1 Introduction

In information technology, the CIA triad of Confidentiality, Integrity, and Availability is at the heart of information security, whose primary focus is the balanced protection of the confidentiality, integrity and availability of data. Recently, the security of machine learning has received a lot of attention. Different types of attacks against learning algorithms’ CIA have been designed and analyzed. These attacks can be broken down into three types: privacy attacks, integrity attacks and availability attacks. Privacy attacks mean the attacker obtains private information about the system, its users or data by reverse-engineering the learning algorithm or program. e.g., Fredrikson et al. (2015), Fredrikson et al. (2014) and Shokri et al. (2017). Integrity attacks mean the attack compromise system’s normal functionality by modifying the analyzed sample’s features to cause classifier to misclassify the instances. e.g., Goodfellow et al. (2014), Biggio et al. (2013) and Carlini et al. (2017). Availability attacks mean the attack cause a denial of service of the system, by being poisoned in training data set and increasing the classification error. e.g., Biggio et al. (2012), Mei and Zhu et al. (2015) and Munoz-Gonz et al. (2017).

From attack method, there are two types of attack the evasion attack and poisoned attack. the evasion attack violates system integrity, by modifying the analyzed sample’s features to evade detection by the model, particularly on the ability to perturb inputs so that they are misclassified by the model. For example, adversarial examples can fool the classifier to predict a test instance as the attacker’s wish by adding adversarial noise to the instance. There are two types of poisoning attacks, Generic and Specific. Generic poisons training data to violate system availability i.e. to cause a denial of service. Specific poisons training data to violate system integrity i.e. the misclassification of specific data points, respectively.

Several schemes have been proposed to conduct poisoning attack against Generic poisoning attack. by Biggio et al. (2012) and later by a number of others like Mei and Zhu et al. (2015) and Munoz-Gonz et al. (2017). However, we have not seen many works about Specific poisoning attacks against multi-classes NNs. In this paper, we apply brute-force algorithm to conduct a Specific poisoning attacks against multi-classes NNs. By doing the experiments on CNN (two convolutional layers and two fully-connected layers) and 32 layers ResNet on the data sets MNIST and CIFAR-10. I find these two popular algorithms could be specific poisoned.

2 Algorithm and experiments

In information security, a brute-force attack aims to attack privacy by systematically checking all possible passwords and passphrases until the correct one is found. As the password’s length increases, the amount of time, on average, to find the correct password increases exponentially. Alternatively, the attacker can attempt to guess the key which is typically created from the password using a key derivation function.

In this paper, I apply brute-force attack as a Specific poisoning attack on image classifiers. Adding one instance from test data set with another label into the training data set, retrain the classifier and check whether the prediction equals the poisoning label. By brute force
attack, attacker can get potential poisoning instances and poisoning labels from the test data set and use these instances to deliver attack. To evaluate brute-force attack strategy I do the experiment on 2 classifiers: four layers CNN and 32 layers ResNet on 2 data sets MNIST and CIFAR10.

Brute-force attack uses brute-force search, also known as generate and test, brute-force search algorithms is a very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement. A brute-force approach for classification algorithm in machine learning would examine all possible arrangements of target test instances and target labels, and, for each arrangement, check whether the target test instances are poisoning predicted as target labels.

While a brute-force search is simple to implement, and will always find a solution if it exists, its cost is proportional to the number of candidate solutions. In target poisoning attack against classification algorithm, the number of candidate solutions grow very fast as the size of the combination of target instances and target labels increases. For instance, in MNIST data set, the number of one target image candidate tested will be the given number of labels excluding the true label, I test 1,000 instances in test data set and do training process 9,000 times.

2.1 brute-force search
To apply brute-force search to target poisoning attack against classification algorithm problems, there are four procedures: first, valid, output and next:

- first (P): generate a first candidate solution by training the model with original data set adding the target instance with target label.
- valid (P, c): check whether c the model predicts the target instance as the target label.
- output (P, c): use the solution c of P as appropriate to the application.
- next (P, c): generate the next candidate for P after the current one c.

The brute-force method is then expressed by the algorithm:

\[
c \leftarrow \text{first}(P)
\]

\[
\text{while } c \neq \Lambda \text{ do}
\]

\[
\quad \text{if valid}(P, c) \text{ then output}(P, c)
\]

\[
\quad c \leftarrow \text{next}(P, c)
\]

\[
\text{end while}
\]

(The first procedure should return \( \Lambda \) if there are no candidates at all for the instance P.)

2.2 brute-force search to targeted poison classifier
In a classification task, given the instance space \( X \) and the label space \( Y \), the learner aims to estimate the underlying function \( f \) that maps \( X \rightarrow Y \).

Label space \( Y \) is set of the Number of classes, range \( (0, N) \), \( N=10 \)

Given a training set \( D_{tr} \), and \( D_{test} = \{x_i, y_i\}_{i=1}^m \) with \( m=1000 \)

The poisoned training set \( D_p = \{D_{tr}, (x_{\text{target}}, y_{\text{target}})\} \)

let \( \theta \) denote the model that the machine learning algorithm learns from training data \( D_{tr} \) and \( \theta \) from training data \( D_p \).

brute-force search algorithm:

For \( x_i \) in \( \{x_i\}_{i=1}^m \)
\(X_{\text{target}} = x_i\)

For \(y_{\text{target}}\) in range \((0, N)\)

If \(y_{\text{target}} \neq y_i\)

new training set \(D_p = \{D_{tr}, (X_{\text{target}}, y_{\text{target}})\}\)

train \(\theta\) with new training set \(D_p\)

if \(f(x_{\text{target}} | \theta_p) = y_{\text{target}}\)

output \((X_{\text{target}}, y_{\text{target}})\)

2.3 Verify

Because “machine learning models remain mostly black boxes” [Ribeiro et al. (2016)], it is unable to explain the reasons behind their predictions. And it is difficult to prove that the output is caused by the poisoning instance. I go one layer backward from the output layer and analyze the output of the softmax layer. Those instances with low score of original label by trained with original \(D_{tr}\) could not be predicted correctly constantly are not poisoned successfully, as well as those instances with low score of targeted label by trained with poisoned \(D_p\) that could not be predicted as targeted labels constantly. There are some instances that have high score of original label by trained with original \(D_{tr}\) and high score of targeted label by trained with poisoned \(D_p\), and these instances are poisoned successfully.

\(S(x_{\text{target}}, y_{\text{target}}, \theta_D)\) is the output of softmax layer of \(\theta_D\) on \((X_{\text{target}}, y_{\text{target}})\).

For \(i\) in \(\{x_i\}_{m}^{-1}\)

\(X_{\text{target}} = x_i\)

For \(y_{\text{target}}\) in range \((0, N)\)

If \(y_{\text{target}} \neq y_i\)

new training set \(D_p = \{D_{tr}, (X_{\text{target}}, y_{\text{target}})\}\)

train \(\theta\) with new training set \(D_p\)

if \(f(x_{\text{target}} | \theta_p) = y_{\text{target}}\) and \(S(x_i, y_i, \theta_D) > 0.8\) and \(S(x_i, y_{\text{target}}, \theta_D) > 0.8\)

output \((X_{\text{target}}, y_{\text{target}})\)

2.4 Exhaustive search algorithm

One way to speed up a brute-force algorithm is to reduce the search space, that is, the set of candidate solutions. By analyzing the successful and unsuccessful instances, I split the search space into 3 types:

Type1: \(S(x_i, y_{\text{target}}, \theta_D) < 0.01\)

Type2: \(0.01 < S(x_i, y_{\text{target}}, \theta_D) \leq 0.2\)

Type3: \(0.2 < S(x_i, y_{\text{target}}, \theta_D)\)

Type3 will cause \(S(x_i, y_i, \theta_D) < 0.8\), so it will not be considered.

The successful rate of type1 is 0.00012, while type2 is 0.04

2.5 Exhaustive search algorithms on Resnet

In first 1000 instances of test set of CIFAR10, the successful rate of type2 is 0.06 by exhaustive search algorithms.

2.6 Experiment Resources

The resources in this experiment is a pc server with GPU GTX1060, the time of brute force on 1000 instances of MNIST is 24 hours, which means 10 seconds per attack to 9000 potential cases with 3 successful cases, the time of brute force on 1000 instances of CIFAR-10 is 24*10 hours, which means 4320 seconds per attack to 200 potential cases with 2 successful cases.
3 conclusions
As the computational capabilities increase, the attackers can do brute force attack on the target algorithms and target data set with more GPUs to accumulate the successful poisoned instances to prepare real attack case. And these successful poisoned instances also provide a chance for us to understand the black box by understanding why the models change prediction when adding one poisoned instance.

Reference:


