A C-IFGSM Based Adversarial Approach for Deep Learning Based Intrusion Detection

Speaker: Yingdi Wang
Email: 17120482@bjtu.edu.cn
Part one: Introduction
Part two: Constraint-IFGSM
Part three: Experiments and Results
Part four: Conclusion
Part One

Introduction
Machine learning algorithms have helped humans solve problems in various fields. Machine learning algorithms have also been increasingly applied to security fields, such as spam detection, malware detection and intrusion detection. There are advantages and disadvantages of intrusion detection based on machine learning. Intrusion detection classifiers can detect unknown types of attacks, but they may also be attacked by adversarial examples.
Adversarial examples will make the machine learning model misclassified by adding a slight modification to real examples, which cannot be distinguished by humans.

Especially for neural network-based intrusion detection, the misclassified of anomaly network traffic by intrusion detection systems may lead to malicious attacks on systems.
The methods of crafting adversarial examples in different applications are restricted by the specific characteristics of dataset.

- Crafting text adversarial examples needs to consider the discrete features of words and the grammar of sentences.

- The methods for crafting malware adversarial examples need to consider both the type of features and its functionality.

- An attacker can construct a pair of glasses to allow the wearer to evade recognition by the facial recognition system by adding noise to a specific area.

- Crafting text adversarial examples needs to consider the discrete features of words and the grammar of sentences.

\[
\hat{i}_{max} = \arg \max_{j \in \Gamma \cap I, X_j = 0} \frac{\partial F_y(X)}{\partial X_j}
\]
crafting adversarial examples for intrusion detection classifiers

- There have been researchers using FGSM to generate adversarial examples for NSL-KDD dataset.

- Although FGSM can add a slight noise to the original examples to cause deep neural networks misclassification, it cannot guarantee that the types and relationship of features in adversarial examples are consistent with original examples.
Part Two

Constraint-IFGSM
This paper proposed a C-IFGSM based method which can adapt to complex datasets with multiple types of features and multiple relationship among features.

The intrusion adversarial examples need to satisfy the following three parts:

- making the intrusion detection classifiers go wrong
- keeping the feature types unchanged
- keeping the feature relationships unchanged
Analysis of network traffic data

01 A variety of feature types
   Due to the complexity of network traffic, there may be various types of statistical features, such as character features, discrete features and continuous features.

02 Specific malicious payloads
   Network traffic data has specific attack effect, and its malicious payload may be reflected in some features, or fixed feature format.

03 Constraint relationships between features
   There may be a certain constraint relationship between the features of network traffic data, which may be caused by some certain attack patterns or correlation between statistical methods of some features.
## Multi-type Features of NSL-KDD Dataset

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
<th>Type</th>
<th>No.</th>
<th>Feature</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>duration</td>
<td>continuous</td>
<td>23</td>
<td>count</td>
<td>continuous</td>
</tr>
<tr>
<td>2</td>
<td>protocol_type</td>
<td>nominal</td>
<td>24</td>
<td>srv_count</td>
<td>continuous</td>
</tr>
<tr>
<td>3</td>
<td>service</td>
<td>nominal</td>
<td>25</td>
<td>serror_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>4</td>
<td>flag</td>
<td>nominal</td>
<td>26</td>
<td>srv_serror_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>5</td>
<td>src_bytes</td>
<td>continuous</td>
<td>27</td>
<td>reror_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>6</td>
<td>dst_bytes</td>
<td>continuous</td>
<td>28</td>
<td>srv_rerror_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>7</td>
<td>land</td>
<td>discrete</td>
<td>29</td>
<td>same_srv_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>8</td>
<td>wrong_fragment</td>
<td>continuous</td>
<td>30</td>
<td>diff_srv_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>9</td>
<td>urgent</td>
<td>continuous</td>
<td>31</td>
<td>srv_diff_host_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>10</td>
<td>hot</td>
<td>continuous</td>
<td>32</td>
<td>dst_host_count</td>
<td>continuous</td>
</tr>
<tr>
<td>11</td>
<td>num_failed_logins</td>
<td>continuous</td>
<td>33</td>
<td>dst_host_srv_count</td>
<td>continuous</td>
</tr>
<tr>
<td>12</td>
<td>logged_in</td>
<td>discrete</td>
<td>34</td>
<td>dst_host_same_srv_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>13</td>
<td>num_compromised</td>
<td>continuous</td>
<td>35</td>
<td>dst_host_diff_srv_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>14</td>
<td>root_shell</td>
<td>discrete</td>
<td>36</td>
<td>dst_host_same_srv_port_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>15</td>
<td>su_attempted</td>
<td>discrete</td>
<td>37</td>
<td>dst_host_srv_diff_host_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>16</td>
<td>num_root</td>
<td>continuous</td>
<td>38</td>
<td>dst_host_serror_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>17</td>
<td>num_file_creations</td>
<td>continuous</td>
<td>39</td>
<td>dst_host_srv_serror_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>18</td>
<td>num_shells</td>
<td>continuous</td>
<td>40</td>
<td>dst_host_rerror_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>19</td>
<td>num_access_files</td>
<td>continuous</td>
<td>41</td>
<td>dst_host_srv_rerror_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>20</td>
<td>num_outbound_cmds</td>
<td>continuous</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>is_hot_login</td>
<td>discrete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>is_guest_login</td>
<td>discrete</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Basic features (1-9):** contain the basic connection information of data packets.
- **Content features (10-22):** contain some useful payload in data packets.
- **Traffic features (23-41):** show the traffic information presented by the current connection and its nearby connections. Traffic features are calculated based on two statistics method, a 2-second time window (23-31) and a 100-connection window (32-41).
multi-relationship features of NSL-KDD dataset

\[ \rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y} \]

- The PCC between the two features is calculated by the ratio of the covariance to the standard deviation between the two features. The larger the absolute value, the stronger the correlation between the two features.
- The correlation between features mainly exists in the traffic features.

- The Pearson Correlation Coefficient matrix heat map of numeric and discrete features in NSL-KDD dataset.
**FGSM & IFGSM**

**Fast Gradient Sign Method, FGSM**

- The formula: \( X^{adv} = X + \epsilon \text{sign}(\nabla x J(X, Y_{true})) \)
- The adversarial example will be obtained by moving the original example \(X\) one step in the direction of the gradient of loss function \(J(\theta, x, y)\).

![Graph](image)

**Iterative Fast Gradient Sign Method, IFGSM**

- IFGSM added multiple iterations with small step size to FGSM.
- The formula: \( X_0^{adv} = X, \ X_{N+1}^{adv} = \text{Clip}_{X,\epsilon}(X_N^{adv} + \alpha \text{sign}(\nabla x J(X_N^{adv}, Y_{true}))) \)
- To ensure the perturbation within the range of \(\epsilon\), there is a relationship between \(\alpha\) and the number of iterations \(T\), \(\alpha = \epsilon/T\)
C-IFGSM Based Adversarial Approach

1 Data Processing
Due to the complexity of network traffic features, it is necessary to preprocess the dataset before building the classification model and crafting intrusion adversarial examples.

2 DNN-based Intrusion Detection
IFGSM is a white-box attack algorithm, which means that it needs to obtain the parameters of the model in advance. So we first need to train a deep learning model for intrusion detection.

3 Crafting Intrusion Adversarial Examples
First part is C-IFGSM algorithm, where we add a constraint matrix to IFGSM. The second part is discretization.
Data Preprocessing

- **One-Hot Encoding.** One-hot encoding can convert the nominal features to numeric features. The number of bits for a numeric feature is the same with the number of its categories. In this way, for each category of a numeric feature, there is only one bit is 1, the others are all 0.

- **Protocol_type:** tcp, udp, icmp, udp

- After one-hot encoding, they will be expressed as \([1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]\)

- Therefore, after one-hot encoding, and the total number of features will change from 41 to 122.

- **Normalization.** After one-hot encoding, there will be a large range difference between discrete and continuous features, which will affect the weights of features. Therefore, we normalize the features to make them all in range of [0,1].

- **Attack Types Mapping.** The NSL-KDD dataset contains 39 types of attack, which belong to 4 categories: Denial of Service(DOS), Probe, Remote to Local(R2L), and User to Root(U2R). We map 39 attack types to 4 categories of attacks and build up an intrusion detection classifier of five-classification(4 attack categories and 1 normal category).
Constraint Iterative Fast Gradient Sign Method (C-IFGSM)

- C-IFGSM adds a constraint matrix $C_n$ to IFGSM. By setting the constraint matrix $C_n$, C-IFGSM can restrict the modification of features in original examples.

$$X_0^{adv} = X, \quad X_{N+1}^{adv} = \text{Clip}_{X,\epsilon}\{X_N^{adv} + \alpha C_n \odot \text{sign}(\nabla_x J(X_N^{adv}, Y_{true}))\}$$

- Hadamard product is a matrix product operation that multiplies the elements in the same position of two matrixes without changing the dimension of the matrix.

$$X^{adv} = X + \alpha C_n \odot \text{sign}(\nabla_x J(X_i^{adv}, Y_{true}))$$

$$= X + \alpha C_n \odot S_n$$

$$= X + \alpha [c_i \times s_i]$$

- By setting the constraint matrix $c_i \times s_i$, we can control the noise value added to the original examples.

  - Setting $c_i$ to 1, the modification will completely depend on the $s_i$ in sign matrix.
  - Setting $c_i$ to 0, the coefficient $\alpha$ will not work and feature $x_i$ will not be modified.
  - Setting $c_i$ to a specific can change the scale of modification and control the value of perturbation added to feature $x_i$. 
Discretization Process

- Because the added perturbation can be a small continuous value, so the nominal features and discrete features in the adversarial examples will change to continuous feature.

- **Discrete Features.** For discrete feature whose value can only be 0 or 1, we can discretize it with a boundary of 0.5.

  \[
  \text{IF } x_i^{\text{adv}} \geq 0.5, \text{THEN } x_i^{\text{adv}} = 1; \\
  \text{IF } x_i^{\text{adv}} < 0.5, \text{THEN } x_i^{\text{adv}} = 0
  \]

- **Nominal Features.** For a nominal feature \(x_i\), after one-hot encoding, \(x_i\) is convert to \([k, k + n]\), where \(n\) is the total number of categories. There is only one feature is assigned 1, and the others are all 0. So we setting the feature with the largest value in \([k, k + n]\) as 1, and setting the rest features as 0.

  \[
  \max(x_i) = 1, i \in [k, k + n] \\
  x_j = 0, \forall j \in [k, k + n] - i
  \]
C-FGSM based Intrusion Adversarial Example Crafting

Algorithm 1 Crafting Adversarial Examples for Intrusion Detection

| Input:  | $D_{original}$, $C_n$, $\alpha$, $T$ |
| Output: | $D_{adversarial}$ |

1. $D_{original} = D_{training} + D_{testing}$
2. $DNN \xrightarrow{D_{training}} F(X)$
3. for each example $X$ in $D_{testing}$ do
   4. Initialize $X^{adv} = X$
   5. for $t = 1, 2, ..., T$ do
      6. setting the constraint matrix $C_n$;
      7. $X^{adv} = X^{adv} + \alpha C_n \odot \text{sign}(\nabla_x J(F(X), y_{true}))$
   8. end for
9. for each feature $x_i$ in $X^{adv}$ do
   10. if $x_i^{adv}$ is a discrete feature then
      11. $f(x_i^{adv}) = \begin{cases} x_i^{adv} = 1 & x_i^{adv} \geq 0.5 \\ x_i^{adv} = 0 & x_i^{adv} < 0.5 \end{cases}$
   12. else if $x_i^{adv}$ belongs to a nominal feature $[x_k^{adv}, x_{k+n}]$ then
      13. $f([x_k^{adv}, x_{k+n}]) = \begin{cases} \max(x_i) = 1 & i \in [k, k+n] \\ x_j = 0 & \forall j \in [k, k+n] - i \end{cases}$
   14. else
      15. continues
   16. end if
17. end for
18. end for
19. return $D_{adversarial}$

- **Input:** The original dataset $D_{original}$ is used to initialize the adversarial examples, the single step size $\alpha$ and the number of iterations $T$ specify the iterative process of generating the adversarial examples.

- **Output:** The adversarial dataset $D_{adversarial}$

Under the limitation of constraint matrix $C_n$, the adversarial examples are generated iteratively. In each iteration process, the constraint matrix $C_n$ is set and updated according to the constraint between features.
Part Three

Experiments and Results
Experiment Setup

Experiment Environment

- Python 3.6 was used as the programming language, JetBrains PyCharm as IDE for adversarial examples generation experiments.
- The pytorch 0.4.1 machine learning framework was used to build the intrusion detection classification model and generate the intrusion adversarial examples.

NSL-KDD Dataset

- The distribution of each category of data in the training and testing sets of NSL-KDD.

<table>
<thead>
<tr>
<th>Categories</th>
<th>DOS</th>
<th>Probe</th>
<th>R2L</th>
<th>U2R</th>
<th>Normal</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>45,927</td>
<td>11,656</td>
<td>995</td>
<td>52</td>
<td>67,373</td>
<td>126,003</td>
</tr>
<tr>
<td>Testing Set</td>
<td>7,460</td>
<td>2,421</td>
<td>2,885</td>
<td>67</td>
<td>9,711</td>
<td>22,544</td>
</tr>
<tr>
<td>Total</td>
<td>53,387</td>
<td>14,077</td>
<td>3,880</td>
<td>119</td>
<td>77,084</td>
<td>148,547</td>
</tr>
</tbody>
</table>
We compared the performance of several widely used neural network models, and selected the model with the highest accuracy as the target of adversarial attack.

### Algorithms Accuracy

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Neural Network (DNN)</td>
<td>0.79</td>
</tr>
<tr>
<td>Convolutional Neural Network (CNN)</td>
<td>0.76</td>
</tr>
<tr>
<td>Multilayer Perceptron (MLP)</td>
<td>0.74</td>
</tr>
</tbody>
</table>

Because the adversarial examples have transferability in neural network structures, so it is representative to select basic DNN model as the attack target.

### DNN model

The network structure of DNN is shown in the table below.

<table>
<thead>
<tr>
<th>Layers</th>
<th>Active Function</th>
<th>Input×Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
<td>Batchsize×122</td>
</tr>
<tr>
<td>Hidden Layer 1</td>
<td>ReLu</td>
<td>122×512</td>
</tr>
<tr>
<td>Hidden Layer 2</td>
<td>LeakyReLu</td>
<td>512×512</td>
</tr>
<tr>
<td>Hidden Layer 3</td>
<td>LeakyReLu</td>
<td>512×256</td>
</tr>
<tr>
<td>Hidden Layer 4</td>
<td>LeakyReLu</td>
<td>256×64</td>
</tr>
<tr>
<td>Output</td>
<td>softmax</td>
<td>64×5</td>
</tr>
</tbody>
</table>
Evaluation Measures

- **Accuracy of Intrusion Detection**
  - By comparing the classification accuracy of the original testing set and the adversarial example set. The more the accuracy rate decreases, the higher the probability of adversarial example set successfully attacks the model.

- **Rank of the matrix**
  - By calculating the rank of the two matrixes, we can determine whether the two datasets are equivalent. If the two datasets are equivalent, we can assume that the feature relationship of original dataset have not been changed.

- **Euclidean Metric**
  - We calculate the Euclidean Metric between the two PCC matrixes of the original dataset and the adversarial dataset to compare whether the linear relationship of the two datasets is similar.

- **Feature Types Matching**
  - By checking the feature types in the adversarial example set, we can know whether they are consistent with the feature types of original examples.
Crafting Intrusion Adversarial Examples

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Parameters</th>
<th>Accuracy</th>
<th>Rank</th>
<th>Euclidean Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only traffic features</td>
<td>( \alpha = 1/255, T = 4 )</td>
<td>0.75</td>
<td>109</td>
<td>0.34</td>
</tr>
<tr>
<td></td>
<td>( \alpha = 2/255, T = 4 )</td>
<td>0.74</td>
<td>109</td>
<td>0.59</td>
</tr>
<tr>
<td></td>
<td>( \alpha = 2/255, T = 8 )</td>
<td>0.70</td>
<td>109</td>
<td>1.04</td>
</tr>
<tr>
<td>Only numerical features</td>
<td>( \alpha = 1/255, T = 4 )</td>
<td>0.59</td>
<td>111</td>
<td>5.15</td>
</tr>
<tr>
<td></td>
<td>( \alpha = 2/255, T = 4 )</td>
<td>0.42</td>
<td>112</td>
<td>7.14</td>
</tr>
<tr>
<td></td>
<td>( \alpha = 2/255, T = 8 )</td>
<td>0.33</td>
<td>113</td>
<td>9.02</td>
</tr>
<tr>
<td>All features</td>
<td>( \alpha = 1/255, T = 4 )</td>
<td>0.59</td>
<td>111</td>
<td>5.12</td>
</tr>
<tr>
<td></td>
<td>( \alpha = 2/255, T = 4 )</td>
<td>0.41</td>
<td>112</td>
<td>7.20</td>
</tr>
<tr>
<td></td>
<td>( \alpha = 2/255, T = 8 )</td>
<td>0.32</td>
<td>113</td>
<td>9.15</td>
</tr>
</tbody>
</table>

- Accuracy decreases when the noise value increases, but the relationship between features will change to a greater extent. The Accuracy decreases when the number of modified features increases, but the Rank will satisfy the original testing set.
- We chose ”Modifying all features, \( \alpha = 2/255, T = 4 \)” as the best parameter setting.
Compared with Baseline

The performance comparison with baseline (IFGSM) under parameters “Modifying all features, $\alpha = 2/255, T = 4$”

<table>
<thead>
<tr>
<th>Evaluation Measures</th>
<th>C-IFGSM</th>
<th>IFGSM</th>
<th>Original Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Accuracy</td>
<td>0.41</td>
<td>0.48</td>
<td>0.79</td>
</tr>
<tr>
<td>Feature Types Matching</td>
<td>1</td>
<td>0.69</td>
<td>1</td>
</tr>
<tr>
<td>Euclidean Metric</td>
<td>7.2</td>
<td>7.82</td>
<td>0</td>
</tr>
<tr>
<td>Rank of the Matrixes</td>
<td>112</td>
<td>111</td>
<td>112</td>
</tr>
</tbody>
</table>

- The adversarial examples generated by C-IFGSM makes the intrusion detection classifier get a lower accuracy, a higher feature types matching, a small Euclidean Metric, and a same rank of dataset.
- With the same parameter settings, C-IFGSM can better adapt to network traffic datasets and generate high-quality intrusion adversarial examples.
Verifying the Transferability

- We input the intrusion adversarial examples into other classification models and compared the accuracy decreases of other models to verify the transferability of the intrusion adversarial examples.
- C-IFGSM can make an apparent decline on the accuracy of decision tree, and an lesser impact on CNN and MLP models.
- the intrusion adversarial examples generated by C-IFGSM can also attack other models, but the adversarial effect is not very good in other neural network structure models.
Part Four

Conclusion
Conclusion

- We analyze different data characteristics between images and network traffic data.
- Based on the analysis, we implement a C-IFGSM based approach to generating intrusion adversarial examples with satisfying the multiple types and relationship of features.
- The experiments prove that C-IFGSM can effectively adapt to network traffic datasets and generate high-quality intrusion adversarial examples.

Future Work

- How to automate the mining of more complex mathematical constraints between features?
- How to propose a new defense method for the intrusion adversarial examples?
Thanks

Speaker: Yingdi Wang
Email: 17120482@bjtu.edu.cn