The challenge of verification and testing of machine learning

Jun 14, 2017

by Ian Goodfellow and Nicolas Papernot

In our second post, we gave some background explaining why attacking machine learning is often easier than defending it. We saw some of the reasons why we do not yet have completely effective defenses against adversarial examples, and we speculated about whether we can ever expect such a defense.

In this post, we explore the **types of guarantees one can expect a machine learning model to possess**. We argue that the limitations of existing defenses point to the lack of verification of machine learning models. Indeed, to design reliable systems, engineers typically engage in both testing and verification:

- By *testing*, we mean evaluating the system in several conditions and observing its behavior, watching for defects.
- By *verification*, we mean producing a compelling argument that the system will not misbehave under a very broad range of circumstances.

Orthogonal to this issue is the question of which input values should be subject to verification and testing. Do we intend to verify or test the system only for "naturally occurring" legitimate inputs, or do we intend to provide guarantees for its behavior on arbitrary, degenerate inputs? Many software systems such as compilers have undefined behavior for some inputs.

Should we test or verify? On which inputs?

Machine learning practitioners have traditionally relied primarily on testing. A classifier is usually evaluated by applying the classifier to several examples drawn from a test set and measuring its accuracy on these examples. By definition, these testing procedures cannot find all of the possible—previously unseen—examples that may be misclassified.

The verification analog of measuring test set error is *statistical learning* theory [V98]. Statistical learning theory provides guarantees that the test error rate is unlikely to exceed some

threshold, but these guarantees are often so conservative that they are not used by engineers in practice.

Even when statistical learning theory is applied, it is typical to consider only “naturally occurring” inputs: guarantees are expressed for points that are drawn from the *same* distribution as the training data.

Bringing adversaries in the equation

To provide security guarantees, it is necessary to improve along both axes: 1) we must use verification rather than testing, and 2) we must ensure that the model will behave safely on unusual inputs crafted by an attacker. One well-studied property that practitioners would like to guarantee is robustness to adversarial examples.

The natural way to *test* robustness to adversarial examples is simply to evaluate the accuracy of the model on a test set that has been adversarially perturbed to create adversarial examples [SZS13]. This applies the traditional testing methodology used in machine learning to a new set of inputs.

Unfortunately, testing is insufficient to provide security guarantees, because an attacker can send inputs that differ from those used for the testing process. For example, a model that is tested and found to be robust against the fast gradient sign method of adversarial example generation [GSS14] may be vulnerable to more computationally expensive methods like attacks based on numerical optimization [SZS13] or saliency maps [PMJ16].

In general, testing is insufficient because it provides a *lower bound* on the failure rate of the system when, to provide security guarantees, an *upper bound* is necessary. In other words, testing identifies n inputs that cause failure so the engineer can conclude that *at least* n inputs cause failure; the engineer would prefer to have a means of becoming reasonably confident that *at most* n inputs cause failure.

These limitations of testing encountered in the context of machine learning are reminiscent of those encountered throughout many other kinds of software design. When discussing methods for guaranteeing program correctness, Edsger Dijkstra said, “testing shows the presence, not the absence of bugs.”

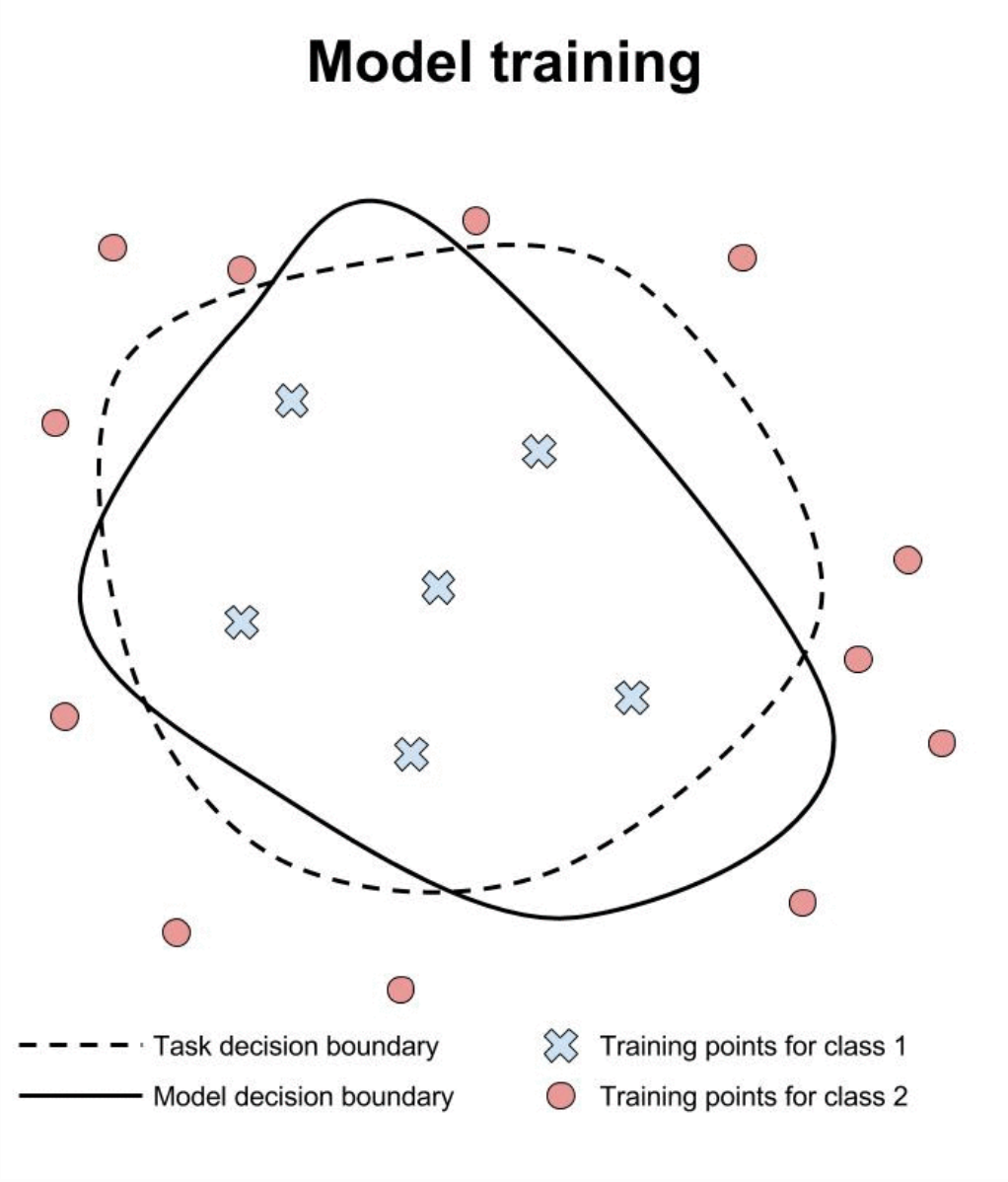
It is clear that **testing** of **naturally occurring inputs** is sufficient for traditional machine learning applications, but **verification** of **unusual inputs** is necessary for security guarantees. **We should verify, but so far we only know how to test**.

Current machine learning models are so easily broken that testing on unusual inputs is sufficient to expose their flaws. Hopefully, we will have better defenses against adversarial examples in the near future. Testing may no longer be sufficient to expose the flaws in such

models, and we will need verification techniques to study the effectiveness of the new defenses. **The development of a guarantee characterizing the space of inputs that are processed correctly is central to the future of ML in adversarial settings, and it will almost certainly be grounded in formal verification.**

Theoretical verification of ML

Verification of machine learning models' robustness to adversarial examples is in its infancy. Current approaches verify that a classifier assigns the same class to all points within a specified neighborhood of a point x. In the animation below, we illustrate this type of approach and compare it to testing individual points in the same neighborhood.



Researchers are working hard to build verification systems for neural networks. Unfortunately, these systems are not yet mature. Pulina et al. developed the first verification system for demonstrating that the output class of a neural network is constant across a desired

neighborhood [PT10]. This first system was limited to only one hidden layer, and was demonstrated only on networks with less than a dozen hidden units. In addition, the sigmoid activation function is approximated using constraints to reduce the problem to SAT.

Later, Huang et al. improved upon this initial method and proposed a new verification system applicable to modern neural network architectures [HKW16]. This system scaled to much larger networks, such as ImageNet classifiers. A remaining limitation of the new verification system is that it relies on a variety of assumptions—for example, that only a subset of the hidden units in the neural network are relevant to each input. These assumptions mean that the system can no longer provide an absolute guarantee of the absence of adversarial examples, because an adversarial example that violates these assumptions (for example, by manipulating one of the units that was assumed to be irrelevant) could evade detection.

Reluplex [KBD17] is another verification system that uses LP solvers to scale to much larger networks. Reluplex is able to become much more efficient by specializing on rectified linear networks [GBB11, JKL09, NH10] and their piecewise linear structure.

These current verification systems are limited in scope because they verify only that the output class remains constant in some specified neighborhood of some specific point x. There are two limitations to this:

1. We do not have a way to exhaustively enumerate all x points near which the classifier should be approximately constant (we cannot imagine all future naturally occurring inputs)
2. The neighborhoods surrounding x that we currently use are somewhat arbitrary and conservative; we tend to use L^p norm balls of small size because human observers agree that, for a small enough norm ball, all enclosed points should have the same class, but in practice the region of constant class should presumably be larger and have a different, less regular, less readily specified shape.

To summarize, verifying machine learning models first requires that we define the set of legal inputs, i.e., the set of inputs that we would like our model to correctly classify. This set of legal inputs is often much larger than the “test set” included in most benchmarks. Researchers will then have to design verification techniques that can efficiently guarantee the correctness of the machine learning predictions made on this entire set of legal inputs. Challenges often encountered in machine learning—such as the need to generalize to new inputs—may make this a particularly difficult goal to achieve, as indicated by efforts cited in this post [PT10, HKW16, KBD17]. If that is the case, techniques developed in other communities may enable partial verification of machine learning models through procedures closer to the ones of testing: a good example is the positive impact of fuzzing in the computer security community.

Is there a “no free lunch” theorem in adversarial settings?

It is worth considering the possibility that no verification system will ever exist because no machine learning model will ever be fully robust and accurate. In particular, the challenge of generalizing to inputs near new, previously unseen but legitimate inputs x seems difficult to overcome.

In the traditional machine learning setting, there are clear theoretical limits to how well a machine learning system can be expected to perform on new test points. For example, the “no free lunch” theorem [W96] states that *all* supervised classification algorithms have the same accuracy on new test points, when averaged over all possible datasets.

An important open theoretical question is whether the “no free lunch” theorem can be extended to the adversarial setting. If we assume that attackers operate by making small perturbations to the test set, then the premise of the “no free lunch” theorem, where the average is taken over all possible datasets including those where small perturbations should not be ignored by the classifier, no longer applies.

Depending on the resolution of this question, the arms race between attackers and defenders could have two different outcomes. The attacker might fundamentally have the advantage, due to inherent statistical difficulties associated with predicting the correct value for new test points. If we are fortunate, the defender might have a fundamental advantage for a broad set of problem classes, paving the way for the design and verification of algorithms with robustness guarantees.

Interested readers will find a preliminary discussion of this question in [PMS16]. The analysis characterizes the trade-off between model accuracy and robustness to adversarial efforts. It shows that in the presence of an adversary capable of finding a distribution that increases the learner’s loss, then the learner benefits from moving to a richer hypothesis class. A richer hypothesis class is informally defined as a more complex class of hypotheses that provides a lower minimum loss against any distribution. Hence, a tension potentially arises in the presence of limited data—because learning a more complex hypothesis would typically require more data in practice.

Reproducible testing with CleverHans

While verification is challenging even from a theoretical point of view, even straightforward testing can be challenging from a practical point of view. Suppose a researcher proposes a new defense procedure and evaluates that defense against a particular adversarial example attack procedure. If the resulting model obtains high accuracy, does it mean that the defense was effective? Possibly, but it could also mean that the researcher’s implementation of the attack was weak. A similar problem occurs when a researcher tests a proposed attack technique against their own implementation of a common defense procedure.

To resolve these difficulties, we have created the CleverHans library. This library contains reference implementations of several attack and defense procedures. **Researchers and product developers can use `cleverhans` to test their models against standardized, state-of-the-art attacks and defenses.** This way, if a defense obtains a high accuracy against a `cleverhans` attack, the test conclusively shows that the defense is strong, and if an attack obtains a high failure rate against a `cleverhans` defense, the test conclusively shows that the attack is strong. Moreover, results in published research are comparable to one another, so long as they are produced with the same version of CleverHans in similar computing environments.

Conclusion

The verification of machine learning models is still in its infancy, because methods make assumptions that prevent them from providing absolute guarantees of the absence of adversarial examples. We hope our readers will be inspired to solve some of these problems. In addition, we encourage researchers to use CleverHans to improve the reproducibility of machine learning testing in adversarial settings.

Acknowledgments

We would like to thank Martin Abadi for his feedback on drafts of this post. Thanks to Marta Kwiatkowska for pointing out a color error in the legend of the animation comparing testing to verification.

References

[GBB11] Glorot, X., Bordes, A., & Bengio, Y. (2011, April). Deep Sparse Rectifier Neural Networks. In Aistats (Vol. 15, No. 106, p. 275).

[GSS14] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

[HKW16] Huang, X., Kwiatkowska, M., Wang, S., & Wu, M. (2016). Safety Verification of Deep Neural Networks. arXiv preprint arXiv:1610.06940.

[JKL09] Jarrett, K., Kavukcuoglu, K., & LeCun, Y. (2009, September). What is the best multi-stage architecture for object recognition?. In Computer Vision, 2009 IEEE 12th International Conference on (pp. 2146-2153). IEEE.

[KBD17] Katz, G., Barrett, C., Dill, D., Julian, K., & Kochenderfer, M. (2017). Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. arXiv preprint arXiv:1702.01135.

[NH10] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10)

(pp. 807-814).

[PMJ16] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016, March). The limitations of deep learning in adversarial settings. In 2016 IEEE European Symposium on Security and Privacy (EuroS&P) (pp. 372-387). IEEE.

[PMS16] Papernot, N., McDaniel, P., Sinha, A., & Wellman, M. (2016). Towards the Science of Security and Privacy in Machine Learning. arXiv preprint arXiv:1611.03814.

[PT10] Pulina, L., & Tacchella, A. (2010, July). An abstraction-refinement approach to verification of artificial neural networks. In International Conference on Computer Aided Verification (pp. 243-257). Springer Berlin Heidelberg.

[SZS13] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.

[V98] Vapnik, V. (1998). Statistical Learning Theory.

[W96] Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. Neural computation, 8(7), 1341-1390.