EPiC Series in Computing

Volume 69, 2020, Pages 40–47

Proceedings of 35th International Conference on Computers and Their Applications

Adversarial Machine Learning: Difficulties in Applying Machine Learning Existing Cybersecurity Systems

Nick Rahimi, Jordan Maynor, Bidyut Gupta

Southern Illinois University, Carbondale, IL 62901, USA

shrahimi@siu.edu, jmaynor@siu.edu, bidyut@cs.siu.edu

Abstract

Machine learning is an attractive tool to make use of in various areas of computer science. It allows us to take a hands-off approach in various situations where previously manual work was required. One such area machine learning has not yet been applied entirely successfully is cybersecurity. The issue here is that most classical machine learning models do not consider the possibility of an adversary purposely attempting to mislead the machine learning system. If the possibility that incoming data will be deliberately crafted to mislead and break the machine learning system, these systems are useless in a cybersecurity setting. Taking this into account may allow us to modify existing security systems and introduce the power of machine learning to them.

1. Introduction

Machine learning (ML) allows us to create models that will automatically recognize patterns, classify input data into several categories, and make predictions. In this investigation, we refer broadly to ML models. These could be ML techniques such as, Perceptrons, Support Vector Machine (SVM) [1], and many more [2]. In this research, we comment on ML algorithms as a whole, rather than referring to a specific one each time. ML is a powerful tool, which is why we can find it in nearly all aspects of life these days. It has, however, limited exposure to the area of cybersecurity for a variety of reasons in which we will explore. Adversarial machine learning, (AML), is a relatively new field of research in this area. The aim is to find safe ways to integrate machine learning to computer security without opening avenues of attack from malicious parties [3].

2. Learning From Data

To understand why research into AML is necessary, we need to comprehend why cybersecurity is fundamentally different than other areas where ML has been successfully utilized. Later we will discuss specific settings where ML is capable to be applied, but first we need a broader understanding of the problem. To be considered capable of "learning", an ML model needs to successfully perform two factors [4, 5]. We need to be able to minimize in-sample error (often seen shortened as Ein) and then if that can be done, then, we need to minimize out-of- sample error (Eout). Ein is the error we obtain from our ML model being applied to our training data. In ML, we typically separate data into two sets: the training set and the test set. Our training set/data is data that is already known. That is the stage where our ML models learn to perform classification or clustering. We can only advance to the next stage of learning if this can be accomplished accurately. The attained Ein should have the smallest possible error margin, given that we have full knowledge of the data coming in. Eout is where we actually start learning from the system. Eout comes from the error we get from a ML model being applied to new incoming data that we have no knowledge on, the testing data. With this data, we do not necessarily know what the outcome should be, we are just interested in what our system believes it is. If our system can get a low error at this point, then we can consider it capable of learning.

3. Data Distribution

One important note with this process is to understand the importance of data distribution as this is a main factor on how precise our ML can be applied. Although we may not know what exactly our test data will be, we expect it to have a similar statistical distribution as our training data. Within cybersecurity domain though that is not guaranteed. In fact, we should be assuming the opposite. In cybersecurity, we need to design our systems with the possibility that there is a malicious party, an adversary, trying to exploit the system. The same holds for ML models being used in cybersecurity. We cannot assume that any testing data coming in will be similar in distribution to the training data. There are a number of ways an adversary can deceive our ML models and perform all manner of mischief. This means that our testing data's distribution can change at any time, because there is an intelligence actively manipulating it, attempting to make an attack. And this brings us back to the definition of Adversarial Machine Learning (AML). Analyzing these types of attacks from an adversary on ML models will allow us to safely integrate ML into security systems without allowing the adversary the ability to get past our systems [6]. But the question is, why exactly do we want ML in our security systems?

4. An Arms Race

We find that in many security systems we are in a reactive arms race with the adversary. Essentially, we find ourselves designing our systems based on the previous discovered attack. This creates a big problem. Our systems are on trained to block the suspicions. To give an example which will be expanded later. Most Intrusion Detection Systems (specifically referring to Network Intrusion Detection Systems) like Snort or Suricata rely on catching adversarial activity to be trained against that activity [7, 8]. What happens if the activity is not caught though? In a worst-case scenario, the system could be trained to think the adversarial activity is legitimate. Even in a best-case scenario, where the IDS catches adversarial network activity, we need to manually train the IDS against that activity in the future. What we would like to do is automatically train our system against possible threats before they happen. That way, we are no longer being reactive, but proactive. That is the aim of AML, integrating ML to cybersecurity while allowing us to stay in a proactive cycle against adversarial threats rather than playing a never-ending game of catch-up with them [9]. Tables 1 and 2 are presenting the sequence of reactive and proactive measurements in arms race between an adversary and a classifier designer.

Sequence	Party involved	Activity
1	Adversary	Examine Classifier
2	Adversary	Create and launch attack
3	Classifier Designer	Examine attack’s outcome
4	Classifier Designer	Implement Countermeasures

Table 1: Conceptual sequence of the reactive arm race

Sequence	Party involved	Activity
1	Classifier Designer	Possible adversary scenarios
2	Classifier Designer	Practice attack
3	Classifier Designer	Examine attack’s outcome
4	Classifier Designer	Implement Countermeasures if the impact of attack is high

Table 2: Conceptual sequence of the proactive arms race

5. Spam Filtering

To give a quick and clear example of what a simple adversarial attack is let us discuss spam. One area of cybersecurity that may often go unnoticed ironically due to its universal presence is spam filtering. We are able to block spam in a few ways such as blacklists, or rule-based filters, but the one we want to focus on is content filters. Content filters typically work by analyzing what words are present in an email and how likely those words are to be found in a spam email. To greatly simplify, often these filters will measure the number of “bad words” to “good words” and judge whether an email is likely to be spam. An adversary can have their spam avoid detection by adding more “good words” to the email or purposely misspelling the “bad words” (Figure 1). Doing so causes the spam filter to more likely classify the spam into the legitimate category. This is also dangerous because if an adversary can successfully avoid a spam filter, they make the filter less able to determine what spam is and what is not. Classifying a spam email as legitimate will train the system to think other spam email may be legitimate. Given that on the internet there is no such thing as distance, security systems not only need to guard against an adversary, they also need to guard against all adversaries everywhere. It may seem obvious, but it’s very easy to get caught up in the idea that there is a single intelligent adversary we are fighting against. In reality, there is an arbitrarily large number of adversaries to fight against [10, 11].

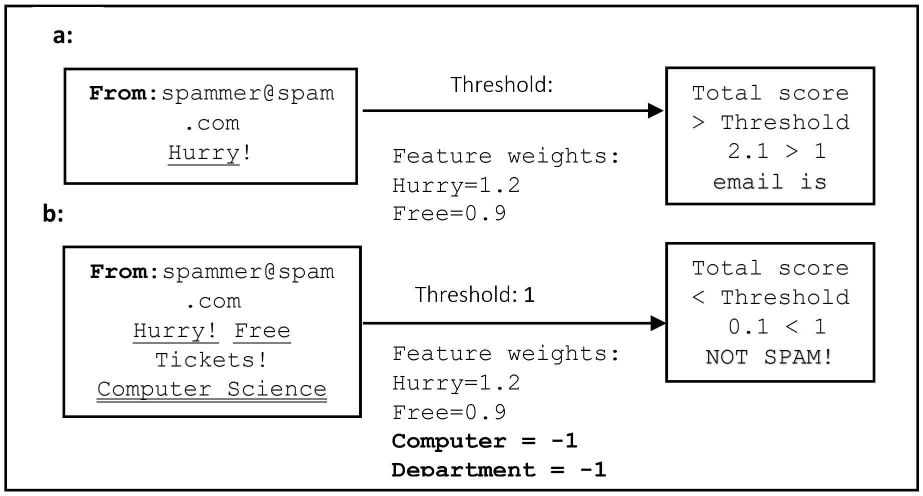

Figure 1: (a) Spam filtering demonstration – (b) Spammer adapts to the filter

6. Optical Character Recognition

Optical Character Recognition (OCR) is the process of converting written text into electronic text without a human in the process [12]. The technique is widely used in offices and academia as the need to convert paper documents to electronic is ever-present. Evaluating what character a written character is requires recognizing and matching patterns. So, ML models are used to perform this process. An attack on this model can be easily done by an adversary with interesting results. The images appear identical to a human observer but one has been specifically designed to fool a neural net classifier. But why might an attack on this sort of software be done? There might not be much advantage to attacking simple text OCR, but this sort of technology is the base for much more powerful software. Self-driving cars, for example, rely on input from a camera to determine what is in front of the car. Any sort of computer vision relies on the computer being able to see and classify what the objects are that it sees. If the classifiers that create computer vision are attacked, a great deal of damage could be done, certainly in the case of the self-driving car scenario. Imagine a situation where a car is unable to tell what a one way sign is (Figure 2). Accidents could easily occur. Any number of situations could be thought of in this area.

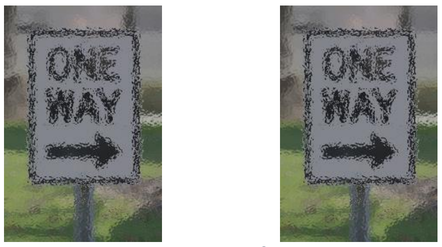

Figure 2: Image on the left was designed to fool an image classifier, note that to the human eye they are identical.

7. Envision Attacks And Intrusion Detection Systems

Adversarial attacks are not usually so clear in their intent, and certainly not so easily remedied. Given that we are discussing cybersecurity, networks are bound to come up. One of the most common attacks on a network is the evasion attack. They take many forms, but the point of the attack is simply finding a way to bypass a security system in place to deliver some sort of malware, spam, or exploit. We often use IDS software to detect any intrusion into our network and report that activity to an administrator. IDS systems use a few different methods to detect a possible intrusion. They are capable of pattern matching with known malware signatures protocol decoding, and detecting anomalies. Now obviously, a system that attempts to match an input to some pattern that is already known would seem well-suited to having ML incorporated. However, doing so opens us up to some nasty attacks from adversaries [9].

8. Poisoning Attacks

Using machine learning in this context presents an extremely dangerous opportunity to adversaries. To know what an attack is, an ML algorithm needs to be trained on normal network activity. Not only that, but this process is often repeated to take the ever-changing network structure into account. That means, whatever network activity looks like during this training or re-training period will be considered normal by the detection software in the future. This is when a poisoning attack can take place. If the adversary injects malicious activity during this period, they may be able to repeat the same attack later with the detection system only displaying a false negative. Essentially, a poisoning attack is a method of contaminating training data with malicious activity so that activity may easily be repeated in the future. In researchers were able to produce a large classification error in a support vector machine by injecting additional mislabeled points into an image. The attack demonstrates that with enough mislabeled points, the SVM was unable to properly classify further images introduced to it. One such solution to guarding against poisoning attacks specifically comes from. The idea is to use the process of bootstrap aggregating, “bagging” [10]. We can create a number of copies of our training data and use each on a different classifier. At the end we aggregate their predictions. From their results we can see that such a process does generally prove to be a viable solution to poisoning attacks.

9. Fighting The Adversary

It should be clear that it is not possible to simply plug ML models into cybersecurity systems in their classical forms. Doing so opens too many routes of attack for an adversary to take advantage of, which would make the system functionally useless. There are several methods we can employ to make these algorithms more intelligent, by anticipating an adversarial influence on our training data. Again, we want to create a scenario where, instead of reacting to an attack and developing a countermeasure, we want to create a proactive environment [13]. Methods we can use are:

- Build a More Robust Classifier
- Implement Several Classifiers
- Make Use of Random Data

This area is an active area of research so expect to see many possibilities being presented in the coming years. For the purpose of this paper, only these few general strategies to implement will be discussed.

10. A Robust Classifier

Creating a more robust classifier gets to the heart of the proactive approach that must be taken with AML. Instead of waiting for an attack to learn how to update our classifier, we will instead take a different approach. We can instead simulate an attack. At testing time, what we need to do is model possible attacks and see how they would affect our system. If there is an effect, if an attack can successfully fool our classifier, we take that into account and update our classifier to guard against it accordingly [14].
Also important is the process of identifying vulnerabilities that allow these attacks to happen. This should be self-evident, but fixing vulnerabilities in software is a constant battle that very much applies here.

10.1 Multiple Classifiers

As discussed earlier in the example of poisoning attacks, bagging is a potential solution here.

10.2 Random Data

The primary purpose of introducing random data to our classification system is to make it impossible for the adversary to completely understand our system. Obviously, when we talk about introducing random data to a ML classification system, we need to consider what the trade-offs are. If too many aspects of a classifier's decision process are random we can certainly expect there to be a drop in performance, compared to a classical ML model.

11. Clustering

An area of AML that will not be covered in-depth here is clustering and whether it can be safely applied to cybersecurity. We and other researchers are actively conducting research in this are; and still there are no definite answers. However, to speak on it briefly, clustering is the process of taking a number of ungrouped data points and finding groups they can belong to. Visually, each group is a set of data points that are generally close together (Figure 3). If an adversary has knowledge of the state of the clusters, it is relatively easy for them to create a new data point near one of the groups that the classifier might mistakenly classify as that group [6].

For the time being, it does not seem safe to incorporate clustering in an adversarial setting, but again, this is a current area of research that hopefully yield improvements to existing algorithms that make this doable.

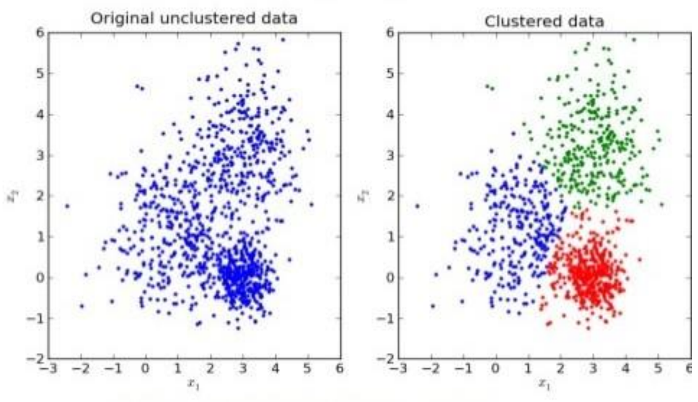

Figure 3: An example of K-Means clustering where the set of all data points have been divided into 3 distinct groups.

12. Conclusion

With the constant advances made in the world of machine learning, it is hard not to want to use it in any applicable computer science setting. A well-trained classifier can make short work of a task that would take an enormous amount of time if done by hand. Cybersecurity, however presents a uniquely dangerous setting for machine learning. Only in this area we do have to deal with a potentially malicious adversary. We must face the fact that some of the data we use to train our ML models is made to break or deceive those systems. For ML to be able to be applied safely in these adversarial settings, a new set of classifiers need to be created. That or we need to pre-emptively fool the adversary with random data and multiple classifiers. The important takeaway is that whatever we do, it must be proactive. Playing the never-ending game of catch-up with an adversary puts us at the disadvantage at every stage. Adversarial machine learning is an important emerging field, one that will no doubt bring a great deal of advancements to the general field of machine learning [15].

References

[1] Amendolia, S. R., Cossu, G., Ganadu, M. L., Golosio, B., Masala, G. L., & Mura, G. M. (2003). A comparative study of k-nearest neighbor, support vector machine and multi-layer perceptron for thalassemia screening. Chemo metrics and Intelligent Laboratory Systems, 69(1-2), 13-20.

[2] Rahimi, Nick, Jacob J. Reed, and Bidyut Gupta. "On the Significance of Cryptography as a Service." Journal of Information Security 9.04 (2018): 242.

[3] Lowd, D., & Meek, C. (n.d.). Adversarial Learning. KDD, 1-3.

[4] McDaniel, P., Papernot, N., & Celik, Z. B. (2016, May). Machine Learning in Adversarial Settings. Systems Security, pp. 68-72.

[5] Rahimi, N., Gupta, B., & Rahimi, S. Secured Data Lookup in LDE Based Low Diameter Structured P2P Network.

[6] Biggio, B., Pillai, I., Rota Bulò, S., Ariu, D., Pelillo, M., & Roli, F. (2013). Is Data

Clustering in Adversarial Settings Secure? *AISec'13: Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security* (pp. 87-98). Berlin: ACM.

[7] Eldow, O., Chauhan, P., Lalwani, P., & Potdar, M. (2016). Computer network security ids tools and techniques (snort/suricata). Int. J. Sci. Res. Publ, 6(1), 593.

[8] Rahimi, N., Sinha, K., Gupta, B., Rahimi, S., & Debnath, N. C. (2016, July). LDEPTH: A low diameter hierarchical p2p network architecture. In 2016 IEEE 14th International Conference on Industrial Informatics (INDIN) (pp. 832-837). IEEE.

[9] Biggio, B., & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition, 84, 317-331.

[10] Chen, T., & Ren, J. (2009). Bagging for Gaussian process regression. Neuro-computing, 72(7-9), 1605-1610.

[11] Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., & Tygar, J. D. (2011, October). Adversarial machine learning. In Proceedings of the 4th ACM workshop on Security and artificial intelligence (pp. 43-58). ACM.

[12] Rahimi, N., Nolen, J. and Gupta, B. (2019) Android Security and Its Rooting—A Possible Improvement of Its Security Architecture. Journal of Information Security, 10, 91-102.

[13] Liaw, A., & Wiener, M. (2002). Classification and regression by random Forest. R news, 2(3), 18-22.

[14] Biggio, B., Fumera, G., & Roli, F. (2008). Adversarial Pattern Classification Using Multiple Classifiers and Randomization.

[15] Praveen, M. V., Majumder, P., Sinha, K., Rahimi, N., & Gupta, B. A Highly Secured Three-Phase Symmetric Cipher Technique.