Topic Proposal: Adversarial Machine Learning

In today’s world, machine learning classifiers are used to automate a variety of tasks, from email spam filtering and computer virus detection to image classification in defense, classification of environmental objects (such as in autonomous driving), and the medical field. In many cases, this involves utilizing supervised learning algorithms to train a classifier to classify input data into categories with data found in the relevant real-world equivalents. However, a significant vulnerability of these classifiers is their susceptibility to adversarial attacks by malicious entities, who may deliberately alter input data in such a way that the classifier misclassifies the data as a different category than the one it truly falls into. This has been seen to be effective in two different circumstances:

1. The malicious party (adversary) crafts adversarial training data that is used in the training of such a classifier, which causes the classifier to learn an incorrect decision-making function (poisoning), or:
2. The malicious party (adversary) crafts adversarial test data similar to clean data that is perturbed in a way that causes the classifier to classify the test data into a category it is not (evasion). In many cases, the alterations made to this adversarial test data remain near-imperceptible to humans.

In this project, we will focus on Scenario 2, where the adversary does not have access to the training data.

Even now, a decade after research into the field of adversarial machine learning began, classifiers continue to remain susceptible to adversarial attacks by malicious parties. As of now, the goal of research into the defense of machine learning classifiers in the field of adversarial machine learning is to modify classifiers such that they are minimally susceptible to adversarial attacks that take advantage of this vulnerability. In order to pursue this goal, researchers have focused on three approaches:

1. Modify Training Data: Adversarial examples, obtained by simulating an adversary or from nature, can be inserted into training data. Alternatively, training data can be “denoised”, as this is thought to make the trained model more robust.
2. Modify Test Data: Denoise test data before invoking the trained model. This is thought to “remove” adversarial perturbations, thus making the model robust to adversarial noise.
3. Modify ML model either during or after training to make it more “smooth”, and hence less susceptible to “small” adversarial perturbations. Since “small” perturbations are those that are imperceptible to humans, this is thought to better align the model with what a human would expect.

—---------------- More on Approach A —--

Blah blah blah

—-- More on Approach B —-

Blah blah

Researchers focusing on challenge A listed have approached it with two primary methods..

For one, researchers have trained classifiers on adversarial samples, allowing such classifiers to recognize adversarial data alongside clean data. This approach of using adversarial data to train a model is currently the most popular approach, as it is thought to be the most accurate of approaches tested with large-scale datasets (and deep learning), and has incurred many advancements in order to prevent other vulnerabilities caused by the use of adversarial samples as training data from forming.

This approach involves the use of different methods by which the creator of a classifier could compile adversarial data to feed into the model. Often, the approach involves using techniques to model the uncooperative ‘game’ between the malicious party (adversary) and the classifier (defender) in order to generate adversarial sample data to use in the testing of a model. Alternatively, what many have done in the past is collect data inputted into the model, some of which is adversarial data, and train on this data. However, by using test data as training data, the process may create classifiers that are vulnerable to ‘poisoning’ (referring to the idea that malicious parties may create adversarial data with the intention that it is added to a training dataset and thus results in a classifier that has learned an incorrect decision making function). In addition, this approach can cause a form of overfitting when used with adversarial data, wherein the classifier uses certain features of adversarial data to predict the category of input data, causing the classifier to become more accurate in predicting the label for adversarial data and less effective with clean data.

In addition to using adversarial samples in training data, researchers have also tried denoising input data before feeding it into the algorithm training the classifier, and then denoising all test data. However, denoising training data has, in some cases, caused classifiers to become vulnerable to general noise in test data.

Researchers hoping to address challenge B have done so by denoising input test data before feeding it into a machine learning classifier at the time of testing, thus eliminating as much adversarial noise as possible. Researchers have found this method to be particularly ineffective, as outlined by [this](https://arxiv.org/pdf/2012.09384.pdf) paper. Even if most perturbations are eliminated, satisfactory robustness has not been found. Strategies for adaptive compression have been used to improve denoising strategies, but

Researchers focusing on challenge C have approached it by trying to modify the algorithm used to train the classifier such that the resultant classifier contains decision-making functions that are not easily subverted by minor perturbations that are unnoticeable or not easily noticeable by humans. One example of this is the technique known as [Defensive Distillation](https://arxiv.org/abs/1511.04508). However, the effectiveness of this technique is unknown with regard to larger datasets—it appears to be highly effective with MNIST and CIFAR, but its efficacy remains unknown in larger-scale datasets and deep learning.

Given that the technique of modifying or introducing perturbed data before training of a classifier has been tested with a variety of different methodologies, and the technique of denoising data has been tested with low success rates, I hope to pursue the third option. And thus, the question I hope to address is as follows: How can we modify the algorithm within a machine learning classifier in order to make it less susceptible to adversarial attacks involving perturbations not easily noticeable by humans?

<https://arxiv.org/pdf/2012.09384.pdf>

<https://arxiv.org/pdf/1712.03141.pdf>

<https://archive.ischool.berkeley.edu/tygar/papers/SML2/Adversarial_AISEC.pdf>

<https://www.semanticscholar.org/reader/2e9ae718598df1d0884725fe60d485d7c1507c18>

<https://www.sciencedirect.com/science/article/pii/S209580991930503X>

<https://arxiv.org/pdf/1611.01236.pdf>

<https://arxiv.org/pdf/1511.04508.pdf>

<https://arxiv.org/pdf/1807.06732.pdf>

Not a paper, just an interesting resource: https://medium.com/mlearning-ai/researchers-discover-possible-reason-why-adversarial-perturbation-works-f65e64d9eb7

Notes

Important Message I sent:

Hi Professor Dughmi,

Before I start, I want to apologize for the confusion caused by my limited understanding of much of machine learning terminology, and preemptively apologize for any confusion I cause in this message. Much of my machine learning-related vocabulary comes from reading papers and from those around me, and thus I don't have a fully rigorous definition of the terms I have used thus far.

I wanted to expand on why I was hesitant to pick between robustness against adversarial perturbations and robustness against general noise and ask for your input regarding this. I said before that adversarial training was objectively better than smoothening algorithms for classifiers at increasing the robustness against adversarial examples for deep models used in classification.

From the literature, it's very clear that adversarial training is the current best method for pure adversarial robustness, especially because methods of smoothening—such as defensive distillation—make resultant classifiers output less accurate predictions when given clean test data, because they become less confident in their predictions. This is also directly stated in Jeong et. al (https://doi.org/10.48550/arXiv.2212.09000), although the researcher's methodology was related to a different strategy.

At the same time, adversarial training is not without its own limitations. As discussed briefly by Liu et. al (https://doi.org/10.48550/arXiv.1909.09034), most adversarial training does not adequately defend against the "generalized types of noise" and "remain vulnerable to iterative attacks." Despite this, as outlined by Ren et. al (https://doi.org/10.1016/j.eng.2019.12.012), "PGD adversarial training achieves state-of-the-art accuracy against a wide range of attacks on several DL model benchmarks such as the modified National Institute of Standards and Technology (MNIST) database, the Canadian Institute for Advanced Research-10 (CIFAR-10) dataset, and ImageNet."

The problem is, when asking the question of what is objectively the better strategy between smoothening classifiers and adversarial training in the context of exclusively adversarial machine learning or exclusively "generalized types of noise," there are clear answers. I don't really know what I'd be adding to the current scientific understanding of the field if I picked one of the two and situated it in a game theory context—it does become a high-level, more philosophical question, but in the end the answer will still be the same as what I've seen in scientific literature.

I do like the idea of conducting a survey but within the context of a game and focusing on the role of the "smooth classifier" strategy as opposed to other, more developed strategies such as adversarial training. However, of the options we have so far:

1. Focusing on exclusively the value that these strategies have with respect to increasing robustness against "general noise."

2. Focusing on exclusively the value that these strategies have with respect to increasing robustness against adversarial perturbations.

3. Focusing on neither, and instead just surveying the strategies, and analyzing out their pros/cons to determine their most viable uses.

None seem like they would particularly add to current scientific understanding, in the format that they're in. At least, I don't see how I could make this unique unless I created some sort of utility cost function involving the costs of the occurrences of both adversarial perturbations and "general noise" and then compared the two strategies by using the function to somehow create a measure of "ideal robustness." If given time, I could probably think about it and come up with such a function, perhaps based on frequency of occurrence of either factor or perhaps their individual costs to the creator of the classifier (these costs could be unique). With the time I have left, I don't know if that's something I could feasibly calculate.

But then again, perhaps I'm not perceiving this the way you did. I would greatly appreciate it if you could offer another perspective for me to reflect on, because right now, I don't think I'm "getting it," so to speak.

Also, in terms of the idea I had of discussing the orthogonality between different methods (and focusing on how to improve the strategy of "classifier smoothing"), I was thinking of coding and testing the robustness of methods that have not been used in conjunction before (such as defensive distillation and ensemble methods of adversarial training, or dropconnect and ensemble training) in order to improve current understanding of the orthogonality between these methods. I don't know if I also miscommunicated that idea, but I wanted to let you know just in case to see your thoughts.

Thank you for all of your help and support so far. Also, apologies if I came off as defensive or nitpicky today during the session; that was never my intention—I just wanted to understand the depth of the research idea you were discussing and again, I don't think I fully understood. Regardless, I really appreciate all of your advice.

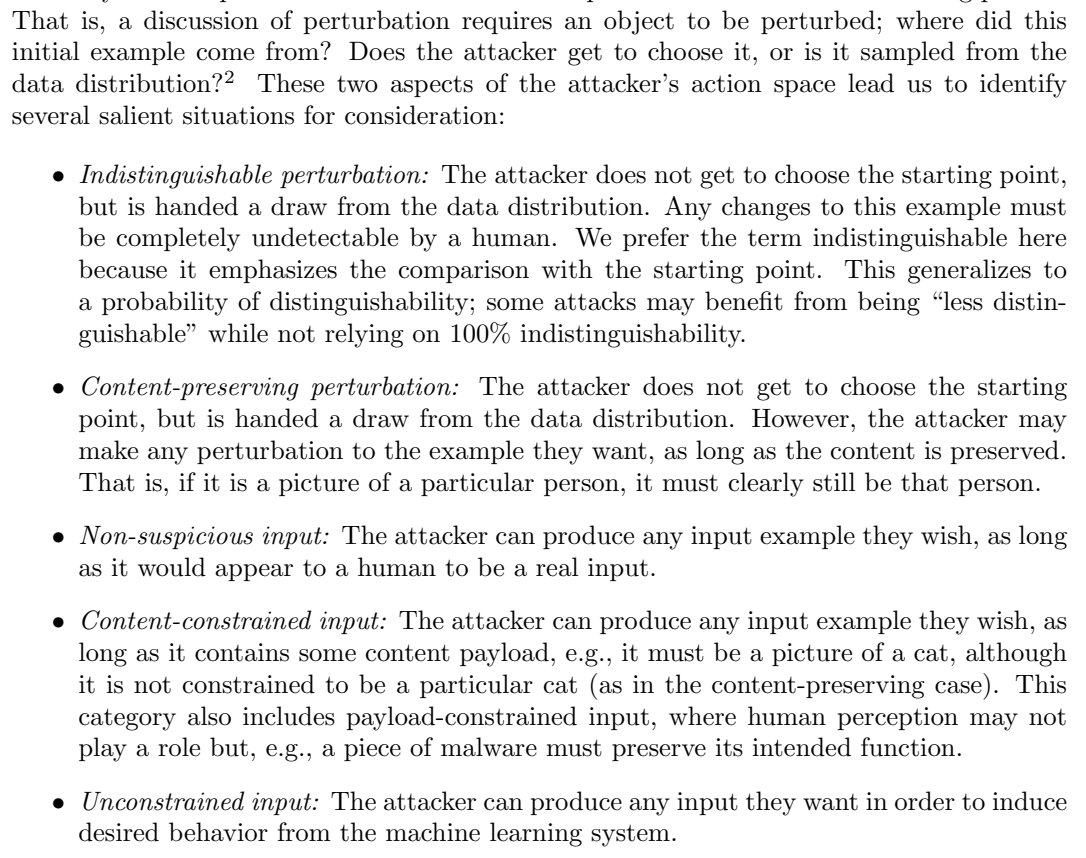
Best regards,

Aadi Shah

In today’s world, machine learning models are used for a variety of tasks

Adversarial threats propose problems for models in a variety of domains using machine learning based classifier systems, such as automated email spam filtering, antivirus software, image classification in medical and defense applications, and text-based sentimental analysis algorithms used on social media data. Adversarial attacks may include inducing perturbations or noise within sample data, or removing features from the data.

Methods of inducing perturbations in an adversarial sample:

* Take the example just past a decision boundary of a machine classifier so that it is assigned a different label than it should be.
  + The main problem is determining the level of perturbation such that the model classifies it as something it is not and a human would classify it as what it is (imperceptible to humans but perceptible to machines)
* 
* White box attacks
  + Adversaries have full knowledge of target model including architecture and parameters and can thus directly craft adversarial samples
  + Examples of attack algorithms:
    - limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm
    - the fast gradient sign method (FGSM)
    - the basic iterative method (BIM)/projected gradient descent (PGD)
    - distributionally adversarial attack
    - Carlini and Wagner (C&W) attacks
    - Jacobian-based saliency map attack (JSMA)
    - DeepFool
* Gray box attacks
  + Adversaries simply know the structure of the target model
* Black box attacks
  + Adversaries can only resort to query access to generate adversarial samples

Games modeling adversarial machine learning:

* Two player noncooperative (sequential, not simultaneous - learner publishes classifier) game - interaction between 2 or more players over a resource that must be shared between the two players
  + Two-player zero sum game
    - Nash equilibrium can be calculated using minimax theorem - only true for zero sum, not general sum
    - Zero sum game may be overly pessimistic - utility loss of learner may not = utility gain of attacker
  + Leader (learner) does not have information about how the follower (adversary) will select a strategy, so incorporates uncertainty via a Bayesian game
    - Ex. learner may not know about cost for adversary to generate adversarial data
    - Each player assumed to have a set of types and probability distribution over types is known, though exact types are not
    - Defensive models assume a set of types and a probability distribution over types, before selecting a strategy based on expected utilities
    - Both leader and follower follow nash equilibrium strategy
  + “The problem is formulated as a constrained optimization problem and solved as a mixed integer linear program for the adversary that is then used by the learner to determine a robust classification strategy”

Methods of increasing a model’s robustness against adversarial attacks:

* Injecting adversarial examples into the training dataset
  + Runs the risk of a label leaking effect, making the model more perceptive of adversarial examples than clean data, decreasing efficiency/accuracy with clean data
  + Robust model: PGD adversarial training
    - “achieves state-of-the-art accuracy against a wide range of attacks on several DL model benchmarks such as the modified National Institute of Standards and Technology (MNIST) database, the Canadian Institute for Advanced Research-10 (CIFAR-10) dataset, and ImageNet”
* Game theory methods for zero sum games with minimax linear optimization solution
  + adversarial classifier reverse engineering (ACRE) - not based on game but formed basis for many adversarial methods
* Game theory methods for non zero sum games with nash equilibrium bilevel optimization
* input/feature transformations and denoising
* formulate an adversarial polytope and define its upper bound using convex relaxations
* Defensive distillation - training a DNN with knowledge transferred from other DNN
  + <https://arxiv.org/abs/1511.04508>

Topic Ideas & Proposals

Examples of game theory used in machine learning:

* adversarial machine learning in cybersecurity: modeled as a two player zero sum game between an attacker and a defender
  + <https://towardsdatascience.com/a-game-theoretical-approach-for-adversarial-machine-learning-7523914819d5>
  + ideas:
    - attacker side: generate adversarial strategies against ml algorithms
    - defender side: detect new transformation and adapt to attacker strategies
    - stackelberg game: leader vs. follower
      * ​​<https://www.stat.purdue.edu/~xbw/talk/advl-talk-50min.pdf>
    - example: email spam
  + develop a framework to find the mixed nash strategy for the defender
  + isolate the best strategy for defenders - model scenarios where they use different strategies and compare for research
  + <https://arxiv.org/abs/2211.14669>
  + <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119723950.ch13>
* develop new adversarial learning algorithms to solve for adversarial manipulations in supervised classification networks such as cnns (image processing specifically)
  + <https://proceedings.neurips.cc/paper/2020/file/0ea6f098a59fcf2462afc50d130ff034-Paper.pdf>
  + develop game theory techniques for training cnns to minimize change in data distribution and thus label change in class labeling in deep learning (stackelberg games?)
    - [https://github.com/alessandrosaviolo/evolving-cnns-using-ga](https://github.com/AlessandroSaviolo/Evolving-CNNs-using-GA)
    - <https://ieeexplore.ieee.org/document/7966196>
    - <https://link.springer.com/chapter/10.1007/978-3-030-99772-4_4>
    - [https://proceedings.neurips.cc/paper/2020/file/0ea6f098a59fcf2462afc50d130ff034-paper.pdf](https://proceedings.neurips.cc/paper/2020/file/0ea6f098a59fcf2462afc50d130ff034-Paper.pdf)
    - creating (evolving) better convolutional neural networks using genetic algorithms
      * <https://opus.lib.uts.edu.au/bitstream/10453/140920/2/02whole.pdf>
* generative adversarial networks and the role of game theory in characterizing it by modeling it as a two player zero sum game
  + isolate the best strategy for mechanism design - model scenarios where both neural networks adopt either the same or different strategies and compare for research to see if there is an optimal way to make gans
    - possibly create a new framework with a strategy that does not already exist in the bounds of gans
      * something with similar efficacy to deep convolutional networks
        + seems a bit ambitious
  + perhaps a possible project could include trying to use game theory in a novel way to better help the models converge