum_i w_i^* \cdot f = \frac{\sum_i w_i \cdot f}{\sum_i w_i} \tag{9}$$

The PSO method was used in this paper to help ANFIS adjust the membership function parameters [70]. The main advantage of PSO technique is its friendly way of calculation in a network topology of given size. The membership functions were triangular in this study.

ANFIS-PSO algorithm

Fig 8 depicts the diagram of the sequential PSO and ANFIS combination [73]. In PSO, swarm starts with a group of random solutions, each of which is called a particle, and $\overrightarrow{s_i}$ represents the particle's position. Likewise, a particle swarm moves in the problem space, where $\overrightarrow{v_i}$ expresses the particle's velocity. A function *f* is evaluated at each time step through input $\overrightarrow{s_i}$. Every particle records its best position related to the best fitness gained to this point, in $\overrightarrow{p_i}$ vector. $\overrightarrow{p_i^g}$ tracks the most appropriate position identified by any neighborhood member. In universal

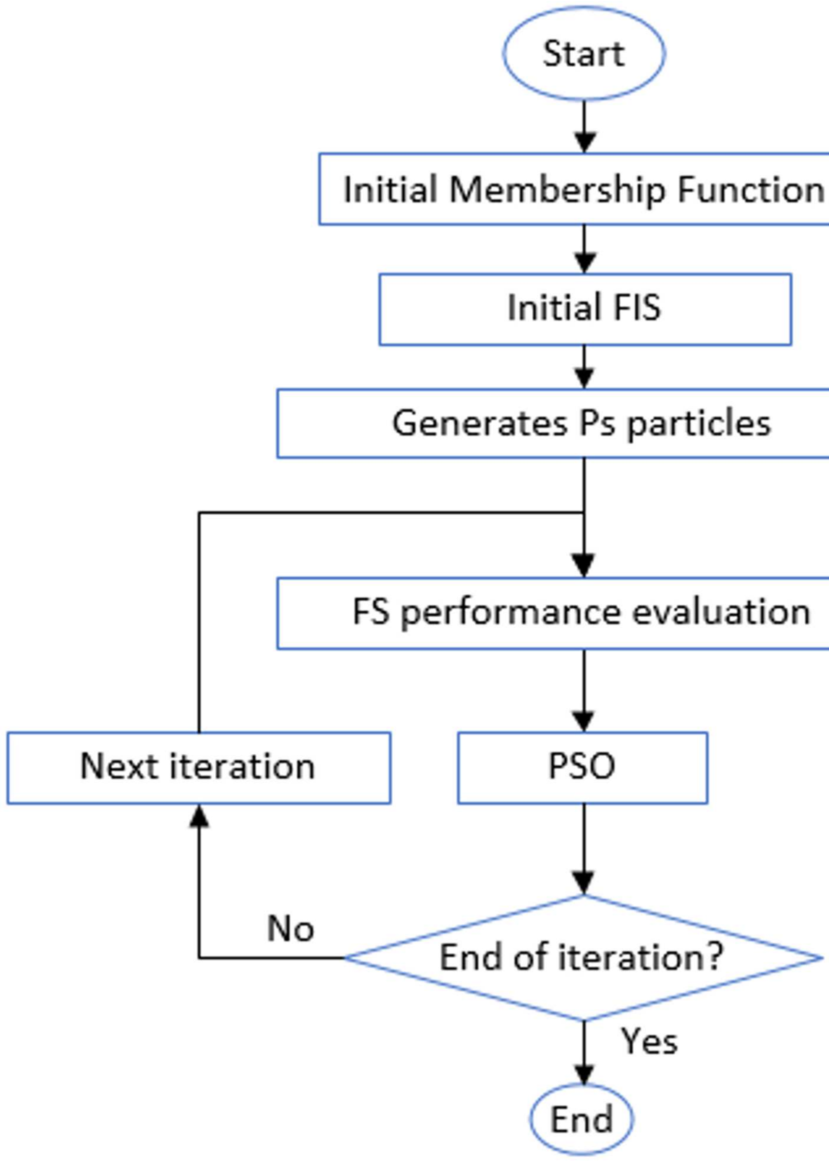


Fig 8. Diagram of sequential combination of PSO and ANFIS.

doi:10.1371/journal.pone.0162627.g008

version of PSO, $\overrightarrow{p_i^g}$ represents the most appropriate point in the entire population. A new velocity is achieved for any particle *i* in each iteration according to the best positions of individual, $\overrightarrow{p_i(t)}$, and $\vec{p}_i^g(t)$ neighborhood. The new velocity can be presented by:

$$\overrightarrow{v_i}(t+1) = w\overrightarrow{v_i}(t) + c_1\overrightarrow{\emptyset_1}.(\overrightarrow{p_i}(t) - \overrightarrow{x_i}(t)) + c_2\overrightarrow{\emptyset_2}.(\overrightarrow{p_i^g}(t) - \overrightarrow{x_i}(t)) \tag{10}$$

In Eq 10, *w* represents the inertia weight. The positive acceleration coefficients are shown by c_1 and c_2. $\overrightarrow{\emptyset_1}$ and $\overrightarrow{\emptyset_2}$ represent uniformly-distributed random vectors in [0,1], in which a random value is tried for every dimension. $\overrightarrow{v_i}$ limit in the [- $\overrightarrow{v_{max}}$, $\overrightarrow{v_{max}}$] series is reliant on the problem. Provided that the velocity exceeds the mentioned limit, in some cases it is rearranged within its suitable limits. The position of every particle alters depending upon the velocities as

Table 3. Parameter characteristics used in this study.

Population Size	Iterations	Inertia Weight	Damping Ratio	Learning coefficient	
				Personal	Global
40	1000	1	0.99	1	2

doi:10.1371/journal.pone.0162627.t003

follows

$$\overrightarrow{s_i}(t+1) = \overrightarrow{s_i}(t) + \overrightarrow{v_i}(t+1) \tag{11}$$

According to Eqs 10 and 11, the particles incline to gather nearby the best. PSO use for designing a FS, or parameter optimization is expressed as:

$$R_i : if\ x_1(k)\ is\ A_{i1}\ And \ldots And\ x_n(k)\ is\ A_{in}\ ,\ Then\ u(k) is\ a_i \tag{12}$$

Here, α_i is a crisp value, k represents the time step, the input variables are $x_1(k), \ldots, x_n(k)$, A_{ij} is a fuzzy set and $u(k)$ signifies the output variable for system.

For the FS in Eq 12 which comprises r rules and n input variables, its free parameters are defined through a position vector:

$$\overrightarrow{\boldsymbol{s}} = [\boldsymbol{m}_{11}, \boldsymbol{b}_{11}, \ldots, \boldsymbol{m}_{1n}, \boldsymbol{b}_{1n}, \boldsymbol{a}_1, \ldots\ldots, \boldsymbol{m}_{r1}, \boldsymbol{b}_{r1}, \ldots, \boldsymbol{m}_{rn}, \boldsymbol{b}_{rn}, \boldsymbol{a}_r] \in \Re^D \tag{13}$$

$$m_{rj} = x_j(k),\ b_{rj} = b_{fix},\ j = 1, \ldots, n \tag{14}$$

Following the process of rule creation and initialization, the preliminary antecedent part parameters are outlined. According to Eqs 13 and 14, the ith solution vector $\overrightarrow{s_i}$ is created as:

$$\overrightarrow{s_i} = [s_{i1}\ s_{i2} \ldots s_{iD}] = [m_{11} + \Delta m^i_{11}, b_{fix} + \Delta b^i_{11}, \ldots, m_{1n} + \Delta m^i_{1n}, b_{fix} + \Delta b^i_{1n}, a_1, \ldots, \\ m_{r1} + \Delta m^i_{r1}, b_{fix} + \Delta b^i_{r1}, \ldots, m_{rn} + \Delta m^i_{rn}, b_{fix} + \Delta b^i_{rn}, a_r] \tag{15}$$

In the equation, Δm_{ij} and Δb_{ij} signify the numbers of small random, α_i designates a random number distributed arbitrarily and homogeneously in the fuzzy system output range. The evaluation function f for $\overrightarrow{s_i}$ is calculated based upon the fuzzy system performance in Eq 15.

PSO looks for the best originator part parameters. P_s represents the population size. Eq 4 sets the elements in position $\overrightarrow{s_i}$. When $t = 0$, the $\overrightarrow{s_1}(0), \ldots, \overrightarrow{s_p}(0)$ or initial positions are created arbitrarily according to the best-performing FS found in ACO ($\vec{s}_{PSO}$). $\overrightarrow{s_1}(0)$ is considered similar to $\vec{s}_{PSO}$. The left $P_s - 1$ particles, $\overrightarrow{s_1}(0), \ldots, \overrightarrow{s_p}(0)$, are created by addition of uniformly-distributed random numbers to $\overrightarrow{s_{PSO}}$ shown as:

$$\overrightarrow{s_i}(0) = \overrightarrow{s_{PSO}} + \overrightarrow{w_i},\ i = 2, \ldots, P_s \tag{16}$$

$\overrightarrow{w_i}$ represents a random vector. The primary speed values of all particles, $\overrightarrow{v_i}(0),\ \ i = 1, \ldots, P_s$, are generated randomly. Each particle's performance is evaluated according to the FS it signifies. f is described as the $E(t)$ or error index mentioned above. The best position ($\overrightarrow{p_i}$) of each particle and the best particle $\overrightarrow{p^i_g}$ in the whole population is obtained according to f. Eqs 10 and 11 overhaul the velocity and position of each particle. The whole learning procedure is accomplished as soon as a pre-defined paradigm is obtained [73].

There are five PSO main parameters used during conducting experiment as shown in Table 3, which are maximum number of iterations, population size of the domain, inertia weight damping ratio and inertia weight, global learning coefficient and personal learning coefficient. For this case study, we determined these parameters optimum values by trial and error procedure.

Evaluation of model performances

Statistical tests offer a certain level of assurance about the validity, non-randomness of the [74]. Specifically, in this paper, we used root mean square error (RMSE), Eq 17 and coefficient of determination (R^2), Eq 18 to compare forecasting errors of between different models and determine the proportion of the variance of one variable that is predictable from the other variable, respectively.

The following are the statistical indicators adopted to examine the ANFIS model performance:

1. root-mean-square error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (O_i - P_i)^2}{n}} \tag{17}$$

2. Coefficient of determination (R^2)

$$\boldsymbol{R}^2 = \frac{\left[\sum_{i=1}^{n} (\boldsymbol{O}_i - \bar{\boldsymbol{O}}_i).(\boldsymbol{P}_i - \bar{\boldsymbol{P}}_i)\right]^2}{\sum_{i=1}^{n} (\boldsymbol{O}_i - \bar{\boldsymbol{O}}_i).\sum_{i=1}^{n} (\boldsymbol{P}_i - \bar{\boldsymbol{P}}_i)} \tag{18}$$

n is the total number of test data, P_i = measurement values and O_i = ANFIS value.

Evaluations

Simulation findings

The preliminary data aids in creating the hybrid soft computing method, and three methods were principally used to predict the data. The scatterplot in Fig 9 shows the estimation of best mobile malware parameters. Next, the fit line with equation $y = \alpha 0 + \alpha l$ was generated.

Performance analysis

The available experimental data were used for assessing the performance of methods and identifying importance of the parameters. The R^2 and RMSE were used to make comparison between the real and predicted values for the soft computing method. Tables 4 and 5 present the summary of comparison between ANFIS-DE, ANFIS-PSO and ANFIS-ACO. The performance analysis prediction of mobile malware using ANFIS-PSO is presented in Fig 10.

The ANFIS-PSO decision surface for mobile malware detection is shown in Fig 11 for the two extracted parameters, Maximum_Frame and Frame_STD.

It can be noted from Fig 10, when the model output is smaller than 0.5, the decision should be uninfected and when the model output is larger than 0.5 the decision should be infected. Finally based on this observation, we created SIMULINK block diagram for ANFIS-PSO detection of android mobile malware Fig 12.

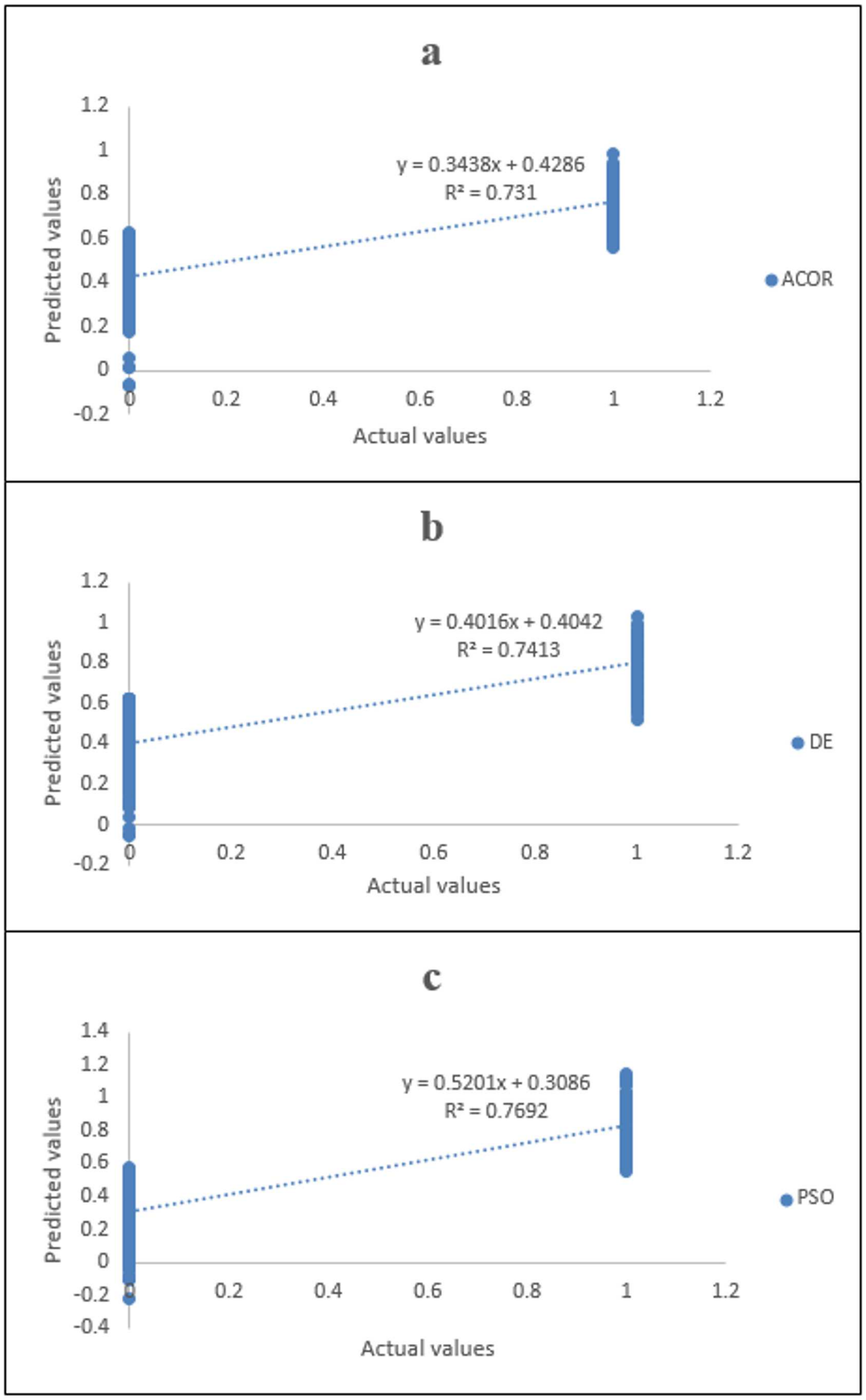


Fig 9. Performance of ANFIS-DE, ANFIS-ACO and ANFIS-PSO for estimation of mobile malware. (a) ANFIS-DE fit line. (b) ANFIS-ACO fit line. (c) ANFIS-PSO fit line.

doi:10.1371/journal.pone.0162627.g009

Table 4. Analysis of Mean Square Error (MSE) and Standard Deviation (StD) for different methods.

		ANFIS-ACO	ANFIS-DE	ANFIS-PSO
Training Data				
	MSE	0.20948	0.19487	0.18605
	StD	0.4577	0.44144	0.43133
Test Data				
	MSE	0.21098	0.19576	0.18581
	StD	0.45932	0.44245	0.43107

doi:10.1371/journal.pone.0162627.t004

Conclusion

The number and sophistication of Android malware are increasing and evolving, which necessitates the development of more effective malware detection systems. Recent advances in the literature suggests that artificial intelligence techniques are a promising approach to detect mobile malware. Generally, mobile malware communicates with a compromised server or a server under the control of an attacker, via a network. Thus, in this work, we focused on network based features. Specifically, application traffic was filtered for its parameters and calculation was performed on these parameters to obtain the required features. We also proposed three system agents to capture and manage the pcap file for the data preparation phase to improve our detection system in terms of efficient data processing.

It is known that ANFIS scheme is computationally efficient and well-adaptable with optimization and adaptive techniques. This scheme can also be combined with expert systems and rough sets for other applications, as well as used with other systems to handle more complex parameters. Another advantage of ANFIS is its speed of operation, which is much faster than in other control strategies. The laborious task of training membership functions is performed in ANFIS using metaheuristic optimization algorithms (due to the nature of fuzzy systems).

A novel hybrid method (integrating ANFIS and PSO) was proposed in this study to forecast the best parameters of a mobile malware analysis. The ANFIS-PSO is compared with two hybrid optimization approaches, namely: ANFIS-ACO and ANFIS-DE. Our findings demonstrated the utility of the proposed method. For example, ANFIS-PSO outperforms other approaches with RMSE, 0.43133 in training and 0.43106 in testing. Its coefficient of determination (R2) also achieves an improved performance (e.g. 0.7692 in training and 0.7721 in testing). For example, a majority (77%) of the variations in predicted result can be explained by the linear relationship between actual value and predicted model. This suggests that the prediction model has a strong positive linear correlation in terms of its accuracy in predicting and detecting Android malware.

Future work includes extending the research to a refined selection of variables (e.g. due to evolution of malware). Another potential research area is to address the known challenges in the selection of input variables, such as identifying and discarding irrelevant variables (noise).

Table 5. Analysis of performance for different methods to identify the optimum parameters of a mobile malware prediction model.

Method	Training		Testing	
	Error (RMSE)	Coefficient of determination (R^2)	Error (RMSE)	Coefficient of determination (R^2)
ANFIS-PSO	0.43133	0.7692	0.43106	0.7721
ANFIS-ACO	0.45769	0.7311	0.45932	0.7392
ANFIS-DE	0.44144	0.7413	0.44244	0.7562

doi:10.1371/journal.pone.0162627.t005

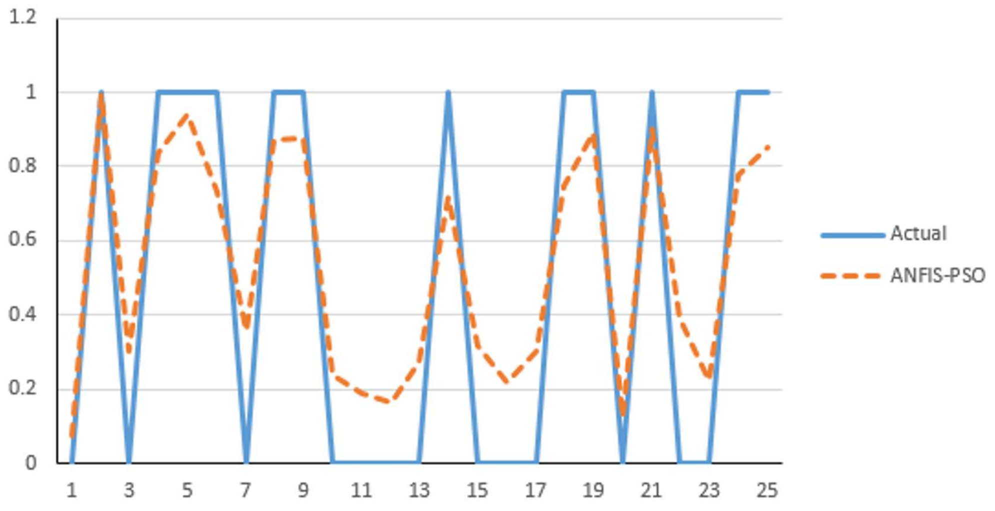


Fig 10. Prediction of the optimum parameters of mobile malware analysis by ANFIS-PSO for testing data.

doi:10.1371/journal.pone.0162627.g010

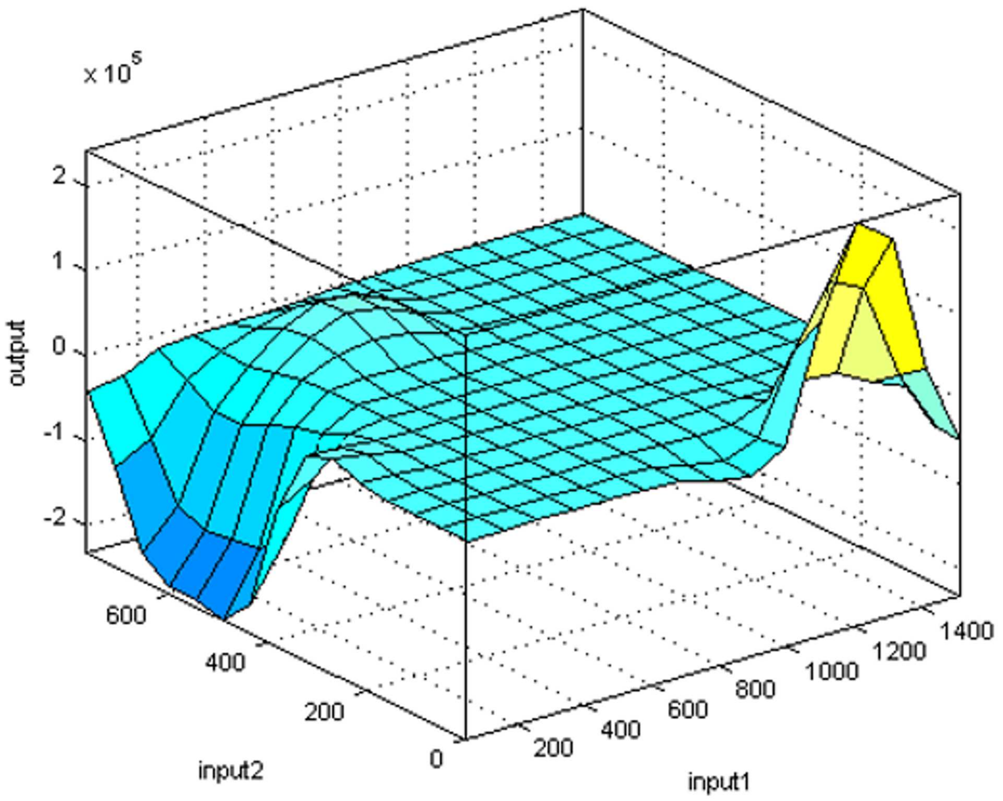


Fig 11. ANFIS-PSO decision surface for the detection model: input1—Maximum_Frame, input2—Frame_STD.

doi:10.1371/journal.pone.0162627.g011

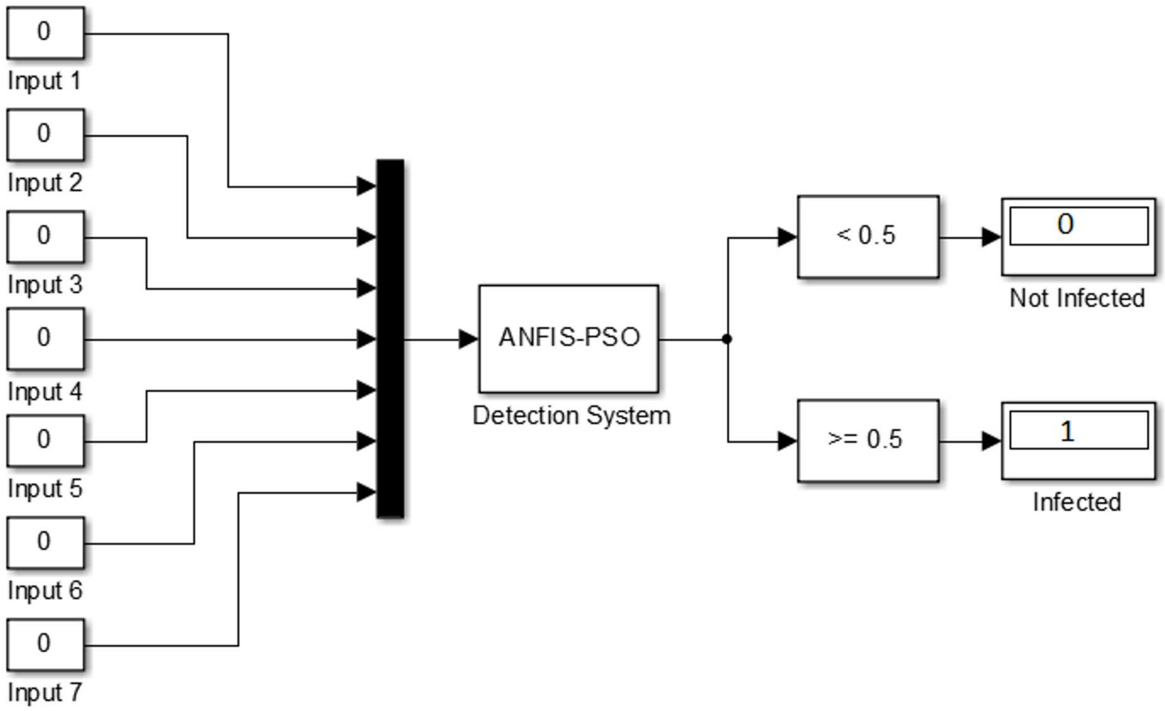


Fig 12. SIMULINK block diagram for ANFIS-PSO detection of android mobile malware.

doi:10.1371/journal.pone.0162627.g012

It is, therefore, useful to design methods that require reduced number of input variables (i.e. reducing the complexity of the model) yet achieving better efficiency and accuracy.

Acknowledgments

This work was supported by the Ministry of Science, Technology and Innovation, under Grant eScienceFund 01-01-03-SF0914.

Author Contributions

Conceptualization: FA NBA SS KKRC.

Data curation: FA NBA.

Formal analysis: FA SS.

Funding acquisition: NBA SS.

Investigation: FA NBA.

Methodology: FA NBA SS.

Project administration: NBA SS.

Resources: FA NBA SS.

Software: FA NBA SS.

Supervision: NBA SS.

Validation: FA NBA SS KKRC.

Visualization: FA SS.

Writing – original draft: FA NBA SS KKR.

Writing – review & editing: FA SS KKR.

References

1. Global Web Index. 80 of internet users own a smartphone [Internet]. 2015 [cited 5 Dec 2015]. Available: https://www.globalwebindex.net/blog/80-of-internet-users-own-a-smartphone
2. Choo KKR. The cyber threat landscape: Challenges and future research directions. Comput Secur. Elsevier Ltd; 2011; 30: 719–731. doi: 10.1016/j.cose.2011.08.004
3. Azfar A, Choo KKR, Liu L. Android mobile VoIP apps: a survey and examination of their security and privacy. Electron Commer Res. Springer US; 2015; 16: 1–39. doi: 10.1007/s10660-015-9208-1
4. Farnden J, Martini B, Choo K-KR. Privacy Risks in Mobile Dating Apps. 2015; 1–16. Available: http://arxiv.org/abs/1505.02906
5. Do Q, Martini B, Choo KKR. A forensically sound adversary model for mobile devices. PLoS One. 2015; 10: 1–15. doi: 10.1371/journal.pone.0138449
6. CNET. Android nabs 53% of US smartphone activations in Q1 [Internet]. 2014 [cited 1 Jun 2014]. Available: http://www.cnet.com/news/android-nabs-53-percent-of-us-smartphone-activations-in-q1
7. Theverge. Android is now used by 1.4 billion people [Internet]. 2015 [cited 30 Sep 2015]. Available: http://www.theverge.com/2015/9/29/9409071/google-android-stats-users-downloads-sales
8. Techcrunch. Android Accounted For 79% Of All Mobile Malware In 2012, 96% In Q4 Alone, Says F-Secure [Internet]. 2013 [cited 1 Jan 2013]. Available: http://techcrunch.com/2013/03/07/f-secure-android-accounted-for-79-of-all-mobile-malware-in-2012-96-in-q4-alone/
9. F-Secure. Q2 2014 Mobile Threat Report [Internet]. 2014 [cited 1 Jun 2014]. Available: https://www.f-secure.com/documents/996508/1030743/Threat_Report_H1_2014.pdf
10. Walls J, Choo K-KR. A Review of Free Cloud-Based Anti-Malware Apps for Android. 2015 IEEE Trust. 2015; 1053–1058. doi: 10.1109/Trustcom.2015.482
11. García-Teodoro P, Díaz-Verdejo J, Maciá-Fernández G, Vázquez E. Anomaly-based network intrusion detection: Techniques, systems and challenges. Comput Secur. 2009; 28: 18–28. doi: 10.1016/j.cose.2008.08.003
12. Damshenas M, Dehghantanha A, Choo K-KR, Mahmud R. M0Droid: An Android Behavioral-Based Malware Detection Model. J Inf Priv Secur. 2015;11. doi: 10.1080/15536548.2015.1073510
13. Distler D. Malware Analysis: An Introduction [Internet]. Information Security. 2001. Available: https://www.sans.org/reading-room/whitepapers/malicious/malware-analysis-introduction-2103
14. Dimitrios D, Sofia AM, Georgios K, Papadaki M, Clarke N, Stefanos G. Evaluation of Anomaly-Based IDS for Mobile Devices Using Machine Learning Classifier. Secur Commun Networks. 2011; 0: 1–9.
15. Feizollah A, Anuar NB, Salleh R, Narudin FA, Ma'arof RR, Shamshirband S. A Study Of Machine Learning Classifiers for Anomaly-Based Mobile Botnet Detection. Malaysian J Comput Sci. 2014; Volume 26. Available: http://umrefjournal.um.edu.my/public/article-view.php?id=5985
16. Zainab AN, Anuar NB. A single journal study: Malaysian Journal of Computer Sciences. Malaysian J Comput Sci. 2009; 22: 1–18.
17. Shojafar M, Javanmardi S, Abolfazli S, Cordeschi N. FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Cluster Comput. Springer US; 2015; 18: 829–844. doi: 10.1007/s10586-014-0420-x
18. Javanmardi S, Shojafar M, Shariatmadari S, Ahrabi SS. FR TRUST: A Fuzzy Reputation Based Model for Trust Management in Semantic P2P Grids. Int J Grid Util Comput. 2014; 1–11. doi: 10.1504/IJGUC.2015.066397
19. Inayat Z, Gani A, Anuar NB, Khan MK, Anwar S. Intrusion response systems: Foundations, design, and challenges. J Netw Comput Appl. Elsevier; 2015; doi: 10.1016/j.jnca.2015.12.006
20. Razak MFA, Anuar NB, Salleh R, Firdaus A. The rise of "malware": Bibliometric analysis of malware study. J Netw Comput Appl. Elsevier; 2016; doi: 10.1016/j.jnca.2016.08.022
21. Szor P. The Art of Computer Virus Research and Defense. Addison-Wesley Professional. 2005.
22. Scarfone K, Mell P. Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Spec Publ. 2012; 1: 111.

23. Sangkatsanee P, Wattanapongsakorn N, Charnsripinyo C. Practical real-time intrusion detection using machine learning approaches. Comput Commun. Elsevier B.V.; 2011; 34: 2227–2235. doi: 10.1016/j.comcom.2011.07.001

24. Egele M, Scholte T, Kirda E, Barbara S. A survey on automated dynamic malware analysis techniques and tools. ACM Comput Surv. 2011; V: 1–49. doi: 10.1145/2089125.2089126

25. D'Orazio C, Choo KKR. An adversary model to evaluate DRM protection of video contents on iOS devices. Comput Secur. Elsevier Ltd; 2016; 56: 94–110. doi: 10.1016/j.cose.2015.06.009

26. Dorazio C, Choo KKR. A generic process to identify vulnerabilities and design weaknesses in iOS healthcare apps. Proc Annu Hawaii Int Conf Syst Sci. 2015;2015-March: 5175–5184. doi: 10.1109/HICSS.2015.611

27. Sharif M, Yegneswaran V, Saidi H, Porras P, Lee W. Eureka: A framework for enabling static malware analysis. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics). 2008; 5283 LNCS: 481–500.

28. Huang C, Tsai Y, Hsu C. Performance Evaluation on Permission-Based Detection for Android Malware. Proceedings of the International Computer Symposium ICS. 2012. pp. 111–120.

29. Van Der Merwe H. Analysis of Android applications. 2012; 1–7.

30. Zhou Y, Jiang X. Dissecting Android malware: Characterization and evolution. Proc—IEEE Symp Secur Priv. 2012; 95–109. doi: 10.1109/SP.2012.16

31. Do Q, Martini B, Choo KKR. Exfiltrating data from Android devices. Comput Secur. Elsevier Ltd; 2015; 48: 74–91. doi: 10.1016/j.cose.2014.10.016

32. Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid. Proc 1st ACM Work Secur Priv smartphones Mob devices—SPSM '11. 2011; 15.

33. Yerima SY, Sezer S, McWilliams G. Analysis of Bayesian Classification-based Approaches for Android Malware Detection. Inf Secur IET. 2014; 8: 25–36. doi: 10.1049/iet-ifs.2013.0095

34. Castellano G, Fanelli AM. Variable selection using neural-network models. Neurocomputing. 2000; 31: 1–13.

35. Dieterle F, Busche S, Gauglitz G. Growing neural networks for a multivariate calibration and variable selection of time-resolved measurements. Anal Chim Acta. 2003; 490: 71–83. doi: 10.1016/S0003-2670(03)00338-6

36. Andersson FO, Åberg M, Jacobsson SP. Algorithmic approaches for studies of variable influence, contribution and selection in neural networks. Chemom Intell Lab Syst. 2000; 51: 61–72. doi: 10.1016/S0169-7439(00)00057-5

37. Sofge DA. Using Genetic Algorithm Based Variable Selection to Improve Neural Network Models for Real-World Systems. Artif Intell. 2002;

38. Chan KY, Ling SH, Dillon TS, Nguyen HT. Diagnosis of hypoglycemic episodes using a neural network based rule discovery system. Expert Syst Appl. Elsevier Ltd; 2011; 38: 9799–9808. doi: 10.1016/j.eswa.2011.02.020

39. Kwong CK, Wong TC, Chan KY. A methodology of generating customer satisfaction models for new product development using a neuro-fuzzy approach. Expert Syst Appl. Elsevier Ltd; 2009; 36: 11262–11270. http://dx.doi.org/10.1016/j.eswa.2009.02.094

40. Samhouri M, Al-Ghandoor A, Fouad RH, Hakim AH, Vasant P, Barsoum N. Electricity Consumption in the Industrial Sector of Jordan: Application of Multivariate Linear Regression and Adaptive Neuro-Fuzzy Techniques. AIP Conf Proc. 2009; 135–143. doi: 10.1063/1.3223918

41. Singh R, Kainthola A, Singh TN. Estimation of elastic constant of rocks using an ANFIS approach. Appl Soft Comput J. Elsevier B.V.; 2012; 12: 40–45. doi: 10.1016/j.asoc.2011.09.010

42. Petković D, Issa M, Pavlović ND, Pavlović NT, Zentner L. Adaptive neuro-fuzzy estimation of conductive silicone rubber mechanical properties. Expert Syst Appl. 2012; 39: 9477–9482. doi: 10.1016/j.eswa.2012.02.111

43. Petković D, Ćojbašić Ž. Adaptive neuro-fuzzy estimation of autonomic nervous system parameters effect on heart rate variability. Neural Comput Appl. 2012; 21: 2065–2070. doi: 10.1007/s00521-011-0629-z

44. Hosoz M, Ertunc HM, Bulgurcu H. An adaptive neuro-fuzzy inference system model for predicting the performance of a refrigeration system with a cooling tower. Expert Syst Appl. 2011; doi: 10.1016/j.eswa.2011.04.225

45. Khajeh A, Modarress H, Rezaee B. Application of adaptive neuro-fuzzy inference system for solubility prediction of carbon dioxide in polymers. Expert Syst Appl. Elsevier Ltd; 2009; 36: 5728–5732. doi: 10.1016/j.eswa.2008.06.051

46. Sivakumar R, Balu K. ANFIS based Distillation Column Control. Int J Comput Appl. 2010;ecot: 67–73. doi: 10.5120/1538-141

47. Kurnaz S, Cetin O, Kaynak O. Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles. Expert Syst Appl. Elsevier Ltd; 2010; 37: 1229–1234. doi: 10.1016/j.eswa.2009.06.009

48. Ravi S, Sudha M, Balakrishnan PA. Design of intelligent self-tuning GA ANFIS temperature controller for plastic extrusion system. Model Simul Eng. 2011;2011. doi: 10.1155/2011/101437

49. Areed FG, Haikal AY, Mohammed RH. Adaptive neuro-fuzzy control of an induction motor. Ain Shams Eng J. Faculty of Engineering, Ain Shams University; 2010; 1: 71–78. doi: 10.1016/j.asej.2010.09.008

50. Petković D, Issa M, Pavlović ND, Zentner L, ojbašić Ž. Adaptive neuro fuzzy controller for adaptive compliant robotic gripper. Expert Systems with Applications. 2012. pp. 13295–13304. doi: 10.1016/j.eswa.2012.05.072

51. Aldair AA, Wang WJ. Controller design for an autonomous underwater vehicle using nonlinear observers. Int J smart Sens Intell Syst. 2011; 4: 224–243. doi: 10.5574/IJOSE.2011.1.1.016

52. Dastranj MR, Ebroahimi E, Changizi N, Sameni E. Control DC Motorspeed with Adaptive Neuro-Fuzzy control (ANFIS). Aust J Basic Appl Sci. 2011; 5: 1499–1504.

53. Manoj SBA. Identification and Control of Nonlinear Systems using Soft Computing Techniques. Int J Model Optim. 2011; 1: 24.

54. Thangaraj R, Pant M, Abraham A, Bouvry P. Particle swarm optimization: Hybridization perspectives and experimental illustrations. Applied Mathematics and Computation. 2011. pp. 5208–5226.

55. Eberhart R, Kennedy J. A new optimizer using particle swarm theory. MHS'95 Proc Sixth Int Symp Micro Mach Hum Sci. 1995; 39–43. doi: 10.1109/MHS.1995.494215

56. Mitchell M. An introduction to genetic algorithms. MIT Press; 1998.

57. Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim. 1997; 341–359. doi: 10.1023/A:1008202821328

58. Pooranian Z, Shojafar M, Abawajy JH, Abraham A. An efficient meta-heuristic algorithm for grid computing. J Comb Optim. 2015; 30: 413–434. doi: 10.1007/s10878-013-9644-6

59. Jiang HM, Kwong CK, Ip WH, Wong TC. Modeling customer satisfaction for new product development using a PSO-based ANFIS approach. Appl Soft Comput J. 2012; 12: 726–734. doi: 10.1016/j.asoc.2011.10.020

60. Catalao JPS, Pousinho HMI, Mendes VMF. Hybrid Wavelet-PSO-ANFIS Approach for Short-Term Electricity Prices Forecasting. IEEE Trans Power Syst. 2011; 26: 137–144. doi: 10.1109/TPWRS.2010.2049385

61. Basser H, Karami H, Shamshirband S, Akib S, Amirmojahedi M, Ahmad R, et al. Hybrid ANFIS-PSO approach for predicting optimum parameters of a protective spur dike. Appl Soft Comput J. 2015; 30: 642–649. doi: 10.1016/j.asoc.2015.02.011

62. Anubis. Anubis: Analyzing Unknown Binaries [Internet]. 2013 [cited 1 Dec 2014]. Available: http://anubis.iseclab.org/ 23888591

63. SandDroid. SandDroid: An Automatic Android Program Analysis Sandbox [Internet]. 2013 [cited 10 Dec 2014]. Available: http://sanddroid.xjtu.edu.cn/ 23888591

64. Tshark. tshark—The Wireshark Network Analyzer 1.12.0 [Internet]. 2013 [cited 21 Jan 2015]. Available: http://www.wireshark.org/docs/man-pages/tshark.html 23888591

65. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA Data Mining Software : An Update. SIGKDD Explor. 2009; 11: 10–18. doi: 10.1145/1656274.1656278

66. Bashir ZA, El-Hawary ME. Applying wavelets to short-term load forecasting using PSO-based neural networks. IEEE Trans Power Syst. 2009; 24: 20–27. doi: 10.1109/TPWRS.2008.2008606

67. Yu W, Li X. Fuzzy identification using fuzzy neural networks with stable learning algorithms. IEEE Trans Fuzzy Syst. 2004; 12: 411–420. doi: 10.1109/TFUZZ.2004.825067

68. Yuan X, Wang L, Yuan Y. Application of enhanced PSO approach to optimal scheduling of hydro system. Energy Convers Manag. 2008; 49: 2966–2972. doi: 10.1016/j.enconman.2008.06.017

69. Kennedy J. The behavior of particle. In: Porto VW, Saravanan N, Waagen D, Eiben AE, editors. Evolutionary Programming VII. Springer Berlin Heidelberg; 1998. pp. 579–589. doi: 10.1007/BFb0040809

70. Shoorehdeli MA, Teshnehlab M, Sedigh AK, Khanesar MA. Identification using ANFIS with intelligent hybrid stable learning algorithm approaches and stability analysis of training methods. Appl Soft Comput. 2009; 9: 833–850. doi: 10.1016/j.asoc.2008.11.001

71. Jang JSR. ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern. 1993; 23: 665–685. doi: 10.1109/21.256541

72. Takagi T, Sugeno M. Derivation of fuzzy control rules from human operator's control actions. Proceedings of the IFAC symposium on fuzzy information, knowledge representation and decision analysis. 1983. p. Vol. 6, pp. 55–60.

73. Juang C. Combination of Particle Swarm and Ant Colony Optimization Algorithms for Fuzzy Systems Design. 2010;

74. Demšar J. Statistical Comparisons of Classifiers over Multiple Data Sets. J Mach Learn Res. 2006; 7: 1–30.