Analysis of Causative Attacks against SVMs Learning from Data Streams

Cody Burkard
University of Washington, Bothell
18115 Campus Way NE, Bothell, WA 98011
cburkard@uw.edu

Brent Lagesse
University of Washington, Bothell
18115 Campus Way NE, Bothell, WA 98011
lagesse@uw.edu

ABSTRACT

Machine learning algorithms have been proven to be vulnerable to a special type of attack in which an active adversary manipulates the training data of the algorithm in order to reach some desired goal. Although this type of attack has been proven in previous work, it has not been examined in the context of a data stream, and no work has been done to study a targeted version of the attack. Furthermore, current literature does not provide any metrics that allow a system to detect these attack while they are happening. In this work, we examine the targeted version of this attack on a Support Vector Machine(SVM) that is learning from a data stream, and examine the impact that this attack has on current metrics that are used to evaluate a models performance. We then propose a new metric for detecting these attacks, and compare its performance against current metrics.

1. INTRODUCTION

As storage costs decrease and the amount of data collected from our environment continues to increase, machine learning classification techniques play a crucial role in our ability to quickly and efficiently analyze the vast troves of data that we are now able to access. Critical problems have been tackled successfully via the use of these algorithms, including spam filtering, network anomaly [8] and malware detection [7], and even activity learning tasks in a pervasive setting[9, 14]. This trend will likely continue, while little work has been done to understand the security implications of these techniques.

An attacker has the potential to perform two types of attacks on a learning algorithm [2]; an *exploratory* attack in which the adversary attempts to discover more about the machine learning model and a *causative* attack that requires a more powerful attacker model where the adversary may manipulate training data. In the latter attack model an adversary has the potential to cause misclassification of future input data, which could be detrimental to the performance of a classifier or even dangerous if the application is used in critical systems such as smart grid load forecasting [1]. As systems such as these continue to be developed, it is important that we begin to understand how an adversary may manipulate the system and what metrics may be used to detect that this is occurring.

Anomaly detection systems such as clustering have been used in the past to detect noise in the training data that is not representative of the distribution of other data points in a batch. When these systems are used, an adversary is restricted in their addition of training data when attempting a causative attack. However, a *boiling frog attack* [8] has been proven in which anomaly detection does not detect that a sample is malicious because the attack occurs gradually over time. During a boiling frog attack, other approaches must be used to detect that an attack is happening.

In this work, we analyze Support Vector Machines(SVMs) as they are trained on stream data that can be modified by an adversary. We create an attack strategy for an attacker to perform a causative attack while under constraints imposed by an anomaly detection defense. This attack is validated on a system that uses SVMs for classification, and is studied through the behavior of learning curves of various metrics. Finally, we propose a new type of metric that is more useful in detecting the attack in this work, and evaluate it against other commonly used metrics.

2. BACKGROUND

In this section we provide context for the rest of this paper by discussing metrics for machine learning, *support vector machines*, and *adversarial machine learning* concepts. We begin with a formal description of SVMs. We then discuss commonly used metrics for evaluation of classification algorithms, and then segue into Section 3 with a description of adversarial machine learning concepts in SVMs.

2.1 Support Vector Machines

SVMs are a popular *supervised learning* technique used to separate data points into a set of known classes. This broader category of learners accepts a set of input sample points paired with their classes $\mathcal{D}_{tr} = \{(\boldsymbol{x_i}, y_i)|\boldsymbol{x_i} \in \mathcal{X}, y_i \in \mathcal{Y}\}_{i=1}^{n}$ as input, with the goal of generating an output hypothesis function f that correctly classifies future input samples x. The output of these learners for every new input sample x is a corresponding prediction of that sample's class, or *label.* . To separate these samples according to their respective classes a hyperplane is learned to separate each class from all other classes. This hyperplane is found in such a way that it optimally minimizes *classification error* of $\mathcal{D}_{tr}$

IWSPA '17 March 24, 2017, Scottsdale, Arizona, USA

DOI: http://dx.doi.org/10.1145/3041008.3041012

according to the true label values of each sample, while also maximizing the distance of each class from the hyperplane relative to the closest training samples to the hyperplane. A *structural risk minimization* process is employed to achieve these results, in which each sample point of $\mathcal{D}_{tr}$ that is close to the hyperplane contributes to the *empirical risk* of the hypothesis being tested, and the goal of the process is to find a set of these *support vectors* that define a hyperplane with the minimum possible risk. In this process, $L(y_i, f(x_i))$ is a loss function which defines the contribution to empirical risk of each data sample in D_{tr}. If new samples are added to the training set that reside near the learned hypothesis f_o, their loss increases, potentially causing the hyperplane to shift. This idea will be visited further in Section 6.

2.2 Metrics

The ability to evaluate the performance of a classifier as it is learning is essential when selection of a machine learning model is occurring. A variety of performance metrics are available for this purpose. In this work, we will focus on accuracy and empirical risk of an SVM model to interpret how well a model is learning. We also evaluate the predictive accuracy of the model using k-fold cross validation. In addition to these metrics and methods, we present another metric that may help to detect attacks, such as the attack we present in this paper.

2.3 Attacking SVMs

Support Vector Machines have been shown to be vulnerable to *causative attacks* [4, 15], in which the attacker is able to modify the learner based on the addition of new samples or the change of training sample labels. In an incrementally learning setting, these attacks may be performed if the attacker can access the data stream. In this case, every additional sample that is sent to the learner by the attacker may be used to update the trained model. Depending on the goal of the attacker, this change could be an *indiscriminate attack*, where the malicious samples have a general but dramatic effect on the model's decision boundary, or a *targeted attack*, in which the attacker is attempting to misclassify a specific part of the feature space, or even a specific sample. In this work, we focus on the impact and detection of a targeted attack on a batch learning classifier as its training data is update.

3. RELATED WORK

Since the inception of *adversarial machine learning*, a large amount of the work in this area has been focused on proving the vulnerability of machine learning algorithms against causative and exploratory attacks. In [3], a taxonomy is proposed for the evaluation of attacks against machine learning algorithm along with a comprehensive analysis of the possible attacks and defenses on machine learning algorithms. The *causative attack* described in this paper has been further proven in a variety of applications such as spam filtering, network anomaly detection, and malware detection[8, 13, 10, 7].

A *boiling frog* attacks is proven in [8], in which an anomalous network detector is evaded by performing a causative attack on the Principal Component Analysis(PCA) based detector. In [6], a causative attack is implemented to maximally decrease performance accuracy of a class by inputting an optimized mislabelled sample somewhere within that classes feature space. The attacked algorithm in this work is an incrementally learning SVM. This attack uses a gradient descent approach to optimizing the attack point. A label flipping attack is examined by Xiao et al.[15] in which an adversary is able to flip a set of arbitrary labels in a training data set to maximize the accuracy loss of the resulting model. This approach once again relies on a gradient descent approach to optimizing the attack point. Later in [4], a framework for analyzing causative attacks against SVMS is proposed.

Little work has been done to provide metrics specifically to understand how vulnerable a system is to machine learning attacks. In [11], classifier robustness is calculated according to the impact that an adversary may have on a learning algorithm.

4. SYSTEM DESCRIPTION

In this section, we define the learning system that we evaluate in this work. The setup of this system is described, followed by the goal of the learner and a naive defense mechanism that is in place to defend it from attacks.

4.1 System Setup

An SVM binary classifier is used to evaluate the work of this paper. The learner is constructed in a supervised manner based on input from a data stream. After receiving an initial training data set from the stream, the model is created based on this data and evaluated by a set of metrics to determine how well it is performing, including predictive accuracy, test accuracy, and empirical risk.

A batch of size b samples is drawn from the same probability distribution as the original training data and used to determine the accuracy of the model. This batch is drawn ten times, and the average of this accuracy score is used as the final accuracy metric. Predictive accuracy is estimated using k-fold cross validation with five folds. Finally, empirical risk is also measured.

To continue learning the probability distribution that generates the stream data, the system continues to collecting labelled training samples from the stream until the number of collected samples is equal to the batch size b. Once this batch is collected, the model is retrained on the combined set of $\mathcal{D}_{tr}$, the original training data, and $\mathcal{D}_{batch}$, the new batch. This combined set is then stored as $\mathcal{D}_{tr}$. Because the distribution of data is not changing, the models performance should slowly increase as it trains on more data, and learns more about the shape of the original distribution. Each time the model is retrained, the same metrics listed above are taken in order to form a learning curve for the model. The goal of this system is to learn the probability distribution that is generating the samples collected from its data stream.

4.1.1 Defense

In order for this system to learn the correct hypothesis, the data that it retrieves from the input stream must reflect the true probability distribution of the initial training data. If an attacker is able to submit mislabelled data to this stream, they may be able to perform a causative attack on the system. To stop this from happening, a simple defense mechanism is enforced by the learning system to help prevent attacks from occurring. If a new training sample

contributes over $\hat{r}$ in loss to the current hypothesis f_0, the sample is rejected as training data in the new batch.

5. ATTACKER MODEL

In this work, the attacker model is distinguished from other previous works in that the ultimate goal of the attack is to cause a single vector to be misclassified. In this section, we define our adversary in accordance with the framework proposed in [4]. The attacker's goal is identified, followed by a description of the attackers knowledge and capability. This information is then used to define an optimal attack strategy.

5.1 Attacker Goal

The goal of this attack is to cause a victim vector, X_{vict}, to be classified incorrectly. The true classification of this vector is the corresponding label of the victim class, y_{vict}, so a successful attack results in the classification of X_{vict} to become y_{att} according to the learned decision function $f_t(x_{vict})$.

In order to cause a vector to be misclassified, the hyperplane that separates the class y_{vict} from y_{att} must learn an incorrect concept such that $f_t(x_{vict}) = y_{att}$, where f_t is the target hypothesis. In a batch classification setting, this can be done by introducing specially crafted training examples $\mathcal{D}_{att}$ that increase the risk of the current hypothesis, causing it to move towards the target hypothesis.

The attackers goal is to find this set of attack samples $D_{att} = x_i...x_n$ that must be added to D_{tr} to cause the current hypothesis to change. Each of these samples must increase the risk associated with training the model on the current hypothesis in order to cause a change.

5.2 Attacker Knowledge

The attacker in this work has full knowledge of the targeted learner. Although we believe that this attack is possible on a black box by "stealing" a learning model from an adversary, we leave this approach to future work.

The knowledge of the attacker includes the original training data D_{tr}, the feature mapping of that data, the type of classifier used as well as the parameters used by that classifier, and any defense mechanisms that are employed by the learning system. This full knowledge allows the attacker to duplicate the model that is being attacked, and evaluate the attack on a local model before attacking a real model.

5.3 Attacker Constraints

The previously discussed defensive measure used to protect the learner we define in this work imposes one constraint on the attacker. The attacker may only send samples to the model that incur a loss that is less than the value of the loss anomaly detection defense, $\hat{r}$. This introduces an upward limit on the amount of risk that each new attack sample x^* may add to the current hypothesis. The constraint is described in the following equation.

$$l(y_i, f_0(x_i)) < \hat{r} \quad \text{for} \quad (x_i, y_i) \in \mathcal{D}_{att} \tag{1}$$

In our experiments, this value is set to 1, which is the worst case for the attacker because they may only introduce attack samples that are labelled correctly according to the current hypothesis.

5.4 Attacker Capability

Due to the existence of a local replica model, the attacker can intelligently craft samples that are not caught by the anomaly defense mechanism. In this work, we also model an adversary that has full control of the data stream, and can introduce any arbitrary set of samples into a training batch. Although this is a very powerful adversary, we argue that this is likely of a system that generates the data stream is infected with malware.

5.5 Attacker Strategy

Given the attacker model described in this section, we create an optimized attacker strategy that is simple and effective for any classification model. Instead of attempting to directly calculate f_t to find $\mathcal{D}_{att}$, we severely limit the potential attack samples and approximate a $\mathcal{D}_{att}$ based on this limited selection. This approach is based on and requires knowledge of the current hypothesis.

The attacker first determines a targeted vector with the goal of misclassifying it. This vector does not have to be part of D_{tr}, and can be constructed of any arbitrary values. After determining the class of this vector based on the trained hypothesis f_o, the attacker finds the closest samples in D_{tr} of the opposite class, and uses that sample as the initial attack point. Once these samples are found, the attacker constructs a first degree polynomial with them, which contains all possible attack points. The attacker then iterates through the procedure of generating attack samples, until the targeted vector is misclassified on the local model. In this procedure, the attacker conducts a binary search on this polynomial to find a sample that contributes the maximum amount of loss within the upward loss constraint of each attack sample. The result of this search is used to update the local algorithm, and is also sent to the model that is being attacked.

In our experiments the constraint on loss for each sample is set to 1. This simplifies the algorithm used slightly because the attacker only has to search for the closest sample to the hyperplane, which is the location of maximum loss contribution. This modified algorithm is described in algorithm 1.

6. EVALUATION

In the following sections, we explain the experiments conducted in this paper as well as our evaluation mechanisms. We provide a novel metric for detecting targeted attacks and describe how it works. The experiments conducted are then described, followed by the results of those experiments.

6.1 Stability

When analyzing the security of a machine learning model, it is helpful to have a set of metrics that can be used to understand its state. However, different attacks have varying impacts on the learning curves of traditional metrics such as accuracy, predictive accuracy with k-fold cross validation, and empirical risk. These metrics may be quite helpful in detecting an indiscriminate causative attack because that type of attack leads to a massive reduction in classification performance for the model, which is what they measure. On the other hand, in the case of the targeted attack described in algorithm 1, these metrics have three problems. First, a targeted attack may not impact the entire feature space

Algorithm 1: Binary Search Targeted Attack on Classifier Model

input : D_{tr}, x_{vict}, batch size, model, step size
output: $\mathcal{Z}$, a set of attack points
x_{att} = locate closest sample to
x_{vict} in attacking class from D_{tr}
y^* = model.classify(x_{att})
set t to a line constructed between x_{vict} and x_{att}
$x^* = x_{att}$
$v^* = x_{vict}$
repeat
  $x1$ = x value of x_{att}
  $x2$ = x value of x_{vict}
  diff = $x2 - x1$
  repeat
    stepsize = $\frac{diff}{2}$
    xTest = $x1$ + stepsize
    testLabel = model.classify(t(xTest))
    if *testLabel == victimLabel* **then**
      $v^* = t(xTest)$
    end
    else
      $x^* = t(xTest)$
    end
    diff = $x2 - x1$
  until *diff < stepsize*;
  $\mathcal{Z}+ = \{x^*, y^*\}$
  if *length($\mathcal{Z}$ == batchsize)* **then**
    update local model on $\mathcal{Z}$
  end
  send $\{x^*, y^*\}$ to model under attack
until *model.classify(x_{vict}) is not y^**;

in the same way that an indiscriminate attack does, so the learning curve should not deteriorate as quickly, making it more difficult to detect an attack. Second, if the ratio of attack data to benign data is small enough, these metrics may not even suffer during a successful attack, because they are based on average scores across many samples. Finally, even if the learning curves are impacted enough to detect an attack after each retrain, an analyst has no way of knowing what portion of the feature space is under attack based on these metrics as they are collected over time.

In order to detect a targeted attack on a batch learning classifier as described in 4, the metrics that are calculated each time the learner is retrained on new data must include not only the amount of risk that is associated with each training sample, but also the amount of risk posed to a specific portion of the learner's feature space. In this system, all previous training data is kept during each successive retraining of the model. We propose a stability metric that takes advantage of this property of the learning system to provide more information about the state of a specific portion of the learner's feature space.

Due to the nature of the attack described in section 5, the portion of the feature space between the original decision boundary f_o and the victim sample x_vict is misclassified by the model after a successful attack. This causes the training data of the victim class that falls within this portion of the feature space to add a significant amount of risk to the hypothesis learned after the attack. This could be detected by monitoring the distance to the hyperplane of each sample in this portion of the feature space after each training cycle.

When there are more benign training samples in one location, they will provide more stability to that locality because of the amount of potential loss that they will contribute as the hypothesis approaches them. The stability metric proposed in this work is based on this concept as well as the idea of monitoring the distance of these samples to the hyperplane to observe changes in the feature space. To calculate this metric for a specific location, a vector of n dimensions x_s is chosen as the central location to be monitored. A search is done on all of the training data in the victim class to find all samples within a chosen radius r, described as a set $\mathcal{S}_{rad} = x \in \mathcal{D}_{tr} | d(x_s, x) < r$ where $d(x_1, x_2)$ is the euclidean distance between two vectors. The distance to the hyperplane of these vectors is then found with the decision function $df(x)$ of the classifier, and these values are then summed together. Finally, this value is divided by the batch size of the learning system in order to normalize this metric. The formal description of this metric is below.

$$stability(x_i) = \frac{\sum_{k \in \mathcal{S}_{rad}} df(k)}{b} \quad (2)$$

6.2 Experimental Setup

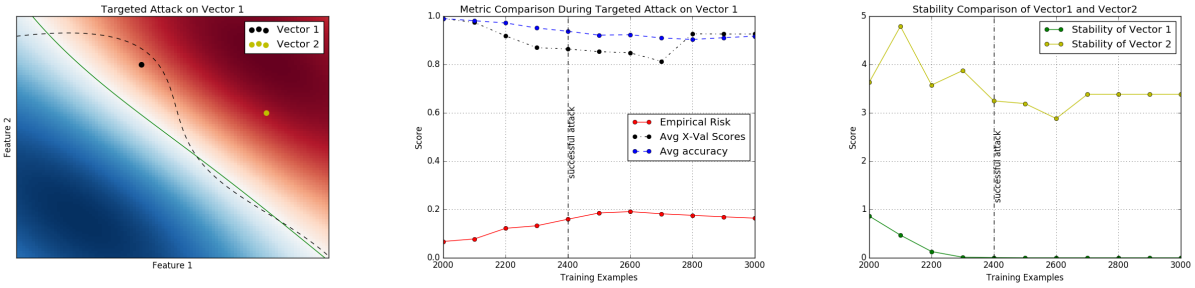
The goal of the experiments described in this section is to evaluate the feasibility of the attack in section 5 on the system in 4, as well as show the use of new metrics such as stability in detecting these attacks. The experiments conducted in this work are described in this section along with the parameters used to conduct those experiments.

The first set of experiments is meant to show the impact of our targeted attack. The system under attack is initially trained on 2000 training samples from the input stream and is retrained on every 100 new samples. The evaluation data contains 10000 samples drawn from the real data stream, split up into ten evaluation data sets that are used to measure accuracy of the learner on the original probability distribution, and averaged together to get the final result. The defense mechanism described in 4.1.1 has an $\hat{r}$ value of 1. The attacker is introduced to this system with full control of new data samples with the goal of misclassifying a victim sample, x_{vict} as quickly as possible. Once this sample is misclassified, the attack continues to input new samples at the target coordinates.

The second set of experiments conducted are meant to evaluate the ability of the stability metric to detect a targeted attack on a victim point. The same attack and system is used for these experiments, but stability is introduced on two separate vectors in the victim class. The radius used for these metrics is equal to the distance of the selected vector from the hyperplane. The attack is run twice on the same set of training data, the first time targeting one vector while the second time targeting another. These vectors are selected for the stability measurement, and show how a higher stability makes it more difficult for an attacker to succeed. Each time a model is retrained in these experiments, all of the metrics described in this work are collected and plotted on a learning curve for the model.

These tests are run using the libraries provided by scikit-learn[12], an open source machine learning tool kit written in python. The algorithm used for each of these experiments

Figure 1: Plots representing targeted attack on vector 1.



(a) Attack on Vector 1 (b) Metrics during attack on Vector 1 (c) Stability during attack on Vector 1

is an SVM with a gaussian kernel, with a penalty parameter value of 1.0 and a gamma value of .001.

6.3 Results

The results of the experiments described above are depicted in the graphs in figures 1, and 2. The first graph in each of these sets is a visualization of the attack taking place in a two dimensional feature space. The dashed line represents the resulting model of the attack, while the solid line represents the models learned from the original training data. These graphs show the ability of the simple targeted attack described in section 5 to cause misclassification of samples deep within a class's feature space, without impacting the entire feature space.

The second graph in each set shows the comparison of empirical risk, cross validation scores, and average accuracy each time the model is trained. The number of samples in each training batch is shown as well as the instance that the attack was successful. We notice two things about this graph. First, although the average accuracy and cross validation scores decrease at the beginning of the attack and slightly after the success of the attack, the accuracy scores do not change after the attack is successful because the decision boundary no longer moves, and the cross validation score actually increases again in the attack on vector 1. Secondly, the empirical risk also begins to lower again after the attack occurs, because as more data is submitted near the targeted vector the decision boundary continues to move away and the average loss contributed by this set of malicious vectors decreases.

The third graphs in each set depict the stability metric of both targeted vectors as they are attacked. The comparison of these graphs provides a great deal of information about the model as it is under attack. First, we notice that the variance in the stability value of these vectors as the attack progresses is a great deal more than the variance of the other metrics observed. Second, the stability metrics of each vector change independently of each other, because only one of these locations in the feature space is under attack. In figure 2(a), we notice that the stability of both vectors decreases slightly, but this is because the decision boundary really is moving closer to both vectors, as seen in the first graph. Finally, we notice that with a higher stability value, it takes more samples to succeed in a targeted attack on the location being measured. In figure 2(a), vector 1 starts with a stability value under 1.0 and is compromised by the fourth batch of malicious data. On the other hand, figure 3(a) shows that vector 2 has a stability score of close to 4.0 to start, and it takes seven batches to compromise this target, almost twice that of vector 1.

7. DISCUSSION

Our results illustrate some of the security challenges of systems that utilize machine learning algorithms. In particular, any time an attacker is able to manipulate or add training data to a machine learning model, they are able to exert some amount of control over the classification of future samples. Previous works have explored how this is possible on a static set of training data [15], or using an indiscriminate attack [5], but to our knowledge this is the first targeted causative attack shown on an SVM.

Targeted causative attack have only begun to be explored, but we believe this is a very probable type of attack in the future as we rely more on machine learning algorithms to make decisions based on the data from their environment. Systems that learn from crowd sensing data may be particularly vulnerable to this type of attack because much of the data that is used will be based on the input of untrusted devices. These systems may also have to account for a changing environment and adjust to *concept drift* using the data collected from these untrusted devices, which may be manipulated by an attacker or even infected with malware. As these systems continue to be developed, it is imperative that we develop new ways to measure changes in these systems as well as create defense mechanisms to protect them other than basic anomaly detection.

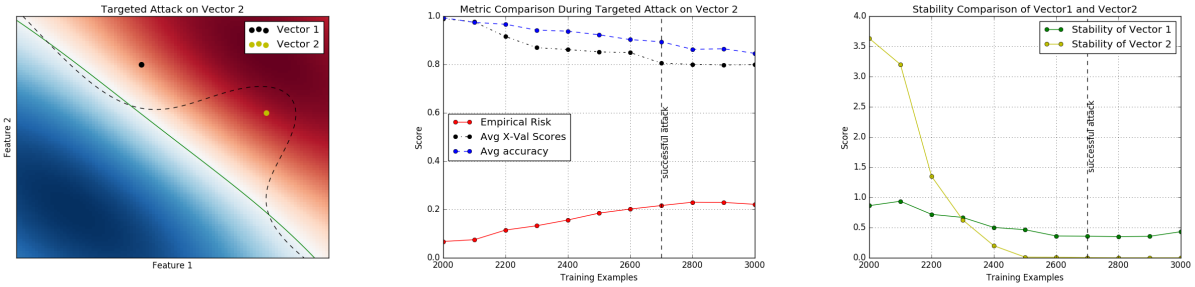
The stability metric from this work is shown to change as a targeted attack occurs. Further work using this metric may include methods for discovering what the most important locations in feature space are for the classes within that space, so that a system may more intelligently choose what locations to monitor using this metric. We also argue that this metric may be used to create a game-theoretic defense mechanism to prevent malicious samples from having a large impact on a decision boundary.

Finally, this is just one example of a metric that may be used to detect targeted causative attacks. This metric is only possible in situations in which previous training data continues to be used to update a model. In an environment with changing concepts or in an online learning setting, this metric cannot be used because this training data is changing, and other metrics may be needed to detect this attack.

8. CONCLUSION

As machine learning algorithms become more prevalent in the technologies used in our daily lives, it is of the utmost

Figure 2: Plots representing targeted attack on vector 2.



(a) Attack on Vector 2 (b) Metrics during attack on Vector 2 (c) Stability during attack on Vector 2

importance that we understand what attacks are possible against these algorithms and how different learning models will react to those attacks. Furthermore, this analysis must be based on various metrics that allow us to gain a deeper insight into the behavior of these algorithms in an adversarial setting. Understanding these fundamental concepts increases our situational awareness in a security context, and is the stepping stone to more elaborate and complete defense techniques that prevent these attacks from occurring.

In this work, we have generated a new targeted attack algorithm according to the framework presented in [4] and empirically evaluated it against a learning system with an anomaly detection defense. We show how this defense is not able to stop a causative attack from occurring on this system. Furthermore, we compare the learning curves of multiple metrics during this attack to prove their lack of ability to detect this attack, and provide a new novel metric for identifying how stable a section of a model's feature space is based on the current training data. Finally, we have discussed how this metric may be used to detect an attack.

We believe that this work provides a better understanding of the threat posed to SVMs in an incremental learning setting, and could potentially lead to further research into generic defense techniques for the attack discussed in this paper. By analyzing metrics that monitor samples within a critical location a models feature space, we show how one might detect a targeted attack as it is occurring. We argue that knowledge gained from this work could be used to model the characteristics of an attack, leading to better detection techniques and countermeasures.

Acknowledgements

The work presented in this paper was partially supported through CAE Cybersecurity Grant H98230-15-1-0284.

9. REFERENCES

[1] Z. Aung, M. Toukhy, J. Williams, A. Sanchez, and S. Herrero. Towards accurate electricity load forecasting in smart grids. In *The Fourth International Conference on Advances in Databases, Knowledge, and Data Applications, DBKDA*, 2012.

[2] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[3] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, pages 16–25, New York, NY, USA, 2006. ACM.

[4] B. Biggio, I. Corona, B. Nelson, B. I. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, and F. Roli. Security evaluation of support vector machines in adversarial environments. In *Support Vector Machines Applications*, pages 105–153. Springer, 2014.

[5] B. Biggio, L. Didaci, G. Fumera, and F. Roli. Poisoning attacks to compromise face templates. In *2013 International Conference on Biometrics (ICB)*, pages 1–7, June 2013.

[6] B. Biggio and P. Laskov. Poisoning attacks against Support Vector Machines. In *In International Conference on Machine Learning (ICML*, 2012.

[7] B. Biggio, K. Rieck, D. Ariu, C. Wressnegger, I. Corona, G. Giacinto, and F. Roli. Poisoning Behavioral Malware Clustering. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, AISec '14, pages 27–36, New York, NY, USA, 2014. ACM.

[8] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, pages 43–58, New York, NY, USA, 2011. ACM.

[9] B. Lagesse, C. Burkard, and J. Perez. Securing pervasive systems against adversarial machine learning. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–4, March 2016.

[10] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Misleading Learners: Co-opting Your Spam Filter. In *Machine Learning in Cyber Trust*, pages 17–51. Springer US, 2009. DOI: 10.1007/978-0-387-88735-7_2.

[11] B. Nelson, B. Biggio, and P. Laskov. Understanding the risk factors of learning in adversarial environments. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 87–92. ACM, 2011.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar. Antidote: Understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 1–14, New York, NY, USA, 2009. ACM.

[14] J. Wen, J. Indulska, and M. Zhong. Adaptive activity learning with dynamically available context. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–11, March 2016.

[15] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53–62, 2014.