Poisoning Attack Defense and Gradient-Shaping Implementation in Python

Selma Emekci* (selmahaticeemekci@gmail.com), Eilan Tang* (eilantang@gmail.com), Shreya Kochar (shreya.kochar@columbia.edu), Aneesha Sreerama (sreerama.a@northeastern.edu), Andrew Wang (andrew.wang.2@stonybrook.edu), Anirudh Sreerama (anirudh.s@berkeley.edu)

Abstract

Machine learning models are susceptible to adversarial attacks that manipulate data to undermine model performance. While there is existing research on adversarial attacks against machine learning models, the vulnerability of linear regression models in particular and the development of effective defense strategies remain understudied. In this paper, we propose a defense approach that uses the FRIENDS mechanism. This defense methodology combines accuracy-friendly perturbation and random varying noise, utilizing a gradient-based approach to minimize prediction errors and simulating poisoning attacks by appending malicious data and fine-tuning the model. Our experimental results show how effective this defense method is, which was achieved by adjusting noise and clip threshold values. Our program offers an open-source and accessible solution for defending linear regression models against poisoning attacks, contributing to the security of AI systems.

Keywords: Adversarial Machine Learning, Python, Poisoning Attacks, Machine Learning

1. Introduction

Machine learning models, including linear regression, are vulnerable to various adversarial attacks. Linear regression is a widely used machine learning technique, however, it is susceptible to adversarial attacks. Specifically, poisoning attacks involve manipulating the data to add malicious data that can mislead the model during the learning process. These attacks aim to undermine the model's performance and make it generalize poorly on unseen data. Furthermore, gradient masking is a technique used by attackers to make their attacks harder to detect by altering the gradients of the poisoned data points. Existing research has focused on adversarial attacks primarily in classification tasks, such as deep neural networks. However, understanding the impact of poisoning attacks on linear regression models and finding effective defense mechanisms needs to be studied further. We seek to explore and develop potential defense strategies to protect linear regression models from poisoning attacks, where attackers deliberately inject malicious data into the training set to manipulate a model's behavior. Adversarial attacks can poison models, leading to harmful outcomes and these attacks pose a significant threat to the integrity of models. Finding effective defense strategies is crucial to building trustworthy ML systems. First, we plan to demonstrate the vulnerability of linear regression models to poisoning attacks and the potential consequences of these attacks. Second, we propose a comprehensive defense approach that effectively mitigates the impact of poisoning attacks while also maintaining model performance. Prior work has proposed a defense mechanism called FRIENDS to combat poisoning attacks. It utilizes two components to break poisons: accuracy-friendly perturbation and random varying noise. This combination has made this defense mechanism highly effective, leading us to implement it into our algorithm.

2. Literature Review

In a paper that proposed a defense strategy named "gradient shaping" which aims to constrain gradient magnitudes, Hong et al. (2020). employed differentially private stochastic gradient descent (DP-SGD) as a representative tool. The research presented in this paper focuses on two specific scenarios, highlighting the impact of poisoning attacks on model training and re-training. These scenarios represent common instances of indiscriminate and targeted poisoning attacks, respectively, each showing distinct aspects of the poisoning phenomenon. Although the paper's introduction of "gradient shaping" as a defense approach demonstrates innovation in countering poisoning attacks, we realize that it does not fully capture the complexity of real-world poisoning attacks, potentially limiting the generalizability of its findings. Further research could explore the integration of real-world factors, such as data quality and model complexity, to further evaluate defense mechanisms. We believe future research could also extend gradient shaping strategies to different machine learning models and tasks, enhancing our understanding of its applicability and limitations. A similar paper introduced a defense mechanism named FRIENDS that defends against various types of poisoning attacks on deep learning models. Liu et al. (2022) demonstrated that this defense mechanism has high effectiveness in breaking various invisible poisoning attacks with minimal impact on model performance. It utilizes two components to break poisons: accuracy-friendly perturbation and random varying noise. This combination has made this defense mechanism highly effective, leading us to implement it into our algorithm. We aim to demonstrate the effectiveness of FRIENDS as part of a

comprehensive defense approach that specifically protects linear regression models from poisoning attacks. In another paper about security attacks in machine learning systems, Xiong et al. (2021). emphasized the need for trustworthy ML system development. This resonates with our objective to propose a defense mechanism that enhances the robustness of linear regression models and maintains their performance in the presence of adversarial attacks. The paper also goes into detail about the importance of efficient ML defensive technologies. We align with this concern and will focus on the implementation and evaluation of defense mechanisms, aiming to mitigate the impact of poisoning attacks on linear regression models.

3. Methodology

The complete program can be found here[1]

3.1. Basic Overview of Program

The code proceeds to test the attack success rate over multiple trials (defined by num_trials). In each trial, a random house size and price are chosen as a malicious data point. The model is trained with the poisoned data and fine-tuned with additional noise. The predicted price for the malicious house size is then compared to the actual price, and if the difference exceeds a threshold (1 in this case), the attack is considered successful. The attack success rate is calculated as the percentage of successful attacks over the total number of trials.

The input features in our program are hard-coded in the X and y arrays, which represent house sizes and corresponding house prices (the target variable), respectively. The model is initialized with variables learning_rate (set to 0.1) and epochs (set to 10,000), which serve as hyperparameters for controlling the optimization process. The controls for the bias and weight of the linear regression model are theta_0 and theta_1, respectively. Perturbations and noise are generated using noise_std and clip_threshold. The noise_std variable controls the standard deviation of the noise added to the model's gradients during the poisoning attack while clip_threshold represents the threshold value used to clip the gradients during attack.

The very first function defined in the program is used to train the model. The train_with_friends_method() function implements the training process using a gradient descent algorithm. It accumulates the gradients for theta_0 and theta_1 over each data point in the training set and then updates the parameters using the gradients and the learning rate. Within each iteration, the function introduces some random noise to the gradients before updating the parameters.

To test the model's effectiveness, the perform_poisoning_attack() function is added to the program. It is used to perform a poisoning attack on the model by appending a malicious data point (house size, corresponding price) to the original data and then calling the training function to update the model with the poisoned data.

3.2. The Pseudocode

Initialize input data (house sizes) and target data (house prices) as arrays.

- X = [1, 2, 3, 4, 5]
- y = [100, 150, 200, 250, 300]

Set initial model parameters, learning rate, and training iterations.

- learning_rate = 0.1
- epochs = 10000
- theta_0 = 50
- theta_1 = 50

Function to train the model using gradient descent to minimize prediction errors.

- For each epoch:
  - Iterate through each value in NumPy array X:
  - Predict house prices
  - Calculate error
  - Update theta_0
  - Update theta_1

Train the model with original data using the function above

Function to simulate a poisoning attack by adding a malicious data point to the dataset and retraining the model.

- Append each data point to corresponding array
- Train model with poisoned array (call function above)

Calculate the predicted price for a new house size using the manipulated model.

- Substitute malicious data point into the array to test if outcome has changed

Evaluate the attack success rate by comparing predicted and actual prices in multiple trials.

- For 200 iterations:
  - Calculate random house price
  - Calculate random house size

- Train without poisoned data
- Train with poisoned data
- Print the difference between the two
- If the difference between the actual price and poisoned price is greater than 1:
  - Increase the attack success count by 1

Display the predicted price for the new house size and the attack success rate.

- Find the attack success rate percentage and print it

3.3. Testing

The full dataset is available here.[2]

3.4. Experiment 1

In the first experiment, the variable values were set as follows:

- noise_std = 8
- clip_threshold = 8.0

The attack success rate was 16.50% after running the program, which was unideal as target was less than or equal to 5%.

3.5. Experiment 2

In an attempt to lower the attack success rate, the standard deviation of the noise was increased as follows:

- noise_std = 10
- clip_threshold = 8.0

The attack success rate increased to 23.50%, showing that the standard deviation of the noise and the attack success rate are directly proportional to each other.

3.6. Experiment 3

In the third experiment, the noise standard deviation was lowered:

- noise_std = 5
- clip_threshold = 8.0

The attack success rate was lowered to 9.00%, it was much closer to the target rate so another variable was manipulated.

3.7. Experiment 4

The clip_threshold was lowered as shown in the fourth experiment:

- noise_std = 5
- clip_threshold = 4.0

The attack success rate wasn't that much different, at 8.00%

3.8. Experiment 5

In order to lower the attack success rate even more, the variable values were set:

- noise_std = 5
- clip_threshold = 2.0

With those values, the attack success rate was lowered to 3.00% and the target range was reached.

3.9. Experiment 6

In the sixth experiment, values were set as follows:

- noise_std = 5
- clip_threshold = 1.0

The attack success rate was at 1.00%.

4. Discussion

Our conclusion was that noise_std should be set to 5 and clip_threshold should be set to 1.0. The python program generates 200 poisoned values and calculates the difference between the actual price and the calculated price. With the values mentioned above, we were able to lower the attack success rate to 1.00%.

5. Summary and conclusions

Python has become a dominant programming language in AI due to its versatility and rich libraries. Adversarial ML is now a critical aspect of AI, addressing concerns about security and malicious attacks on models. Our open-source Python program specifically focused on defending against Poisoning attacks works to combat some of these challenges. By combining Gradient Shaping and FRIENDS algorithms, our program maximizes defense against malicious data insertion during model training, ensuring robust protection for linear regression based AI systems.

The program allows for accessibility and collaboration because it is open-source. Developers and researchers are welcome to use and enhance our code, contributing to its continual improvement. Our defense mechanism, built on Python 3.0, ensures an efficient and industry-standard implementation.

Looking ahead, we are optimistic about the future potential of this project. We envision expanding its application to diverse machine learning programs and integrating it with advanced malicious data detection algorithms. The journey to enhance AI security remains ongoing, and we are dedicated to advancing our program to fortify AI systems against emerging adversarial risks, protecting their integrity across various domains. As Python continues to drive AI innovations, our project serves as a promising milestone in secure AI models.

Acknowledgements

Thanks to Shreya Kochar, Andrew Wang, Aneesha Sreerama, and the Marvel Research Program Research Team.

References

Hong, S., Chandrasekaran, V., Kaya, Y., Dumitras, T., Papernot, N., 2020. On the effectiveness of mitigating data poisoning attacks with gradient shaping. CoRR abs/2002.11497. URL: https://arxiv.org/abs/2002.11497, arXiv:2002.11497.

Liu, T., Yang, Y., Mirzasoleiman, B., 2022. Friendly noise against adversarial noise: A powerful defense against data poisoning attacks. ArXiv abs/2208.10224. URL: https://api.semanticscholar.org/CorpusID:251719086.

Xiong, P., Buffett, S., Iqbal, S., Lamontagne, P., Mamun, M.S.I., Molyneaux, H., 2021. Towards a robust and trustworthy machine learning system development. CoRR abs/2101.03042. URL: https://arxiv.org/abs/2101.03042, arXiv:2101.03042.

Notes

1. GitHub Link: https://github.com/Selma-Emekci/Poisoning-Attack-Defense-in-Python

2. Full Dataset: https://docs.google.com/spreadsheets/d/1RO_0z6kaEtxs8JwxAhVCoyOqyUJrD6A0tlj5ehKqUK8/edit?usp=sharing