Introduction to Adversarial Machine Learning

Ramesh Kumar Sah

Background

Machine learning can be essentially understood as applied function approximation, where the approximation is learned by analyzing a dataset containing several examples on inputs and their corresponding outputs.

ML is usually practiced believing that the environment is benign during both training and testing. Specifically, the inputs x are assumed to be all drawn independently from the same probability distribution at both training and testing.

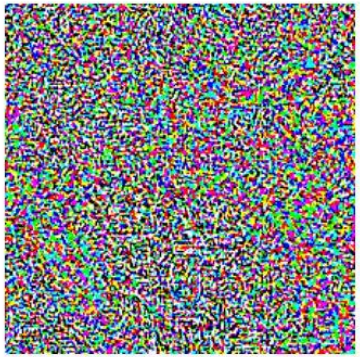
Such assumption implicitly rules out the possibility that an adversary could alter the distribution at either training (poisoning) or test time (evasion).

Adversary

Broadly speaking, an adversary is an agent that tries to harm or degrade a system.

An adversarial example is a sample of input data which has been modified very slightly in a way that is intended to cause a machine learning classifier to misclassify it.

$+ .007 \times$



$=$

$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

Source : Explaining and Harnessing Adversarial Examples, Goodfellow et al.

Nature of the Classifier

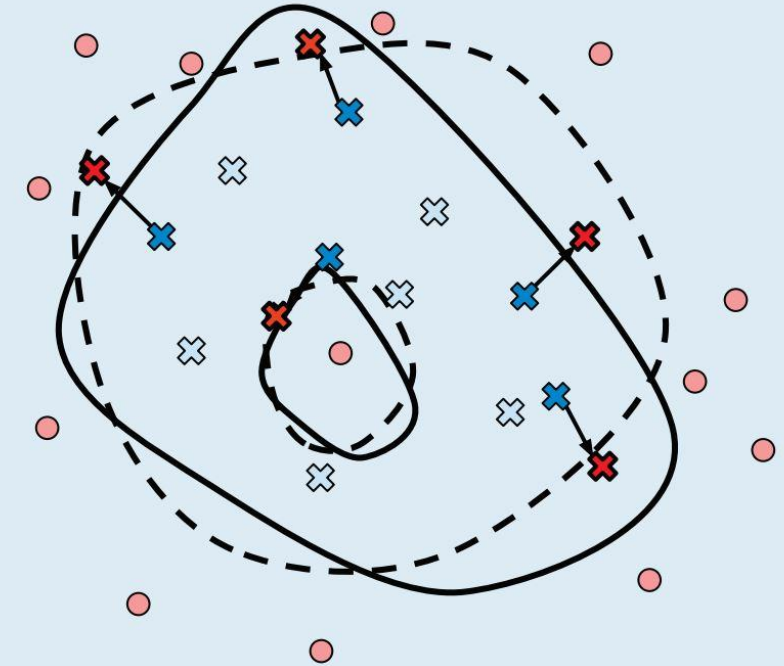
The classifier generally learns only an approximation of the true boundaries between regions for a number of reasons:

- **Learning from few samples**
- **Limited parametric model**
- **Imperfect optimization of model parameters.**

The model error between the approximate and expected decision boundaries is exploited by adversaries.

Figure 2. Example problem with two classes: the ideal decision boundary between the two models and the approximate boundary learned by the model.

Note that in higher dimensions, all examples are "close" to decision boundaries, as illustrated in this low-dimensional problem by the "pocket" of red class points included in the blue class.



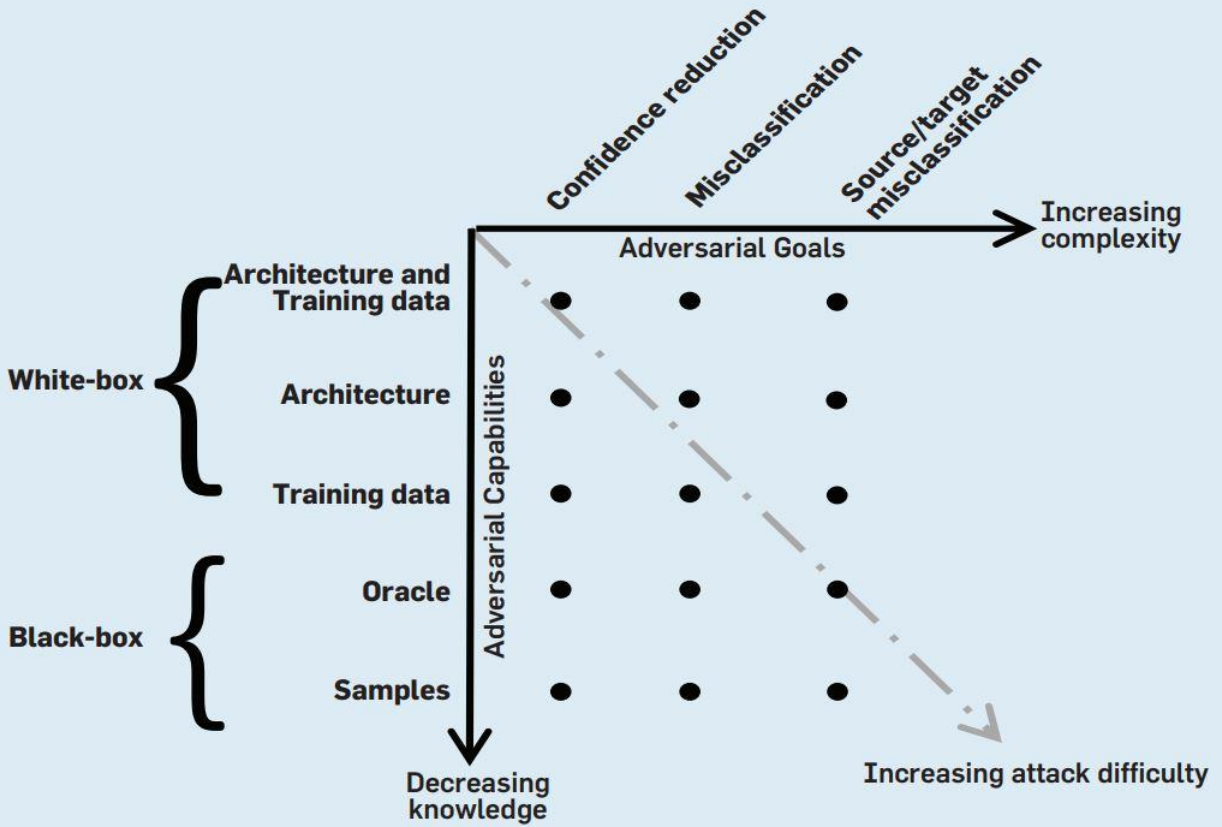
Source: [Making machine learning robust against adversarial inputs](), Goodfellow et al.

Adversary’s Goals and Capabilities

1. The adversary’s strength is characterized by its ability to access the model’s **architecture, parameter values, and training data.**
2. Adversaries are also distinguished by their ability to **submit inputs directly to the machine learning model or only indirectly through the data pipeline**.

An adversary with **access to model architecture and parameter values** is said to operate in a **white-box scenario** and an adversary with **no knowledge of the target system** operates in a **black-box scenario**.

Figure 4. A taxonomy of adversaries against machine learning models at test time.[22]

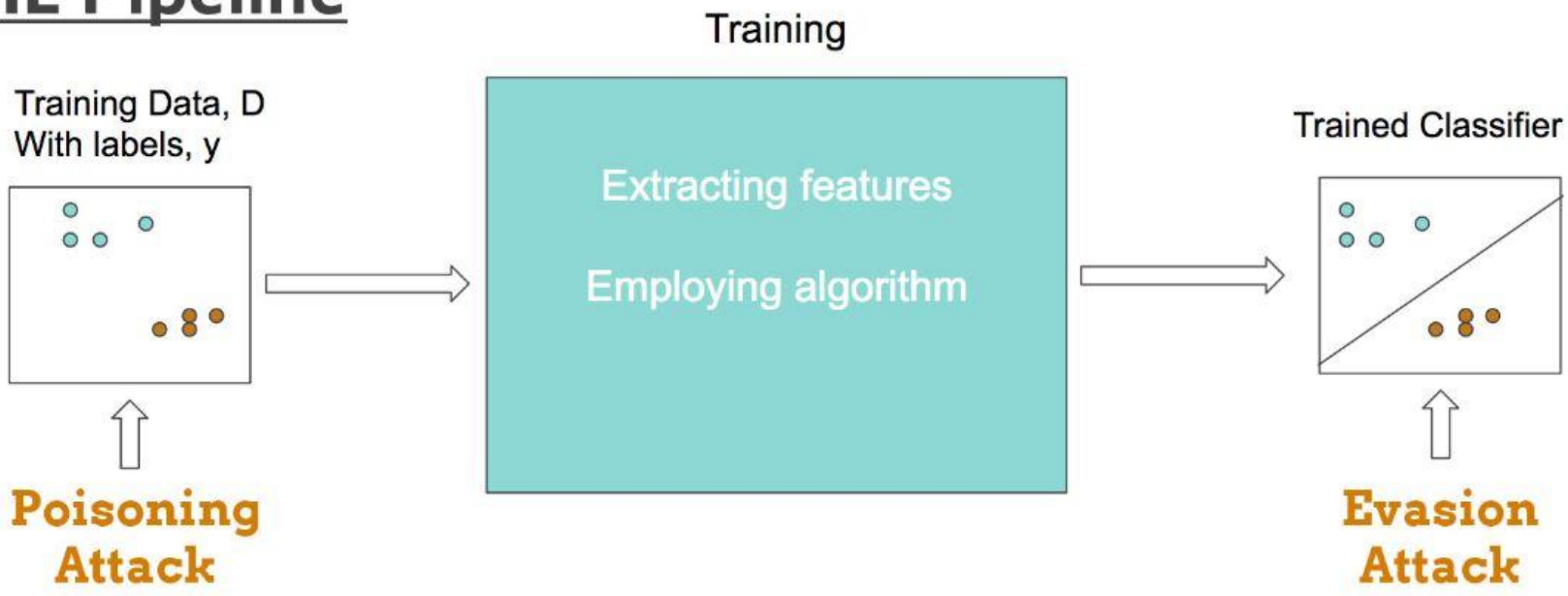


Source: [Making machine learning robust against adversarial inputs](), Goodfellow et al.

Attack Methods

1. **False learning or Poisoning Attack :** Corruption of training process by inducing false connections
2. **Evasion Attack**
    - **Untargeted Misclassification** : Misclassify an input into a class other than its correct class.
    - **Targeted Misclassification** : Classify an input into a class other than its actual class defined by the adversary.

ML Pipeline



Source : [Intro to Adversarial Machine Learning]

Adversarial Example Generation

How to generate an input that can fool the target model ?

In a white box scenario, constructing an adversarial example can be formulated as an optimization problem.

In a black-box scenario, the attacker must rely on guesswork.

Here it is necessary to impose constraints that ensure the adversary is not able to truly change the class of the input, rather only induce perturbations in the input.

White-box attacks

An adversarial example x* is found by perturbing an original input x, by solving a constrained optimization problem.

Solve for x* that causes the most expected **loss**, subject to a constraint on the **maximum allowable deviation** from the original input x.

OR

Solve for perturbation t such that it causes a **misclassification** with the **smallest possible value**.

Fast Gradient Sign Method (FGSM)

Introduced by Goodfellow et al., FGSM is a single step attack. It works by adjusting the input to maximize the loss using the backpropagation algorithm.

$$x^* = x + \epsilon.sign(\nabla_x J(x, y_{true}))$$

This is untargeted attack approach, since the adversarial example generated is not intended to maximize prediction probability for a particular class.

For the targeted attack we slightly modify the earlier equation to get.

$$x^* = x - \epsilon.sign(\nabla_x J(x, y_{target}))$$

Here we are trying to minimize the loss for input x such that the model predicts the label as Ytarget.

Basic Iterative Method

This is a simple extension to the FGSM method, where instead of applying the perturbation in a single step, it is applied multiple times with a small step size. This method was first introduced by Kurakin et al.

$$x_0^{adv} = x, x_{N+1}^{adv} = Clip_{x,\epsilon}(X_N^{adv} + \alpha.sign(\nabla_x J(x_N^{adv}, y_{true})))$$

We clip the result of backpropagation and summation to ensure that they are in an ε neighbourhood of the original input, within the range [X - ε, X + ε].

Iterative Least-Likely Class Method

This iterative least likely method tries to generate an adversarial example which will be classified as the class with the lowest confidence score in the prediction of the clean input.

$$y_{LL} = argmin_y p(y|X)$$

$$x_0^{adv} = x, x_{N+1}^{adv} = Clip_{x,\epsilon}(X_N^{adv} - \alpha.sign(\nabla_x J(x_N^{adv}, y_{LL})))$$

Black-box attacks

The black-box attacks are more realistic model of an adversary, in which the adversary has access to the model only through a limited interface.

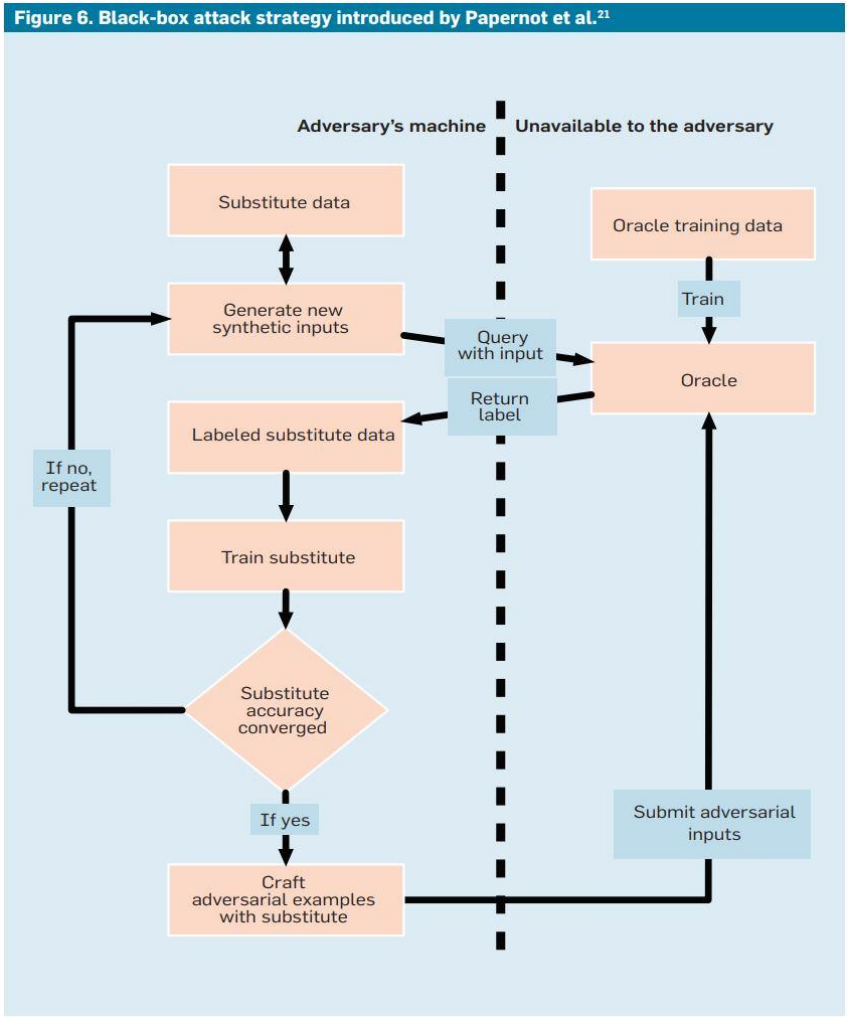
A possible strategy for the adversary is

- To train a new model, different from the one being targeted, called substitute.
- Use the substitute to compute the gradients required for the attack
- With assumption that both the target model and substitute operate in similar ways (have similar decision boundary), an adversarial example computed using the substitute is highly likely to be misclassified by the target model

Two ways to pursue such a strategy

1. The attacker collects and labels its own training set. This works with absolutely no access to the target model but can be expensive in terms of time and human effort.
2. The attacker is able to query the target model by sending it inputs and collect the outputs. Then a much less-expensive approach is to strategically send algorithmically generated inputs in order to reverse engineer the target model, without any (or very little) training data.

   Through interactions with the API, the adversary is going to create a training set for its substitute model, in a way that ensures the substitute makes predictions similar to the one made by the target model and is good indicator of where the targeted model will make mistakes.



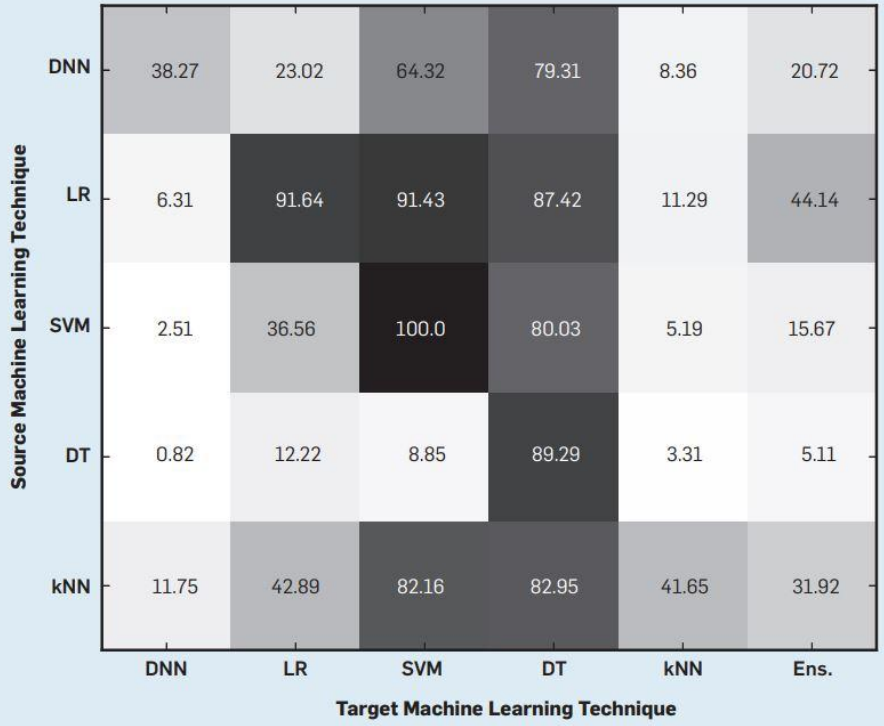
The main challenge resides in selecting the inputs the adversary sends to the API to reduce the total number of queries made - to reduce the detectability of the attacks.

Source: [Making machine learning robust against adversarial inputs](), Goodfellow et al.

Transferability of attacks

Figure 7. Transferability matrix. The source model is used to craft adversarial examples and the target model to predict their class.

The transferability reported is the average rate of mistakes made by the target model on adversarial examples crafted using the source model. The models considered are a deep neural network, logistic regression, support vector machine, decision tree, and *k*-nearest neighbor. The ensemble target model refers to an ensemble making predictions based on a majority vote among the class predicted by each of the five previous models. All adversarial examples were produced with similar perturbation norms.[20]



Source: [Making machine learning robust against adversarial inputs](), Goodfellow et al.

Defenses

1. **Model Training** : Adversarial Training and Defense Distillation
2. **Input Validation and Preprocessing**
3. **Architecture Modification**

- Adversarial training seeks to improve the generalization of a model when presented with adversarial examples at test time by proactively generating adversarial examples as part of the training process.
- Defensive distillation smooth the model’s decision surface in adversarial directions that could be exploited by an adversary.
- Input validation and preprocessing aims to validate and preprocess the input to remove potential adversarial perturbations before it is feed to the machine learning model.
- Architecture modification changes the structure of the machine learning model to defend against adversarial examples.

Reference

1. Making Machine Learning Robust Against Adversarial Inputs, Goodfellow et al.
2. Breaking Deep Learning with Adversarial Examples
3. Explaining and Harnessing Adversarial Examples, Goodfellow et al.
4. Adversarial Examples in the Physical World, Kurakin et al.