UDC - 004.725:004.852

RESEARCH THE MULTIDIMENSIONAL LOGISTIC FUNCTION IN THE INTRUSION DETECTION SYSTEM WITH MACHINE LEARNING

Timur V. Jamgharyan
National Polytechnical University of Armenia
105 Teryan St., 0009, Yerevan,
t.jamgharyan@yandex.ru
ORCID iD: 0000-0002-9661-1468
Republic of Armenia

https://doi.org/10.56243/18294898-2023.2-64

Abstract

The paper presents the results of the research *the* model for changing the logistic function. The research was conducted on datasets generated based on the source code abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip malware source code base. *R*esearch was conducted to evaluate the accuracy of the developed intrusion detection system with machine learning. As a mathematical apparatus for the research, a multidimensional logistic function softmax was chosen. The simulation of the developed software at different iterations and visualization of the results was carried out.

Keywords: machine learning, dataset, malware, intrusion detection system, multidimensional logistic function, probability distribution vector, softmax.

Introduction

The building a network intrusion detection system (IDS), it is important to reduce the number of points of failure. One of the requirements is the formation of such a data set that the resulting output value of the probability of a certain event (P(A)=1, (P(A)-probability of the event) was obtained with the minimum quantitative value of the input data. The solution to this problem is achieved in various ways, each of which works within the specified constraints. In particular, for small, predetermined data sets, step functions and/or the sigmoid function are used. But the use of the sigmoid function is limited by the fact that when a neural network is activated based on it, the number of logits grows in proportion to the training set. The step function is commonly used in binary learning. The most optimal task of training a neural network is solved by using a multidimensional logistic function (softmax) in the process of training to activate the layer / layers of the neural network.

Equation (1) describes the operation of the logistic function [1].

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_i)} \quad (1)$$

where x_i -vector of values; n - the number of classes.

When developing systems with machine learning (ML), it is important to correctly set softmax values to reduce type 1 and type 2 errors (*true positive, false positive, true negative, false negative*). The task that needs to be solved every time when designing systems with ML is the task of setting such parameters of the softmax function, on the basis of which the function would activate «weights» with the minimum value of logits and their least number. Different researchers use different approaches to arrive at a general solution. In particular, in [2] a model for checking the class hierarchy based on classifications is presented, in [3] the solution of the problem is obtained with an unlimited parameterized value of *softmax*. The research [4] proposes a new method of adversarial learning based on the detection and removal of large values of «weight» coefficients, rather than their algorithmic reduction. The research was conducted on the SVHN and CIFAR-10 datasets. The increase in attacks on network IDS built using M2M&ML (Machine-to-Machine, M2M) technology forces researchers to look for new ways to increase the reliability of the results issued by softmax. But in the case of malware, this approach is inefficient due to the fact that it is a priori difficult to obtain the exact code base of malware accordingly, the softmax function «activated» on one value will be inactive on another, although the type of malware will be the same.

It becomes relevant for researchers of security systems using ML to set such a value of logits so that when they change, the neural network «does not forget» its previous state (the softmax=1 value would be obtained with a minimum set of logits and an «active» state of the neurons of the previous layers). To achieve the goal of preserving the previous state of the neural network, it is necessary to develop such a softmax type that, with the value of its output vector, the number of logits is constant. In this paper, research the use of datasets of malicious polymorphic software abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip detected by piecewise context hashing [5].

The novelty of the reseacrch lies in the alternate (left/right) introduction of the *hash function modifier* [1] into the «incomplete» (piecewise-context) hash of the malicious polymorphic software for *softmax* activation. In this case, the prototype for calculating the hash is the hash value obtained by the CTPH method (context triggered piecewise hashing) from malicious polymorphic software. The method of generating a new hash value based on the previous value is used to generate *rainbow tables*[2]. The use of a modifier is necessary to «controlled change» the value of the logit sets when softmax is activated.

Conflict Setting

It is necessary to get the value of the output vector softmax =1 with the minimum value of logits.

[1] *The hash function modifier - is a string of data that is passed to the hash function along with a preimage to calculate the hash*.

[2] *Rainbow table is a data array containing pre-calculated hash values for the research object* [6].

Materials and methods

Software implementing *softmax* is integrated into the convolutional neural network (CNN) that searches for malware (detailed in [5]) using the CTPH method). The change in softmax occurs according to the following scheme (Fig. 1).

Algorithm

The hash value obtained by the CTPH method from the malware datasets, together with the hash function modifier, is input to the inserter software. The weight setting is determined by the number of malware hash values detected by the CTPH method.

a) insert to the left of the *hash modifier* - when the value of the hash function obtained by the CTPH method is greater than the current value of the *hash modifier*,

b) insert to the right of the *hash modifier* - if the value of the hash function obtained by the CTPH method is less than the current value of the *hash modifier*.

When a number value with an indefinite result (NaN, Not-a-Number) appears in the handler, the execution of the entire program is «stopped», which resets all values.

The buffer always contains n-1 dataset values (the n-dataset currently being processed)

The *hash modifier* value is obtained by computing the hash function of the current system time in *POSIX time* format. This implementation allows to carry out granular (discrete) reverse engineering of both softmax and the entire neural network, achieving maximum activation of the «weights» of the neural network, with a minimum value of logits in *softmax* [7].

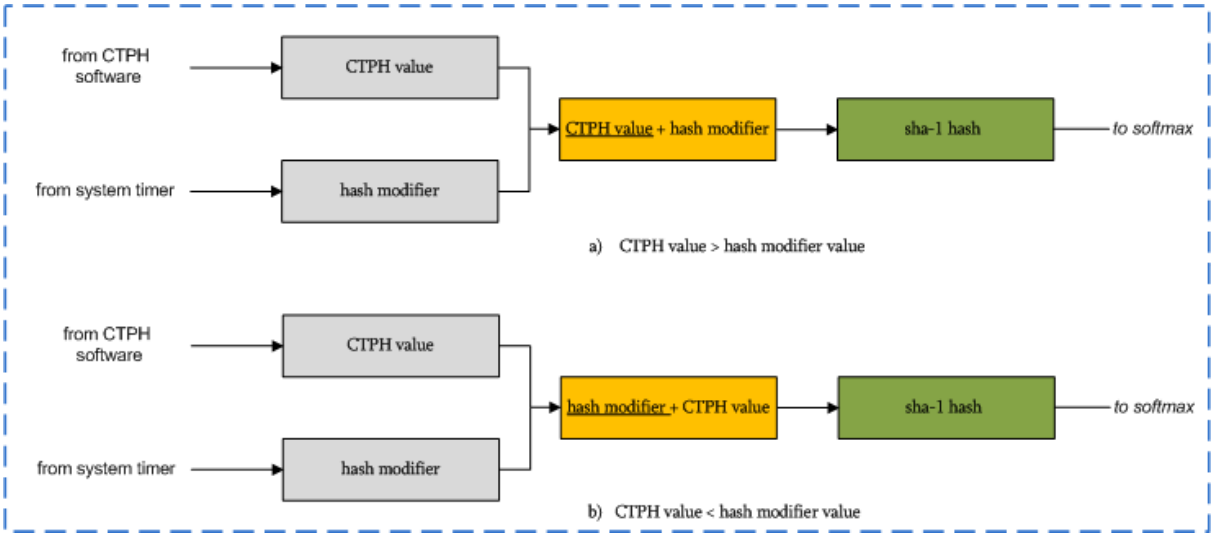

Fig. 1 Scheme for inserting hashes based on system time

Boundary conditions

- network error is determined by the deviation from the hash modifier value, by an amount equal to the size of the current CTPH (20,40,80 bytes),
- if the hash modifier value is equal to the value of the hash function obtained by the CTPH method, the calculation is not performed (Not-A-Number, NaN),
- the maximum number of classes $n \leq 2^{20}$ (determined by the «state matrix»),

- limit upper value of the vector of values =40960 (*determined by the capacity of the single element of the «state matrix»*[3]),
- lower value of the value vector=0, not considered.

Experimental procedures

Based on the Dell Power Edge T-330 server, the Hyper-V role is installed in the Windows Server 2016 Standard operating system environment. A software-defined network (Software Defined Network, SDN) is deployed in which Parrot OS is installed with the *metasploit* framework installed, a CHR (Cloud Hosted Router, CHR) virtual cloud router based on the Router OS operating system with an NTP (Network Time Protocol, NTP) server installed, Ubuntu v20.04 OS in which are installed: IDS Snort version 3.0, Clion development environment with developed software and convolutional neural network.

As the number of classes under research, the following were sequentially selected:

- 512 datasets, 80 bytes each, for the 1st training epoch,
- 1024 datasets, 40 bytes each, for training epoch II,
- 2048 datasets, 20 bytes each, for training epoch III.

The malware datasets detected by the CTPH method are combined with a hash value derived from the current system time. The local NTP server is used as the time source. The use of a local NTP server made it possible to reduce the time delay and, as a result, improve the accuracy of calculations. The generated new dataset is fed to the input of the CNN. The data obtained at each training epoch model many different probability distributions. As a measure of comparison of probability distributions, the cross entropy function (2) [8,9] was chosen.

$$C = -\sum_{j=1}^{n} t_i \log y_i \qquad (2)$$

where t - expected softmax value; y - the resulting softmax value.

The total network error must not exceed the boundary conditions. Below is a fragment of the developed source code for calculating softmax based on the CTPH value of the polymorphic software flip.

The software was developed using the TensorFlow and NumPy machine learning library. The cross entropy measurement was done using sklearn.metrics.log_loss software.

Research Results

Using a CNN to detect polymorphic malware is justified only if a relevant dataset is available. Training a neural network with a value of softmax =1, but a small number of logits (the «weight» of each is large, but there are few of them) leads to a distortion of the *softmax* results. In particular, in Tab. 1 and Tab. 2 can be seen that increasing training epochs and/or iterations in epochs leads to approximately equal results. It becomes possible to maneuver by changing the number of sets with the CTPH size unchanged, or by changing the CTPH size to leave the number of datasets under researches unchanged (In both cases, the total network

[3] *The «state matrix» is an NxN matrix with datasets of malicious polymorphic software*.

error is equal to and within the allowable values, which confirms the truth (applicability) of equation 1 for malicious, obfuscated software).

Visualizations also show that it is very important to determine the moment of «retraining» of the neural network and reverse-enginering the network under study (in this research using time stamps and «inserts») to calculate the total network error. The accuracy of detecting malicious polymorphic software increases after training a neural network with a dataset previously filtered from «noise». Knowing the degree of accuracy of malware detection allows you to set the percentage of detection of malicious polymorphic software when configuring ML IDS without going beyond the limits of reliable operation. The accuracy of detection of malicious polymorphic software (%) at the third epoch of training and five iterations without the use of a local source of exact time is presented in Tab. 1, with the use of a local source of exact time in Tab. 2.

Table 1

Polymorphic malware detection accuracy during the third training epoch (%) without using a local source of accurate time

malware	I iteration	II iteration	III iteration	IV iteration	V iteration
abc	12,3	13,1	15,8	21,3	17,3
cheeba	12,0	13,4	16,3	26,7	16,1
december_3	12,7	13,2	15,7	29,4	13,7
stasi	11,6	14,2	16,5	26,4	16,2
otario	12,8	27,6	17,9	28,3	14,9
dm	11,1	12,6	14,7	21,5	12,7
v-sign	13,5	14,9	15,9	24,3	13,5
tequila	13,8	15,8	13,4	22,7	23,8
flip	12,1	14,4	18,3	21,2	16,2

Table 2

Polymorphic malware detection accuracy during the third training epoch (%) using a local source of accurate time

malware	I iteration	II iteration	III iteration	IV iteration	V iteration
abc	8,3	9,1	25,8	32,7	16,5
cheeba	10,0	11,6	26,3	30,4	14,7
december_3	9,3	10,6	20,6	32,5	19,6
stasi	8,4	10,8	23,5	34,1	25,2
otario	7,5	9,5	20,4	32,5	16,4
dm	8,7	11,6	24,6	27,2	16,7
v-sign	11,3	13,6	23,4	25,7	19,2
tequila	8,1	17,8	27,5	31,5	24,3
flip	10,7	11,4	24,2	35,8	27,2

As can be seen from Tab. 1 and Tab. 2, the use of a local source of accurate time has increased the percentage of malicious software detection.

The all research results are presented in [10].

Conclusion

The paper considers the change in the *softmax* logistic function for different sizes of datasets obtained by the method of piecewise-context hashing of polymorphic obfuscated software. A timestamp value is embedded in the datasets.

The following results are obtained:

Reducing the «weight» of logits with their quantitative increase, reduces the overall error of the convolutional neural network, but reduces its «controllability». In particular, with large «weights», the activation of the neural network occurred after 5-7 layers, a decrease in the «weight» of logits lowered the «activation» threshold to 1-2 layers. The most applicable results are obtained with a CTPH size of 20 and/or 40 bytes and a maximum value of the value vector equal to 40960. In all cases of computing, the availability of quality data sets is essential.

References

1. I.Goodfellow, Y.Bengio, A.Courville. Deep Learning, (2016)// 802, MIT Press.
2. H.Kim, P.Parviainen, T.Berge,K.Malde: Inspecting class hierarchies in classification-based metric learning models. [Online].Available: https://arxiv.org/abs/2301.11065
3. X.Yu, L.Ying: On the Global Convergence of Risk-Averse Policy Gradient Methods with Dynamic Time Consistent risk Measures. [Online].Available: https://arxiv.org/abs/2301.10932
4. M.Azizmalayeri, et al: A Data-Centric Approach for Improving Adversarial Training Through the Lens of Out-of Distribution Detection. [Online].Available: https://arxiv.org/abs/2301.10454
5. Jamgharyan T. «Research of Obfuscated Malware with a Capsule Neural Network», Mathematical Problems of Computer Science 58, pp. 67-83, 2022. doi: 10.51408/1963-0094
6. Stallings W., Cryptography and Network Security, Principles and Practice, fifth edition (2011)// 900, Prentice Hall.
7. S.Pontes-Filho et al: Towards the Neuroevolutional of Low-level Artifical General Intelligence. [Online].Available: https://arxiv.org/abs/2207.13583
8. Chattopadhyay A.et al. Variational Information Pursuit for Interpretable Predictions [Online].Available: https://arxiv.org/abs/2302.02876
9. Sharma S. wt al. Learning Prototype Classifiers for Long-Tailed Recognation [Online].Available: https://arxiv.org/abs/2302.00491
10. Software source code and full research results. [Online].Available: https://github.com/T-JN?tab=repositories

ՄԵՔԵՆԱՅԱԿԱՆ ՈՒՍՈՒՑՈՒՄՈՎ ՆԵՐԽՈՒԺՈՒՄՆԵՐԻ ՀԱՅՏՆԱԲԵՐՄԱՆ ՀԱՄԱԿԱՐԳԻ ԿԱԶՄՈՒՄ ԲԱԶՄԱԶԱՓ ԼՈԳԻՍՏԻԿ ՖՈՒՆԿՑԻԱՅԻ ՀԵՏԱԶՈՏՈՒՄ

Ջամղարյան Թ.Վ.
Հայաստանի ազգային պոլիտեխնիկական համալսարան

Հոդվածում ներկայացված է լոգիստիկ ֆունկցիայի փոփոխության մոդելի հետազոտության արդյունքները: Հետազոտությունն իրականացվել է *abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip* վնասաբեր պոլիմորֆ ծրագրային

ապահովման ելակետային կոդի հիման վրա կառուցած տվյալների հավաքածուներով: Հետազոտությունն իրականացվել է մեքենայական ուսուցումով ներխուժումների հայտնաբերման համակարգի ճշգրտությունը գնահատելու համար: Որպես ուսումնասիրության մաթեմատիկական ապարատ՝ ընտրվել է բազմաչափ softmax լոգիստիկ ֆունկցիան: Իրականացվել է ծրագրային ապահովման իրագործման մոդելավորում տարբեր կրկնություններում և արդյունքների արտացոլում:

Բանալի բառեր. մեքենայական ուսուցում, տվյալների հավաքածու, անորոշ հեշավորում, ներխուժումների հայտնաբերման համակարգ, բազմաչափ լոգիստիկ ֆունկցիա, հավանականությունների բաշխման վեկտոր, softmax.

ИССЛЕДОВАНИЕ МНОГОМЕРНОЙ ЛОГИСТИЧЕСКОЙ ФУНКЦИИ В СИСТЕМЕ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ С МАШИННЫМ ОБУЧЕНИЕМ

Джамгарян Т.В.
Национальный политехнический университет Армении

В статье представлены результаты исследования модели изменения многомерной логистической функции. Исследование проводилось на наборах данных сформированных на основе исходного кода вредоносного полиморфного программного обеспечения abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip. Исследование проводилось с целью оценки точности работы разработанной системы обнаружения вторжений с машинным обучением. В качестве математического аппарата для исследования выбрана многомерная логистическая функция softmax. Проведено моделирование работы программного обеспечения при разных итерациях и визуализация результатов.

Ключевые слова: машинное обучение, наборы данных, зловредное ПО, многомерная логистическая функция, вектор распределения вероятностей, *softmax*.

Submitted on 11.04.2023
Sent for review on 21.04.2023
Guaranteed for printing on 03.07.2023