Preventing Data Poisoning Attacks By Using Generative Models

Merve Aladag
Computer Science and Engineering
Istanbul Sehir University
Istanbul, Turkey
Email: mervealadag@std.sehir.edu.tr

Ferhat Ozgur Catak
Information Security and Communication Technology
NTNU Norwegian University of Science and Technology
Gjøvik, Norway
Email: ferhat.o.catak@ntnu.no

Ensar Gul
Computer Science and Engineering
Istanbul Sehir University
Istanbul, Turkey
Email: ensargul@sehir.edu.tr

***Abstract*—At the present time, machine learning methods have been becoming popular and the usage areas of these methods have also increased with this popularity. The machine learning methods are expected to increase in the cyber security components like firewalls, antivirus software etc. Nowadays, the use of this type of machine learning methods brings with it various risks. Attackers develop different methods to manipulate different systems, not only cyber security components, but also image detection systems. Therefore, securing machine learning models has become critical. In this paper, we demonstrate a data poisoning attack towards classification method of machine learning models and we also proposed a defense algorithm which makes machine learning models more robust against data poisoning attacks. In this study, we have conducted data poisoning attacks on MNIST, a widely used character detection data set. Using the poisoned MNIST dataset, we built classification models more reliable by using a generative model such as AutoEncoder.**

***Keywords*—Data Poisoning, Support Vector Machine, Machine Learning, Optimization**

I. Introduction

In recent years, with the rapid advances in technology, there has been a tremendous increase in the usage of systems and the data produced by these systems [1]. With this increase in the rate of produced data, the data can now be accessed by the systems directly and the systems are able to use these produced data without being programmed in detail. Systems are able to provide meaningful results by learning the data on their own and they are provided by machine learning methods which are an artificial intelligence application and these methods are now used in cyber security areas [2].

Moreover, with the increase in cybercrime, machine learning methods are used to detect malicious behavior in systems, malware, and malicious traffic on the network [3]. These and similar machine learning solutions in the field of cyber security are also being used in commercial products. Some of these products are; *Exabeam, Fortscale, E8 Security*. Attackers have begun to use adversarial machine learning methods to avoid such detection methods and to violate the security of relevant areas by using gaps in machine learning techniques.

The adversarial machine learning has been used to describe the technique employed in the field of machine learning which attempts to fool models through malicious input in either training time or decision time. The methods of adversarial machine learning are divided into two according to the time of the attack:

- *Data Poisoning Before Model Training*: The attacker changes some labels of training data set before the model learns.
- *Data Preparation According to Trained Model*: The attacker enforces model to produce inverse of true output data after model is trained.

Both types of attacks are extremely dangerous considering the consequences and effects. When we examine commercial machine learning products, it is seen that data poisoning attacks creates a greater threat. Almost all commercial products require the training data set from the installation organization itself. For attackers, poisoning these datasets can be quite easy.

In the scope of this study, we first proposed an algorithm that acknowledges that the attacker has access to data, and he conducts a data poisoning attack prior to model training. Before model training, the attacker manipulates data by changing the labels of the malicious samples with labels of normal samples, and he shows malicious samples as normal samples. Then, the attacker adds these manipulated data samples in the training data set, so that he manipulates the data before the model is trained. On the other hand, we proposed a defense mechanism to reduce the impact of this data poisoning attack and to make the model robust to these types of attacks.

The rest of the paper is organized as follows: The related work is presented in Section II. Section III gives brief preliminary information. Section IV presents our system model in details. Our model evaluation and real system test results are presented in Section V. The concluding remarks are given in Section VI.

II. Related Work

By controlling several devices, attackers are able to manipulate the data that is created for machine learning applications [4]. Within the scope of such attacks, Biggio et al. performed the first systematic poisoning attack against the linear regression model and proposed the TRIM algorithm, which is a stronger algorithm than the classical models used for the detection of poisoning points on the training data [4].

Khalid et al. identified potential machine learning threats and presented examples of training data poisoning and adversarial machine learning attacks from these threats. They also recommended a training data poisoning attack that has less impact on accuracy [1].

Paudice et al. proposed a data poisoning attack that changed the labels of labeled data and influences the ability to classify classification method from machine learning algorithms. They then proposed a defense mechanism based on the K-NN algorithm to ensure label clearance against this attack that aimed to identify data labels which are manipulated by the attacker [5].

In the Internet of Things (IoT) area, products collect data dynamically and they learn these data online [6]. Therefore, products in the IoT area are particularly vulnerable to data poisoning attacks. In their study, Nathalie et al. presented a new methodology for identifying and filtering malicious data to train a supervised learning model in the IoT field [6].

III. Preliminary Information

In this section, we will briefly describe the methods we have used in the context of our study.

A. Data Poisoning Attack

There is a need for a data set that has reached a certain volume and a machine learning model suitable for this data set to use learning methods in a system, device or product. For this reason, in commercial products, the model is created according to the customer data, so there is no ready model for each type of data. Thus, when a product which uses machine learning methods is sold to a customer, the data structure of the customer is monitored at a certain time interval in order to generate a model according to the customer data and needs. With this way, the usual structure of the data is profiled by the product.

In this study, we assume that the attacker has access to the data and he can manipulate instances where normal behavior is learned and he shows abnormal behavior data as normal behavioral data. In this way, the attacker enables the machine learning algorithms to perform the learning function on the manipulated data.

B. Auto-Encoder

Auto-encoder is a generative model of the artificial neural network that reproduces the data by learning the structure of the data with no labels. The structure and features of the data are learned with this model and the data is tried to be re-created. Figure 1 shows an auto-encoder model. Details of an auto-encoder model is given at Figure 1.

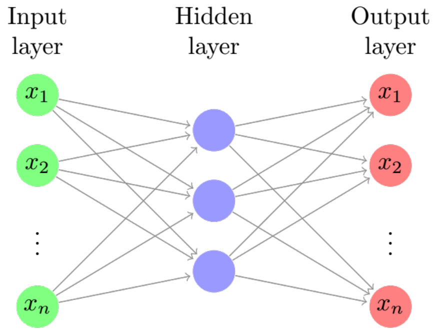


Fig. 1. Auto-Encoder model

C. Support Vector Machines

Support Vector Machines (SVM) is a supervised machine learning algorithm which can be used for both classification and regression problems. In this study, we have used the classification version of SVM algorithm. This algorithm is used for binary classification problems which are consisting of two different classes of the target value (i.e. $y_i \in \{-1, +1\}$) [7].

Consider a data set $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \cdots (\mathbf{x}_n, y_n)$ into two different classes, where input instances are n-dimensional feature vectors $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$. If the input data set with two classes can be separated by a linear hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$. After solving optimal values of $\mathbf{w}$ and b using Lagrange multipliers, then the classifier hypothesis becomes

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{1}$$

where α_i and b are calculated by using an SVM algorithm.

IV. System Model

In this research, firstly labeled data was obtained. Then, an optimization-based data poisoning attack was performed on the data set. After that, with the auto-encoder model, a defense mechanism against this attack was proposed. The general steps of the study are shown in Figure 2.

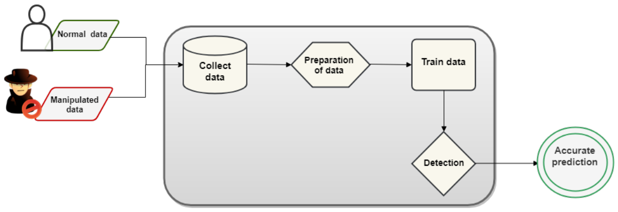


Fig. 2. General steps of the study

This study was prepared using Python programming language and included the Python libraries Keras, Scikit-learn, Scipy, Numpy and Mathplotlib.

A. Manipulated Data

In this study MNIST data set is used. The MNIST data set contains 28×28 pixel image of handwritten instances from 0 to 9. Figure 3 shows an example of MNIST data .

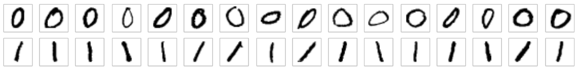


Fig. 3. Example of MNIST data image

The number of features is 784 ($d = 28 \times 28 = 784$). In this research, each feature (pixel value) is normalized by dividing 255, because of gray-scale representation. In addition, we need to create a data set where the target value was composed of two different classes and these classes were determined as $-1, 1$. Because SVM classification algorithm builds only binary models. In order to convert the MNIST data set to the binary data set where the target (class) values are labeled as $-1, +1$, number 5 is selected as the target and the labels which are equal to 5 are labeled as $+1$, and all remaining values which are $0, 1, 2, 3, 4, 6, 7, 8, 9$ labeled as -1. By using this conversion, the multi-class MNIST dataset has been transformed into the detection of a single number for the purpose of using the binary classification problem. Eventually, 92 of 1000 input instances are equal to 5, therefore they are labeled as $+1$, and remaining 908 of 1000 input instances are labeled as -1.

The solution speed of the optimization problems on large volume data is directly proportional to the hardware capabilities. For this reason, in our study, we have selected only 1000 input instances containing all numbers from 0 to 9 with their labels from MNIST data set according to our hardware capacity, and we have divided these 1000 samples into 2 parts and marked 600 samples as training data and the remaining 400 samples as test data set for our experiments.

B. Capability of The Attacker

We assumed that the main purpose of the attacker is to manipulate the data collection phase to poison the training data set. After manipulating the collected data, the attacker can convert some abnormal behavioral samples as normal behavioral samples. Therefore, the classification model learns abnormal behavioral samples as normal behavioral samples. For this purpose, the attacker replaces the abnormally behaved data label of the test data set with the normal behaved data label, and poisons the data by adding the manipulated data to the training data set before the model is in training step.

However, in order to prevent this manipulation from being noticed, an optimization problem occurs for the attacker to decide which of the abnormally behaved data must be changed. By solving this optimization problem, the attacker aims to reduce the classification performance on the manipulated data as much as possible while increasing the classification performance on the manipulated data that the product has.

In order to solve this optimization problem; the cost function of the true data is minimized and the cost function of the manipulated data is maximized. Equation is as follows;

$$\underset{w}{arg\,min} \left(\sum_{\mathbf{x}_i \in D^{(1)}} l(y_i, f(\mathbf{x}_i)) - \sum_{\mathbf{x}_i \in D^{(2)}} l(y_i, f(\mathbf{x}_i)) \right) \tag{2}$$

The total data set consists of two different sets, $\mathcal{D} = \{D^{(1)} \cup D^{(2)}\}$. The $D^{(1)}$ data set consists of samples whose class label is not modified. The attacker changes the label of samples with $+1$ to -1 in the $D^{(2)}$ data set. But the attacker can do this operation within a budget, $\mathcal{B}$. Accordingly, the cost of each sample ($\mathbf{x}_i$) which the class label is changed is b_i. If b_i is 0, it shows that the class label of the relevant example has not been changed; If $b_i = 1$, it shows that the class label of the corresponding example has been changed. Therefore, the constraint of the attacker's optimization problem can not exceed this budget. The new optimization problem using this constraint is as follows;

$$
\begin{aligned}
&arg\,\underset{\mathcal{B}}{min}\left(\sum_{\mathbf{x}_i\in D^{(1)}} l(y_i, f(\mathbf{x}_i)) - \sum_{\mathbf{x}_i\in D^{(2)}} l(y_i, f(\mathbf{x}_i))\right)\\
&\quad s.t. \quad \sum_{i=1}^{i=|D^{(2)}|} b_i \leq \mathcal{B}
\end{aligned}
\tag{3}
$$

With the solution of this optimization problem, the attack is successfully accomplished by finding the poisoning points to reduce the classification performance on the manipulated data, which improve the classification performance of the security product on the manipulated data.

C. Creation of Auto-Encoder Model

The auto-encoders derive the identity function, $h_{\mathbf{w}}(\mathbf{x}) \approx \mathbf{x}$, which will produce very similar results, $\hat{\mathbf{x}} \in \mathbb{R}^m$, to the input vector, $\mathbf{x} \in \mathbb{R}^m$. The *binary cross entropy* function is used as the loss function when creating the auto-encoder model. With this function, the loss calculated for each vector component is not affected by other components. The formulation of the corresponding function is as follows;

$$
\begin{aligned}
l &= \sum_{i=1}^{C'=2} t_i log(f(s_i))\\
&= -t_1 log\left(f(s_1)\right) + (1-t_1) log\left(1-f(s_1)\right)
\end{aligned}
\tag{4}
$$

s_1 and t_1 are the score and basic reality tag for C_1 class respectively. In this study, the compression ratio of the auto-encoder model was fixed to 0.8 and 4 different activation functions were used separately and their effects on the model were investigated. The activation functions used are *Sigmoid, ReLu, Tanh* and *Softsign*. The results obtained with the usage of different activation functions are given in *Experimental Results* section. Furthermore, the effect of the difference in compression ratios on the model was measured while the auto-encoder model was created. In this respect, activation function was fixed as *sigmoid* and 10 $\{0.1, 0.2, 0.3...., 1\}$, different compression ratios were tried. The related results are given in the *Experimental Results* section.

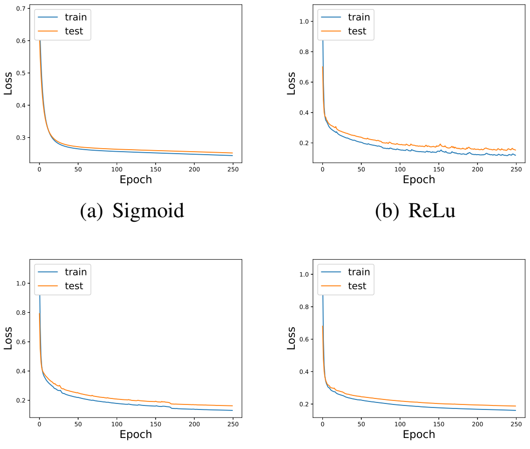
V. Experimental Results

Firstly, the data set was trained by using the Support Vector Machines algorithm and the accuracy on the test data set was observed as 1.0. Then, the data poisoning attack on the test data set was performed and the poisoned data (changed labels) were added to the training data set. In the context of this study, we proposed a mechanism for the data poisoning attack and we passed the training data set to the auto-encoder model regardless of when the attack was carried out in the scope of the proposed mechanism. In this way, we enabled the auto-encoder model to learn the natural structure of true data.

After the natural data was learned by the auto-encoder model, the structure of the poisoned data was estimated by the auto-encoder model. Then, the accuracy of the poisoned data set with the data set estimated by the auto-encoder model was measured and the result was found as 0.40. As it can be seen from these two values, although the attack was successfully carried out, the auto-encoder model marked 0.60 of the data as different from the poisoned data. Which means that the negative (normal behavior) values of 0.60 were predicted as positive (abnormal behavior) by auto-encoder model.

In addition, the performance of this defense mechanism against the attack was monitored in accordance with the activation and compression ratios in the auto-encoder model. Figure 4 shows the training histories of different activation functions and the loss values of each auto-encoder model are measured according to the activation function. When the related results are examined, it can be seen that the loss value of *SoftSign* activation function is converged faster than other activation functions.

Figure 5 shows the training history of the different compression ratios and the accuracy of the auto-coder model in each epoch. In this way, the value that the auto-coder model predicts the closest to the original data was measured. It was observed that compression ratio which gives the best accuracy rate was 1.0.



(a) Sigmoid

(b) ReLu

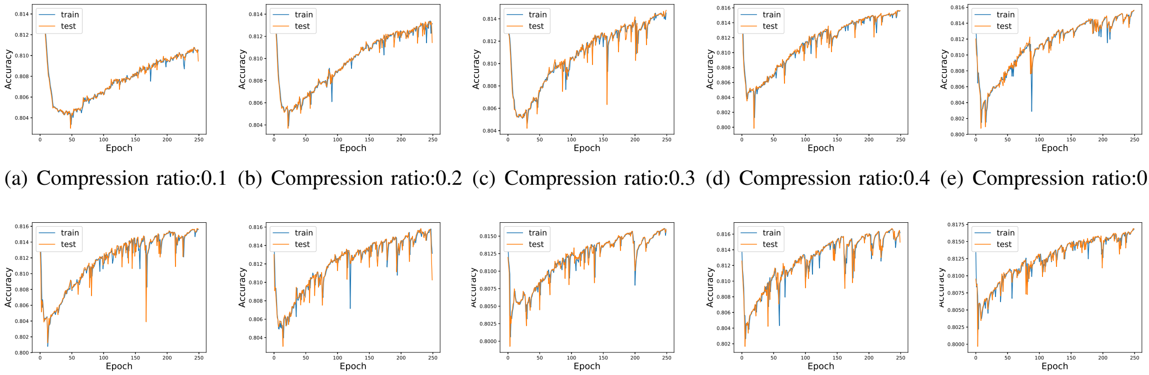
(c) Tanh

(d) SoftSign

Fig. 4. Training history of different activation function

VI. Conclusion and Future Works

With this study, we performed an optimization based data poisoning attack which manipulated the training



(a) Compression ratio:0.1 (b) Compression ratio:0.2 (c) Compression ratio:0.3 (d) Compression ratio:0.4 (e) Compression ratio:0.5

(f) Compression ratio:0.6 (g) Compression ratio:0.7 (h) Compression ratio:0.8 (i) Compression ratio:0.9 (j) Compression ratio:1.0

Fig. 5. Training history of different compression ratio

stage of the classification method from machine learning models. Before the training phase of the classification model, we were able to add manipulated data on the true data so that the model could learn the manipulated data as well. We then proposed the auto-encoder model to make the classification models more robust to such attacks, and by observing the classification performance, we showed that the model marked the manipulated data as it should be. In future studies, we suggest that auto-encoder model diversity and gradient-based optimization can be studied.

References

[1] F. Khalid, M. A. Hanif, S. Rehman, and M. Shafique, “Security for machine learning-based systems: Attacks and challenges during training and inference,” *CoRR*, vol. abs/1811.01463, 2018.

[2] I. Amit, J. Matherly, W. Hewlett, Z. Xu, Y. Meshi, and Y. Weinberger, “Machine learning in cyber-security - problems, challenges and data sets,” *arXiv e-prints*, p. arXiv:1812.07858, Dec. 2018.

[3] M. H. B. R, V. R, and S. K. P, “A short review on applications of deep learning for cyber security,” *CoRR*, vol. abs/1812.06292, 2018.

[4] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, “Manipulating machine learning: Poisoning attacks and countermeasures for regression learning,” *CoRR*, vol. abs/1804.00308, 2018.

[5] A. Paudice, L. Muñoz-González, and E. C. Lupu, “Label sanitization against label flipping poisoning attacks,” *arXiv e-prints*, p. arXiv:1803.00992, Mar. 2018.

[6] N. Baracaldo, B. Chen, H. Ludwig, A. Safavi, and R. Zhang, “Detecting poisoning attacks on machine learning in iot environments,” in *2018 IEEE International Congress on Internet of Things (ICIOT)*, July 2018, pp. 57–64.

[7] W. Chu, S. S. Keerthi, and C. J. Ong, “A general formulation for support vector machines,” in *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP ’02.*, vol. 5, Nov 2002, pp. 2522–2526 vol.5.