How to Steal a Machine Learning Classifier with Deep Learning

Yi Shi, Yalin Sagduyu, and Alexander Grushin

Intelligent Automation, Inc.,
Rockville, MD 20855, USA
Email:{yshi, ysagduyu, agrushin}@i-a-i.com

***Abstract*—This paper presents an exploratory machine learning attack based on deep learning to infer the functionality of an arbitrary classifier by polling it as a black box, and using returned labels to build a functionally equivalent machine. Typically, it is costly and time consuming to build a classifier, because this requires collecting training data (e.g., through crowdsourcing), selecting a suitable machine learning algorithm (through extensive tests and using domain-specific knowledge), and optimizing the underlying hyperparameters (applying a good understanding of the classifier's structure). In addition, all this information is typically proprietary and should be protected. With the proposed black-box attack approach, an adversary can use deep learning to reliably infer the necessary information by using labels previously obtained from the classifier under attack, and build a functionally equivalent machine learning classifier without knowing the type, structure or underlying parameters of the original classifier. Results for a text classification application demonstrate that deep learning can infer Naive Bayes and SVM classifiers with high accuracy and steal their functionalities. This new attack paradigm with deep learning introduces additional security challenges for online machine learning algorithms and raises the need for novel mitigation strategies to counteract the high fidelity inference capability of deep learning.**

***Index Terms*—Machine learning; adversarial machine learning; classifier; deep learning; exploratory attacks.**

I. Introduction

In the present day, information systems are subject to frequent exploits and attacks, and it is not possible to assume that any given program will be used as intended. This holds true not only for traditional programs where all behaviors are programmed manually, but also for *machine learning* algorithms, which are becoming increasingly prevalent in current information systems. The emerging field of *adversarial machine learning* studies how effective learning can take place under the presence of an adversary [1]–[4]. However, the security limitations of such learning are not yet well understood, and deserve further study, in order to characterize and improve the overall security of information systems.

If the effects of adversarial machine learning are identified and appropriate mitigation techniques are developed, then the applications can be diverse and far-reaching. Machine learning algorithms are presently utilized by different applications including (but not limited to): search engines, social media/network platforms, intelligence analysis applications, intrusion detection, bot detection, recommender systems, online review systems, online matchmaking systems, spam email filtering, document classification, Internet of Things, cyber-physical systems, and unmanned vehicle controllers.

A machine learning algorithm or an information system built upon one can be "attacked" in several ways [1]–[4] (illustrated in Fig. 1):

1) *Causative* attacks (also known as poisoning attacks), e.g., [5], [6], attempt to provide incorrect/misrepresentative training data to the algorithm, such that it does not learn the intended function.
   - In supervised learning, this may be achieved by mislabeling the training data, e.g., in adversarial active learning [2], where the algorithm requests training labels from potentially untrusted sources.
   - In reinforcement learning, a causative attack can take place by providing incorrect rewards or punishments.
   - In unsupervised learning, the attack can be launched by sampling training examples in a biased way, such that they do not reflect the underlying statistical distribution.
2) *Exploratory* attacks, e.g., [7]–[9], occur after the algorithm has been trained, and attempt to uncover information about its inner workings, in order to identify weaknesses of the algorithm. This attack may attempt to extract
   - the decision boundary used by the algorithm (e.g., hyperplanes of the Support Vector Machine (SVM) algorithm),
   - a general set of rules that the algorithm follows,
   - a set of logical or probabilistic properties about the algorithm, or
   - information about the data that was used (or not used) to train the algorithm.
3) *Evasion* attacks, e.g., [10]–[12], are also launched upon trained algorithms, and involve providing the algorithm with input (test) data that will result in an incorrect output.
   - A classic example of an evasion attack is the generation of spam emails that will fool a trained filter into accepting them as legitimate.
   - Another example is the creation of social bots (e.g., automated tweet generators) that will fool bot

978-1-5090-6356-7/17/$31.00 ©2017 IEEE

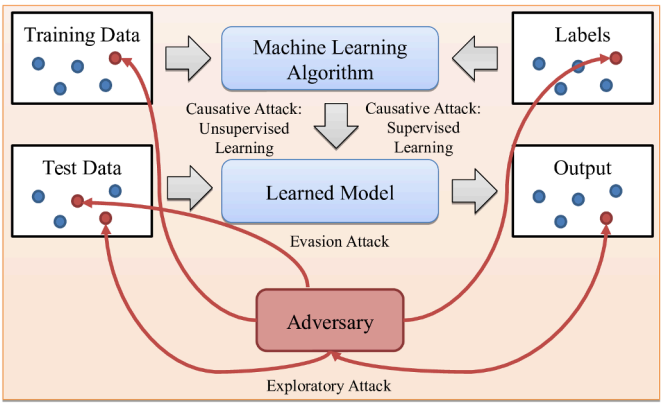

Fig. 1. Adversarial machine learning.

detectors.

- Evasion attacks can be made more effective by exploiting vulnerabilities identified via exploratory attacks, or created via causative attacks.

Clear understanding of potential attacks against machine learning systems is of paramount importance. In online systems, the user can poll the machine learning system with the input data (e.g., through an application programming interface (API) in a web-based system) and observe its output (e.g., classifications). Such systems are open to *evasion*, *exploratory*, and in some cases *causative* adversarial machine learning attacks. In this paper, we address the problem of exploratory attacks where the state-of-the-art focuses on *model inversion attacks* [7]–[9]. In such attacks, the adversary typically knows the type and structure of the classifier, and attempts to learn the model parameters. In this paper, we relax this assumption for the adversary and let the adversary launch an attack for building *a functionally equivalent classifier without knowing the type, structure and parameters of the classifier*.

An adversary launches a three-step black-box attack by 1) polling the classifier with input data, 2) observing labels returned by the classifier, and 3) using the input data and the labels to train a *deep learning classifier*. The output is a deep learning classifier that is functionally equivalent to (i.e., produces high fidelity labels similar to) the classifier under attack. The functionally equivalent machine built by the adversary implicitly infers training data, type of the classifier, and hyperparameters of the classifier. For testing, the same input data is given to the classifier under attack and the classifier that is built by the adversary. The success of the adversary is measured by similarity of labels returned by the classifier under attack and the classifier that is built by the adversary.

We consider the application of text classification and use Naive Bayes and SVM as the classifiers under attack. Results demonstrate that:

- Deep learning can reliably build a functionally equivalent classifier for Naive Bayes (2.10% error) and SVM (2.56% error).
- Simple classifiers such as Naive Bayes and SVM cannot effectively build a functionally equivalent classifier for deep learning (error is 11.18% for Naive Bayes and 8.61% for SVM).

Our results have significant security implications. First, classifier theft poses a risk to the intellectual property of private companies and other organizations, which have invested a significant amount of time and effort to gather training data for a classifier, to train the classifier, and to tune its hyperparameters. If the classifier is stolen, then an organization's competitive advantage is reduced; in the long term, the possibility of classifier theft can even discourage organizations from developing novel machine learning systems. Additionally, once a classifier is stolen, the adversary is free to analyze it (with an unlimited number of queries), in order to identify its potential weaknesses, as well as its underlying functionality. This can allow the adversary to subsequently launch an evasion attack against the original classifier, or to infer private information about the organization that developed it (such as its priorities and resources) or the underlying training data. For example, if the original classifier is trained on social media data, then social media users' privacy can potentially be compromised through an analysis of the functionally equivalent classifier, which may necessitate the use of privacy-preserving access to users' social relationships [13].

The resulting security threat can exist not only for individual organizations, but also at the national level. For example, suppose that a government organization trains a classifier using sensitive data that is not available to other governments and private organizations. Despite the fact that no one else has access to this data, an adversary (who may belong to an internal or foreign hostile group) may still be able to replicate the functionality of the classifier. Subsequently, the adversary may attempt to use the stolen classifier for malicious purposes, subvert the original classifier through an evasion attack, or infer information about the data that was used to train it.

The rest of the paper is organized as follows.

- Section II presents the system model for launching an attack to steal a machine learning classifier.
- Section III introduces the classifiers under attack and the classifier used by the adversary.
- Section IV presents the approach to train classifiers for a text classification application.
- Section V presents the approach to train the adversarial classifier with deep learning.
- Section VI concludes the paper.

II. How to Launch an Attack to Steal A Machine Learning Classifier

Suppose a classier has been already trained by selecting training data, type of the classifier, and hyperparameters of the classifier. This classifier is made for online access as a black-box system without revealing any information other than the input and output relationships. Each user can poll this classifier by providing test data as the input and observing the labels as the output of classification.

978-1-5090-6356-7/17/$31.00 ©2017 IEEE

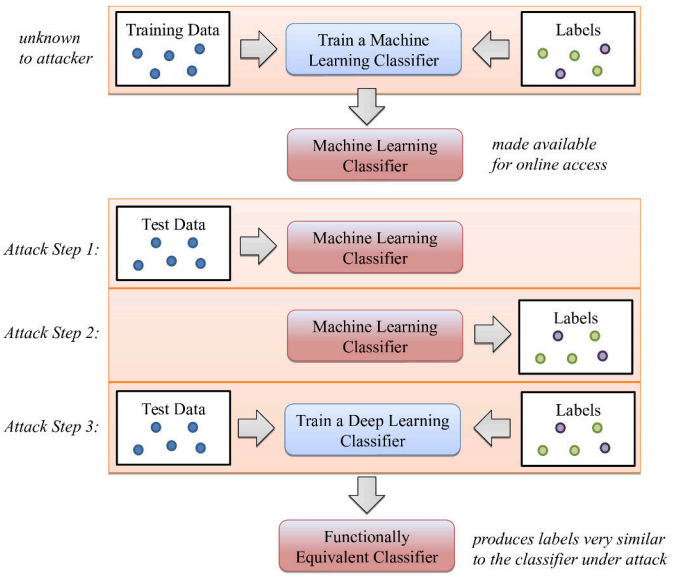

Fig. 2. Steps to steal an online machine learning classifier.

As shown in Fig. 2, the adversary launches a three-step attack to steal the classifier:

1) The adversary polls the classifier with input data.
2) The adversary observes labels returned by the classifier.
3) The adversary uses the input data and the labels to train a deep learning classifier and optimize its hyper-parameters.

The equivalent machine implicitly infers all aspects of the classifier as a black-box system including

- training data,
- type of the classifier, and
- hyperparameters of the classifier.

III. Machine Learning Classifiers

Different methods can be used for classification purposes such as Naive Bayes, Support Vector Machine (SVM) and Artificial Neural Network (ANN) to extract the structure of the training data, build a model and classify the test data.

- Naive Bayes is a probabilistic classifier based on applying Bayes' theorem with (naive) independence assumptions between the features. Naive Bayes builds conditional probabilities of features (represented as vectors of feature values) under each class and determines the mostly likely class for the features of test data.
- SVM is a non-probabilistic linear classifier that represents the features of training data as points in space and maps them so that the features of the separate classes are divided by a gap that is as wide as possible. SVM maps features of test data into the same space and predicts them to belong to a class based on which side of the gap they fall on.
- An ANN is a machine for computing some function and consists of simple elements called *neurons* that are joined by weighted connections, a.k.a. *synapses*. A neuron j performs a basic computation over its input synapses w_{ji} (connecting neurons i and j), and outputs a single scalar value y_j, which can be interpreted as its activation, or firing rate. For example, the following computation is commonly used: $y_j = f\left(b_j + \sum_{i \neq j} w_{ji} y_i\right)$. Here, b_j is a constant bias term, while $f(x)$ is some non-linear function, such as $f(x) = 1/(1 + e^{-x})$. In a *feedforward neural network* (FNN) architecture (topology), neurons are arranged in layers. The activations of neurons in the input layer are set externally, while the activations of the hidden layer neurons and output layer neurons are computed as specified above; the latter represent the result of the network's computation.

Provided that the hidden layer in ANNs has a sufficiently large number of neurons, the aforementioned simple architecture can (in theory) approximate any mathematical function to an arbitrary degree of precision, via its synaptic weights [14]. However, for many real-world problems, a good approximation would require a prohibitively large number of hidden layer neurons. To overcome this limitation, over the past decade, researchers have focused on deep architectures that contain many hidden layers. The training of such architectures is called *deep learning*, and was once prohibitively time-consuming, because learning is typically achieved through gradient descent on the network's output error (defined as a function of its weights), and the components of this gradient are often very small for the weights of connections that are far away from the input layer (this is known as the vanishing gradient problem). However, the process is now facilitated by the availability of fast hardware and large datasets.

Once trained, the layers of the deep neural network can transform raw data (such as text data) into representations that become progressively more sophisticated and abstract with each layer, allowing the network to produce complex decisions (e.g., classifying text documents) at the output layer. As a result, deep learning has achieved unprecedented results in areas such as speech processing, machine translation and computer vision, and is presently being utilized within (or integrated into) many practical applications and products.

To implement classifiers, we use efficient implementations of

- Natural Language Toolkit (NLTK 3.0 [15]) for Naive Bayes,
- scikit-learn [16] for SVM, and
- Computational Network Toolkit (CNTK) for deep learning with FNNs; CNTK describes neural networks as a series of computational steps via a directed graph (where leaf nodes represent input values or network parameters, while other nodes represent matrix operations upon their inputs) [17].

IV. Building a Classifier

First, we train classifiers (Naive Bayes, SVM and deep learning) and evaluate their accuracy in text classification. We use Reuters-21578 (Distribution 1.0 Text Categorization

978-1-5090-6356-7/17/$31.00 ©2017 IEEE

Collection Data Set [18]) for both training and test data. Reuters-21578 is a collection of documents that appeared on Reuters newswire in 1987 and was manually classified into text categories. We use Reuters-21578 text data as follows.

- We consider 5952 documents, where top two topics "earn" and "acq" constitute the two classes.
- We use 4234 documents for training.
  - 1533 of these documents are from class 1.
  - 2701 of these documents are from class 2.
- We use 1718 documents for testing.
  - 677 of these documents are from class 1.
  - 1041 of these documents are from class 2.
- We build features in terms of word distributions.
  - Deep learning uses all word distributions as features.
  - The classifier under attack (other than deep learning) uses top word distributions as features.

The classifier error is calculated as the percentage of mismatched labels with respect to the ground truth and is given in Table I, where error on class 1 (or class 2) is the probability that a document in class 1 (or class 2) is classified into the other class and the overall error is the average probability that a document is classified into the wrong class. As expected, deep learning with FNNs has the lowest classification error.

TABLE I
THE CLASSIFIER ERROR (RELATIVE TO THE GROUND TRUTH).

Type of classifier	Error on class 1	Error on class 2	Overall error
Naive Bayes	10.68%	13.98%	12.69%
SVM	22.55%	9.20%	14.44%
Deep learning	3.55%	3.46%	3.49%

V. STEALING A CLASSIFIER

Next, we use half of the 1718 testing documents for polling a classifier of unknown type and structure, and using its labels to train the deep learning classifier as the functionally equivalent machine. Then, we use the other half of these 1718 documents for testing the similarity of labels returned by the classifier under attack and the functionally equivalent classifier built by deep learning. The error in similarity is calculated as the percentage of mismatched output labels between the classifier under attack and the equivalent machine. The output of the classifier under attack is viewed as the "ground truth", in this case.

Table II shows the error of adversarial classifier. This error is calculated as the percentage of mismatched labels with respect to the classifier under attack. Deep learning can produce very similar output as other classifiers while the reverse is not true. That is, deep learning can be effectively used to steal the functionality of other classifiers while other classifiers cannot reliably infer the functionality of a deep learning classifier. This is due to the versatility of deep learning,

TABLE II
THE ADVERSARIAL CLASSIFIER ERROR (RELATIVE TO THE OUTPUT OF THE CLASSIFIER UNDER ATTACK).

Type of attack	Error on class 1	Error on class 2	Overall error
Deep learning infers Naive Bayes	1.87%	2.27%	2.10%
Naive Bayes infers deep learning	9.91%	12.02%	11.18%
Deep learning infers SVM	2.59%	2.55%	2.56%
SVM infers deep learning	7.00%	9.69%	8.61%

where the space of learnable deep classifiers includes instances that are functionally similar to other classifier instances. On the other hand, the space of other classifiers (Naive Bayes, SVM and similar classifiers) is smaller and thus may not include instances that are functionally similar to certain deep classifier instances. In these experiments, hyperparameters of deep learning are optimized by the adversary as follows:

- If the classifier under attack is Naive Bayes: 90 neurons are used per layer, train criterion is "CrossEntropyWithSoftmax", layer type is "RectifiedLinear", mean variance is not normalized, non-uniform initial input is used, batch size is 15, momentum is 0.9, maximum number of epochs is 20, and likelihood threshold is 0.55.
- If the classifier under attack is Naive Bayes SVM: 90 neurons are used per layer, train criterion is "CrossEntropyWithSoftmax", layer type is "Tanh", mean variance is normalized, uniform initial input is used, batch size is 25, momentum is 0.9, maximum number of epochs is 18, and likelihood threshold is 0.5.

If we use the deep learning classifier that is built by the adversary, the classification errors are shown in Table III. These classification errors are combinations of the error of the original classifier (relative to the ground truth) and the error of the adversarial classifier (relative to the original classifier). In particular, the classification error is $13.85\% \in [12.69 - 2.10, 12.69 + 2.10]$ if the original classifier is Naive Bayes, and is $13.74\% \in [14.44 - 2.56, 14.44 + 2.56]$ if the original classifier is SVM.

TABLE III
THE ADVERSARIAL CLASSIFIER ERROR (RELATIVE TO THE GROUND TRUTH).

Classifier under attack	Error on class 1	Error on class 2	Overall error
Naive Bayes	11.57%	15.33%	13.85%
SVM	20.77%	9.20%	13.74%

Next, we evaluate how the number of queries made by the adversary impacts the prediction accuracy of the adversary.

978-1-5090-6356-7/17/$31.00 ©2017 IEEE

TABLE IV
THE IMPACT OF TRAINING SET SIZE ON THE ADVERSARIAL CLASSIFIER ERROR (RELATIVE TO THE OUTPUT OF THE CLASSIFIER UNDER ATTACK).

Training data size	Error on class 1	Error on class 2	Overall error
Deep learning infers Naive Bayes			
10	6.95%	10.10%	8.73%
100	5.35%	5.36%	5.36%
250	3.74%	3.71%	3.73%
500	3.48%	3.51%	3.49%
859	1.87%	2.27%	2.10%
Deep learning infers SVM			
10	18.12%	14.91%	16.07%
100	5.83%	7.27%	6.75%
250	3.88%	3.82%	3.84%
500	2.91%	2.91%	2.91%
859	2.59%	2.55%	2.56%

We used 859 documents (one half of 1718 documents, as stated earlier) to poll the classifier under attack and obtain results in Table II. In reality, we may not be able to query the original classifier for its classification for too many inputs (due to cost or other restrictions) and thus the training data for the functionally equivalent classifier may be limited. To evaluate the effects of such limitations, we now reduce the number of inputs and obtain results in Table IV for the adversarial classifier error (relative to the output of the classifier under attack). As we expected, the error increases as the number of documents in training data decreases.

Note that the fidelity of exploratory attacks can be further improved with the use of *active learning* [2], [19]. In active learning, the adversary optimally selects the input data to request labels from the classifier under attack.

VI. CONCLUSION

We presented an exploratory machine learning attack approach that uses deep learning to build a functionally equivalent machine for an online classifier of unknown type and structure where the adversary can request labels for input data. Results show that deep learning can reliably infer the functionality of Naive Bayes and SVM classifiers (i.e., the functionally equivalent classifier built by the adversary produces labels close to the classifier under attack), while the reverse is not true. Therefore, there is an emerging need to identify strategies for mitigating such exploratory black-box attacks with deep learning to protect online information systems.

ACKNOWLEDGMENTS

We thank Dr. Prateek Mittal for valuable discussions on adversarial machine learning.

REFERENCES

[1] L. Huang, A. Joseph, B. Nelson, B. Rubinstein, and J. Tygar, "Adversarial Machine Learning," *Proc. ACM Workshop on Artificial Intelligence and Security*, 2011.
[2] B. Miller, A. Kantchelian, S. Afroz, R. Bachwani, E. Dauber, L. Huang, M. Tschantz, A. Joseph, and J. Tygar, "Adversarial Active Learning," *Proc. ACM Workshop on Artificial Intelligence and Security*, 2014.
[3] P. Laskov and R. Lippmann, "Machine Learning in Adversarial Environments," *Journal of Machine Learning*, 81(2):115-119, 2010.
[4] M. Barreno, B. Nelson, R. Sears, A. Joseph, and J. Tygar, "Can Machine Learning be Secure?" *Proc. ACM Symposium on Information, Computer and Communications Security*, pp. 16-25, 2006.
[5] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," *Proc. IEEE European Symposium on Security and Privacy*, 2016.
[6] L. Pi, Z. Lu, Y. Sagduyu, and S. Chen, "Defending Active Learning against Adversarial Inputs in Automated Document Classification," *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec. 2016.
[7] G. Ateniese, L. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers," *International Journal of Security and Networks*, 10(3):137-150, 2015.
[8] M. Fredrikson, S. Jha, and T. Ristenpart, "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures," *Proc. ACM SIGSAC Conference on Computer and Communications Security*, 2015.
[9] F. Tramer, F. Zhang, A. Juels, M. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs," *Proc. USENIX Security*, 2016.
[10] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion Attacks Against Machine Learning at Test Time," *Proc. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2013.
[11] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Celik, and A. Swami, "Practical Black-Box Attacks Against Deep Learning Systems Using Adversarial Examples," *arXiv preprint arXiv:1602.02697*, 2016.
[12] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial Examples in the Physical World," *arXiv preprint arXiv:1607.02533*, 2016.
[13] C. Liu and P. Mittal, "LinkMirage: Enabling Privacy-Preserving Analytics on Social Relationships," *Proc. Network and Distributed System Security Symposium (NDSS)*, 2016.
[14] F. Scarselli and A. Tsoi, "Universal Approximation using Feedforward Neural Networks: a Survey of Some Existing Methods, and Some New Results," *Neural Networks*, 11:15-37, 1998.
[15] http://www.nltk.org
[16] http://scikit-learn.org
[17] https://www.cntk.ai
[18] Reuters-21578, Distribution 1.0, http://www.daviddlewis.com/resources/testcollections/reuters21578
[19] S. Tong and D. Koller, "Support Vector Machine Active Learning with Applications to Text Classification," *Journal of Machine Learning Research*, 2:45-66, 2001.

978-1-5090-6356-7/17/$31.00 ©2017 IEEE