Verifying the Robustness of KNNs against Data-Poisoning Attacks

Yannan Li, Jingbo Wang, Chao Wang
10/19/2022
Limitation of Prior Works

• Verifying n-poisoning robustness of KNNs
  • Jia et al., Certified robustness of nearest neighbors against data poisoning attacks and backdoor attacks. AAAI 2022.
  • Only verifies part of problem (not handle complex “parameter tuning”)

• Verifying n-poisoning robustness of decision trees
  • Method works for decision trees only (but not for KNNs)

Our method is the only method for the entire KNN algorithm and is more accurate than [Jia et al.] for the prediction step
Outline

• Background
  • Data Poisoning Attacks
  • KNNs (k-nearest neighbors)

• Data Poisoning Robustness of KNNs

• Our Method

• Evaluation

• Conclusion
Outline

• Background
  • Data Poisoning Attacks
    • KNNs (k-nearest neighbors)
• Data Poisoning Robustness of KNNs
• Our Method
• Evaluation
• Conclusion
Background – *machine learning steps*

- Collect Data
- Train Model
- Deploy Model

*Chen et al. attacked VGG-Face in “Targeted back-door attacks on deep learning systems using data poisoning”, arXiv, 2017*

**< 0.2% poisoning***

**97% attack success rate***
Background – *mitigations*

- Collect Data
- Train Model
- Deploy Model
- Verify Model
Security Property – *n-poisoning robustness*

<table>
<thead>
<tr>
<th>Current Dataset</th>
<th>Training Dataset</th>
<th>Learned Model</th>
<th>Prediction Result of x</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td>M</td>
<td>(M(x) = \text{dog})</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(n = 3) Possible Clean Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD ABCD ABCD ABCD ABCD</td>
</tr>
<tr>
<td>ABCD ABCD ABCD ABCD ABCD</td>
</tr>
<tr>
<td>ABCD ABCD ABCD ABCD ABCD</td>
</tr>
<tr>
<td>ABCD ABCD ABCD ABCD ABCD</td>
</tr>
</tbody>
</table>

**Combinatorial explosion!**  
*Training size = 100 and n = 5, almost 8*\(\times\)*10\(^7\)* situations!* 

Secure Definition: \(\forall i, M_i(x) = M(x)\)
Outline

• Background
  • Data Poisoning Attacks
  • KNNs (k-nearest neighbors)

• Data Poisoning Robustness of KNNs

• Our Method

• Evaluation

• Conclusion
KNN \((k\text{-Nearest Neighbors})\)

Candidate \(K\) Values

Parameter Tuning

Opt \(K\)

Label Prediction

\[ K = 3 \]
\[ NN = \{square^2, \, star^1\} \]

\(Label = square\)
KNN parameter tuning: 4-fold cross validation

For one $K_i$

<table>
<thead>
<tr>
<th></th>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_3$</th>
<th>$G_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$err_{G_1}^{K_i}$</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>$err_{G_2}^{K_i}$</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>$err_{G_3}^{K_i}$</td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>$err_{G_4}^{K_i}$</td>
<td>❌</td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

$err_{K_i} = \frac{1}{4} \sum_{j=1}^{4} err_{G_j}^{K_i}$
KNN process

Training dataset $T$ → Parameter Tuning → OptK → Label Prediction → predicted label of $x$
Outline

• Background
  • Data Poisoning Attacks
  • KNNs (k-nearest neighbors)

• Data Poisoning Robustness of KNNs

• Our Method

• Evaluation

• Conclusion
Poisoning Impact

(1) *Direct influence*: change neighbors of test input $x$
   - Only need to check poisoning situations near $x$

(2) *Indirect influence*: change the optimal $K$
   - Need to check all the poisoning situations
Baseline method

$T_1'$ → Parameter Tuning → $K_1$ → Label Prediction → $label_1(x)$

$T_2'$ → Parameter Tuning → $K_2$ → Label Prediction → $label_2(x)$

$T_3'$ → Parameter Tuning → $K_3$ → Label Prediction → $label_3(x)$

... → Combinatorial explosion!

Same?
Outline

• Background
  • Data Poisoning Attacks
  • KNNs (k-nearest neighbors)

• Data Poisoning Robustness of KNNs

• Our Method

• Evaluation

• Conclusion
Our method

Training set $T$ and Max poisoning number $n$ → Overapprox Parameter Tuning → Opt KSet → Overapprox Label Prediction → Test input $x$ → Verified or not?
Our method

Training set $T$ and Max poisoning number $n$

Overapprox Parameter Tuning

$Opt\ KSet$

Overapprox Label Prediction

Test input $x$

Verified or not?
Our method – *Overapprox Parameter Tuning*

(Original) Parameter Tuning

Overapprox Parameter Tuning
Our method – *label changes via removal*

Remove 1 neighbors: Consider K+1 neighbors \ 1 points
Remove 2 neighbors: Consider K+2 neighbors \ 2 points
Remove 3 neighbors: Consider K+3 neighbors \ 3 points

…

Theorem: Just need to consider removing $\leq n$ points from $K+n$ nearest neighbors.
Our method – “Misclassified” becomes “Correctly Classified”

Intuition: Remove other labels

Current Label: Square (Misclassified)
New Label: Star (Correct)

$K=3$, $n=2$, and star being correct label.
Our method – “Correctly Classified” becomes “Misclassified”

Intuition: Remove correct labels

Current Label: Star (Correct)
New Label: Star (Correct)

$K=3$, $n=2$, and *star* being correct label.
Our method

Training set $T$ and Max poisoning number $n$

Overapprox Parameter Tuning

Opt $KSet$

Overapprox Label Prediction

Test input $x$

Verified or not?
Our method – *overapprox prediction*

• Input: Optimal KSet, test x, training T, poisoning n
• Output: label(x) remains the same?

For each K, when removing $\leq n$, same label(x)

For K in Opt KSet, same label(x)

Direct attack: change neighbors

Indirect attack: change opt K
Outline

• Background
  • Data Poisoning Attacks
  • KNNs (k-nearest neighbors)

• Data Poisoning Robustness of KNNs

• Our Method

• Evaluation

• Conclusion
Experimental Set Up

• Benchmarks
  • 2 small datasets
  • 4 larger datasets

• Research Questions
  • RQ1: **Accuracy** in proving n-poisoning robustness:
    • Compared to the baseline method (to obtain ground truth on small datasets)
    • Compared to the state of the art [Jia et al, AAAI 2022]
  • RQ2: **Efficiency** in handling realistic datasets:
    • Evaluated using the larger datasets
<table>
<thead>
<tr>
<th>Name</th>
<th>#Training</th>
<th>#Test ((x))</th>
<th>#Class ((output))</th>
<th>#Feature ((in))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>135</td>
<td>15</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Digits</td>
<td>1,617</td>
<td>180</td>
<td>10</td>
<td>64</td>
</tr>
<tr>
<td>HAR</td>
<td>9,784</td>
<td>515</td>
<td>6</td>
<td>561</td>
</tr>
<tr>
<td>Letter</td>
<td>18,999</td>
<td>1,000</td>
<td>10</td>
<td>36</td>
</tr>
<tr>
<td>MNIST</td>
<td>60,000</td>
<td>10,000</td>
<td>10</td>
<td>36</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>50,000</td>
<td>10,000</td>
<td>10</td>
<td>288</td>
</tr>
</tbody>
</table>
## Results – speed and accuracy on small datasets

<table>
<thead>
<tr>
<th>Max Poisoning</th>
<th>Baseline Time (s)</th>
<th>Our Time (s)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 1</td>
<td>60</td>
<td>1</td>
<td>93.3%</td>
</tr>
<tr>
<td>n = 2</td>
<td>4770</td>
<td>1</td>
<td>93.3%</td>
</tr>
<tr>
<td>n = 3</td>
<td>&gt;9999</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

**Iris** (#training=135, #test = 15, #class=3, #feature=4)

- Our method is several orders-of-magnitude faster than the baseline
- Accuracy > 93%

<table>
<thead>
<tr>
<th>Max Poisoning</th>
<th>Baseline Time (s)</th>
<th>Our Time (s)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 1</td>
<td>8032</td>
<td>1</td>
<td>96.1%</td>
</tr>
<tr>
<td>n = 2</td>
<td>&gt;9999</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

**Digits** (#training=1617, #test = 180, #class=10, #feature=64)
Result - speed and accuracy on large datasets

MNIST (time=16min, #training=60000, #test = 10000, #class=10, #feature=36)

CIFAR10 (time = 25min, #train=50000, #test = 10000, #class=10, #feature=288)

• Existing method* can only verify prediction phase
• Existing Method* can verify much less percentage

*Jia et al., Certified robustness of nearest neighbors against data poisoning attacks. AAAI 2022.
Conclusion

• The first method for soundly verifying n-poisoning robustness for the entire KNN algorithm
  • parameter tuning step + prediction step

• Demonstrated its accuracy and efficiency on popular supervised-learning datasets
  • small datasets + larger datasets