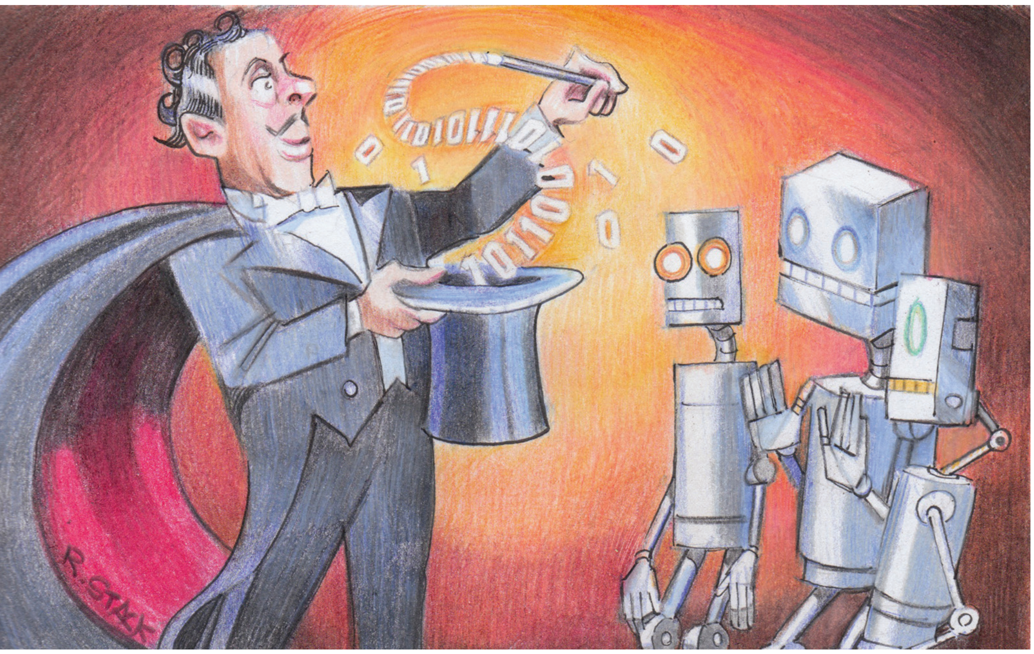
Editors: Sean W. Smith, sws@cs.dartmouth.edu | William Enck, whenck@ncsu.edu

Machine Learning in Adversarial Settings

Patrick McDaniel, Nicolas Papernot, and Z. Berkay Celik | Pennsylvania State University



Advances in machine learning have led to transformational new fields of technology and introduced capabilities not previously possible. Emerging applications in self-driving cars, data analytics on massive datasets, adaptive and interactive entertainment, and Web search and sentiment analysis are but a few of the technologies that will impact society in the decades to come.

Perhaps no technology field has relied on or benefited from advances in machine learning more than systems and computer security. Machine learning is the basis for almost all nonsignature-based detection, whether identifying malware, network intrusions, spam, rogue processes, fraudulent transactions, or other malicious activity. Indeed, machine learning has become so intertwined with security that the technical community's ability to apply machine learning securely will likely be crucial to future environments.

In this article, we consider whether today's use of machine learning in security-sensitive applications is vulnerable to nonobvious and potentially dangerous manipulation. Here, we examine sensitivity not only in the context of computer security but also in any application whose misuse might lead to harm—for instance, crashing an autonomous vehicle or bypassing a content filter. We explore the use of machine learning in this area particularly in light of recent advances in the computationally efficient creation of adversarial samples targeted at widely used classes of machine-learning approaches.

Machine Learning in Practice

Consider a generalized use of machine learning as a classifier and an example system identifying spam email. (For brevity, we restrict ourselves to machine-learning classifiers—identifying a sample as being from some output class, among a predefined finite set of [potentially] many classes—trained on labeled data. Many other kinds of machine-learning systems and training techniques exist; our arguments apply almost universally.) A classifier is a system that takes an input sample and identifies it as one of several output classes (or none, if the sample can't be identified confidently). In this example, the system determines whether the item is in the "spam" or "not spam" class.

In machine learning, each sample is input into the classification process as a vector of features that describe the sample. For email, typical features might be keywords, sender and recipient domain names, existence of embedded content, or number of emails of a particular type. The system determination is based on how that set of input features is interpreted by the model for the classification process—in this case, a model of how email input features indicate spam or not.

Conceptually, a model encodes semantic information about how certain features or sets of features relate to the output class. For example, certain keywords or keyword combinations could be strong indicators of an email being spam. In practice, models will encode many different such relationships, each weighted on the basis of the association's strength. An aggregate calculation over the feature associations with respect to the input features results in an output classification and/or confidence score.

To date, the key assessment metric for these systems has been accuracy: How often does the model pick the correct class for a sample? Several accuracy measures exist, including precision, sensitivity, and specificity. These quality assessments directly relate to assumptions about the expected distribution of the classification system input and don't account for adversarial behavior, which often falls outside of this expected input distribution. In other words, accuracy can be viewed as a measure of the system's average performance, whereas the security evaluation is interested in worst-case performance.

Adversarial Samples

One of the limitations of machine learning in practice is that it's subject to adversarial samples. Adversarial samples are carefully modified inputs crafted to dictate a selected output. In the context of classification, adversarial samples are crafted to force a target model to classify them in a class different from their legitimate class—for instance, spam emails that bypass a spam filter. The modifications, called *perturbations*, are introduced to yield a specific adversary-selected misclassification. In general, adversaries want to perturb the sample as little as possible so that to a human observer, for example, it remains indistinguishable from the original unaltered sample.

In the context of classification, adversarial samples are crafted to force a target model to classify them in a class different from their legitimate class.

Over the past few years, several algorithms used to automate adversarial-sample generation have emerged for multiclass classifiers built, for example, with deep neural networks. In late 2013, Christian Szegedy and his colleagues were the first to reveal the vulnerability of trained deep neural networks to slight perturbations of their inputs when they cast sample generation as an approximate optimization.[1] Ian Goodfellow and his colleagues followed with a fast gradient sign method, which linearly approximates the cost function in the neighborhood of legitimate samples to allow faster crafting of adversarial samples.[2] Finally, Nicolas Papernot and his colleagues proposed an iterative crafting algorithm that uses the model's Jacobian to select perturbations yielding the adversary's desired classification. They showed that adversaries can reliably achieve chosen adversarial target classes for any legitimate source class.[3] Their iterative approach also allows greater control over the introduced perturbations, thus reducing their magnitude. These more recent works expand on the classical adversarial machine-learning efforts described by Pavel Laskov and Richard Lippmann.[4] For example, related past work explored the formalization of worst-case errors against learned binary classifiers,[5] reverse engineering of binary linear classifiers to identify inputs they misclassify,[6] and contamination of training data jeopardizing binary classifiers' integrity and availability.[7]

Consider the following real-world scenario in which an autonomous vehicle uses a camera to identify and recognize roadside signs (see Figure 1). Once a sign has been identified, its image is fed to a neural network for classification in one of the predefined sign classes. Here, the neural network identifies the sign as a stop sign. Now, consider adversaries capable of altering the input of this neural network. They can force the model to output a wrong class upon processing a slightly perturbed variant of the stop sign's image. If adversaries can transfer perturbations to the neural network's image input, the autonomous system can be misled into misclassifying signs—reading stop signs as yield signs, for instance—potentially resulting in vehicles crashing into one another.

Again, to humans, adversarial samples are often indistinguishable from original samples. Humans would classify both images in Figure 2 as stop signs. In real-world tests using the Papernot algorithm, a trained deep-learning neural network classifies Figure 2a as a stop sign and Figure 2b as a yield sign. In actuality, the image on the left is an ordinary image of a stop sign, whereas the image on the right is an adversarial sample crafted by solving the earlier optimization problem.

Learning Models from Training Data

To understand why adversarial samples exist, it's important to explore how learning models are

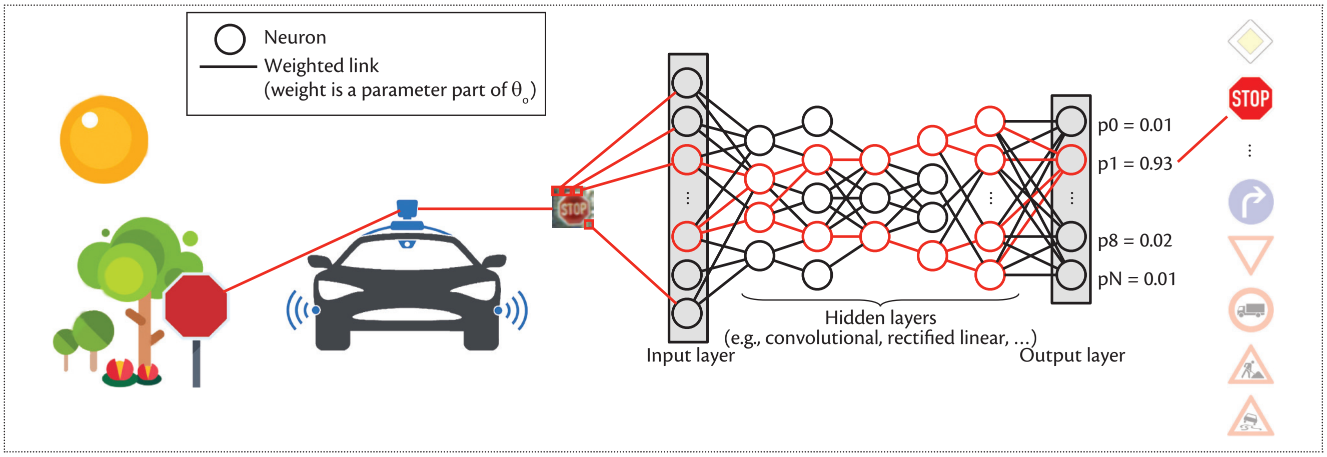


Figure 1. An autonomous vehicle uses a camera to identify and recognize roadside signs. Once a sign has been identified, its image is fed to a neural network for classification in one of the predefined sign classes. Here, the neural network identifies the sign as a stop sign.

built. Although there are other approaches, the models we discuss here are trained in a supervised fashion using labeled training data. This training data is a corpus of samples taken from the expected input distribution and labeled with their class. In the case of our spam system, this sample data would be a large number of emails that indicate whether or not they are spam. In the sign recognition system, the training data would include numerous signs and their type: stop, yield, and so on. These labels are taken as ground truth in constructing the models to be used at runtime.

Generally, model training begins with a null model representing no information. The training method iteratively processes each input sample in the training data and updates the model. This iterative refinement process strengthens or weakens the classification associations as supporting evidence is identified. Generally, the larger and more diverse the training data is, the more accurate the system becomes.

The refinement process of the input data's internal representations is specific to the kind of machine-learning technique employed: shallow and deep neural networks represent the model as a complex feed-forward network of mathematical neurons (parameterized elementary computing units), support vector machines use high-dimensional hyperplanes to separate classes, and random forests represent the model as a collection of learned decision trees. Some machine-learning techniques don't store a model—for instance, nearest neighbors—but simply use lazy evaluation to compare unseen samples to the training samples.

Regardless of technique, the model represents an approximation of the phenomena being modeled; unless the training data contains all possible input feature vectors, it can't fully capture a complete model of the target domain. In nonadversarial environments, this often isn't a problem. Data representative of the expected input distribution is sufficient for training. With enough input emails or images of signs to train on, input normally encountered at runtime will be sufficiently similar to allow the model to output a correct classification prediction by extrapolating from training samples.

Exploiting Natural Complexity in Decision Boundaries

A problem arises when adversaries exploit the system by providing input samples that aren't within the expected input domain. Here, they use information about the system to find where the model is inaccurate owing to items missing from the training set.

Consider an unsophisticated sample-generation algorithm in which adversaries simply test different input samples until they find a combination of input features that reliably achieves the desired classification. For example, spammers could simply modify email typography, vocabulary, addresses, and domains; test against the system; and see which are marked as legitimate (not spam). Indeed, this is common practice today; each new spam campaign contains carefully tested and selected email features that reliably bypass online spam filtering systems.

Figure 3 illustrates model training and use. In this figure, the plane represents all possible input feature vectors. For each sample, the input feature values uniquely identify its coordinates in the plane. Two classes A and B (that is, spam and not spam) are regions in a two-dimensional plane separated by the smooth curved line. All samples above the smooth curved line are in class A, and those below are in class B. This line is called the *real* *decision boundary*. The model is trained using the input samples labeled X. On the basis of these samples, the training algorithm approximates the class separation as the linear dashed line—the *model decision boundary*. The distance between the real and model decision boundary is called the *model error* or space of adversarial samples (adversarial regions).

One might intuit that the model the algorithm learned by was faulty, but this isn't true.

This is a legitimate and highly accurate model for the training data: every sample in the input sample distribution is correctly classified. Indeed, one can't do better than this without more samples or information; a natural error is introduced by the fact that the training data can't, in almost all circumstances, cover the entire feature space or provide enough data to illuminate the real decision boundary with anything other than approximate accuracy. Making matters worse, the real decision boundary generally becomes more complex as the phenomenon becomes more nuanced and the feature and dimension space becomes larger.

It's this complexity that adversaries exploit. They simply take a sample and use trial and error (as in our earlier spam example) or information about the model error (as in the recently developed adversarial sample algorithms) to find a few perturbations that "move" the sample into the region of adversarial samples.

Herein lies the crux of these systems' vulnerability. Because adversaries can control the input sample features, they explicitly drive the malicious sample into the regions of the input space that are ambiguous with respect to the model. In short, they search for or calculate a sample that's in one class (for example, spam or stop sign) but, owing to the ambiguity resulting from incomplete training data, is classified as being in another class (for example, not spam or yield sign).

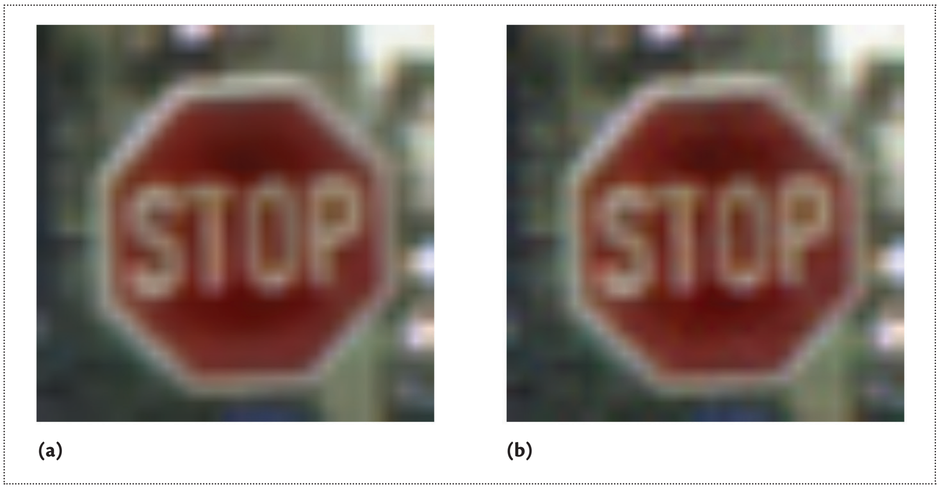


Figure 2. To humans, adversarial samples are indistinguishable from original samples. (a) An ordinary image of a stop sign. (b) An image crafted by an adversary.

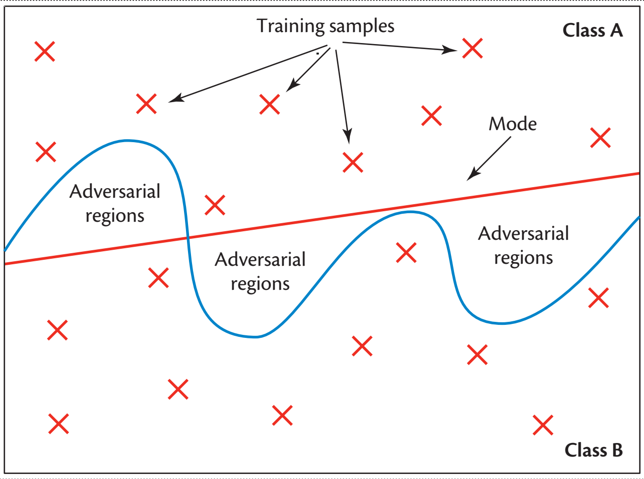


Figure 3. Model training and use. The plane represents all possible input feature vectors. For each sample, the input feature values uniquely identify its coordinates in the plane. Two classes A and B (that is, spam and not spam) are regions in a two-dimensional plane separated by the smooth curved line.

The Importance of Model Resilience

We argue that to address adversarial action, a new metric for machine-learning model quality is needed: *model resilience*.[8] Model resilience can be defined as robustness to perturbations of its input. Simply put, the more perturbation needed to move a sample from its legitimate class to an adversarial class, the more robust the model is to adversarial manipulations of its inputs.

In practice, we can achieve resilience in several ways. In the simplest approach, we can simply require higher confidence in outputs. This would move the decision boundaries further apart and thus leave fewer regions of ambiguity. This of course would affect model accuracy. Other approaches would be to refine the training process to smooth decision boundaries, or to measure each input's likelihood of being an adversarial sample based on its characteristics, for example, closeness to the centroid of a nonselected class. Such approaches aren't well understood, but they're certain to be a necessary element to securing the future of machine learning in adversarial settings.

Machine learning is driving rapid innovation and providing new insights into how we can interpret and control complex data and environments. With these advances, adversaries will seek to circumvent their controls and drive systems for their malicious ends. In recognition of this reality, the machine-learning and security communities must endeavor to inoculate systems against such misuse. Thus, we must revisit our measures of quality for machine-learning techniques and weigh not only the results they produce but also their ability to resist samples carefully generated by adversaries. ■

References

1. C. Szegedy et al., "Intriguing Properties of Neural Networks," *Proc. Int'l Conf. Learning Representations* (ICLR 14), 2014; http://arxiv.org/abs/1312.6199.
2. I.J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *Proc. Int'l Conf. Learning Representations* (ICLR 15), 2015; http://arxiv.org/abs/1412.6572.
3. N. Papernot et al., "The Limitations of Deep Learning in Adversarial Settings," *Proc. 1st IEEE European Symp. Security and Privacy* (EuroS&P 16), 2016; http://arxiv.org/abs/1511.07528.
4. P. Laskov and R. Lippmann, "Machine Learning in Adversarial Environments," *Machine Learning*, vol. 81, no. 2, 2010, pp. 115–119.
5. M. Kearns and M. Li, "Learning in the Presence of Malicious Errors," *J. Computing*, vol. 22, no. 4, 1993, pp. 807–837.
6. D. Lowd and C. Meek, "Adversarial Learning," *Proc. Knowledge Discovery in Data Mining* (SIGKDD 05), 2005, pp. 641–647.
7. B.A. Nelson, "Behavior of Machine Learning Algorithms in Adversarial Environments," PhD thesis, Dept. of Computer Science, Univ. California, Berkeley, 2010.
8. N. Papernot et al., "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks," to be published in *Proc. 37th IEEE Symp. Security and Privacy*, 2016.

Patrick McDaniel is a Distinguished Professor in the School of Electrical Engineering and Computer Science at the Pennsylvania State University. Contact him at mcdaniel@cse.psu.edu.

Nicolas Papernot is a graduate student at the Pennsylvania State University. Contact him at ngp5056@cse.psu.edu.

Z. Berkay Celik is a graduate student at the Pennsylvania State University. Contact him at zbc102@cse.psu.edu.