Attacks Against Machine Learning

Ref

A Taxonomy and Survey of Attacks Against Machine Learning

[https://gala.gre.ac.uk/id/eprint/25226/7/25226%20LOUKAS_Taxonomy_And_Survey_Of_Attacks_Against_Machine_Learning_%28AAM%29_2019.pdf]

- Machine learning methodologies operate with the assumption that their environment is benign.
- This assumption does not always hold, as it is often advantageous to adversaries to maliciously modify the training (poisoning attacks) or test data (evasion attacks).

- Such attacks can be catastrophic given the growth and the penetration of machine learning applications in society.

Attack Surface

A system built on Machine Learning can be viewed as a generalized data processing pipeline.

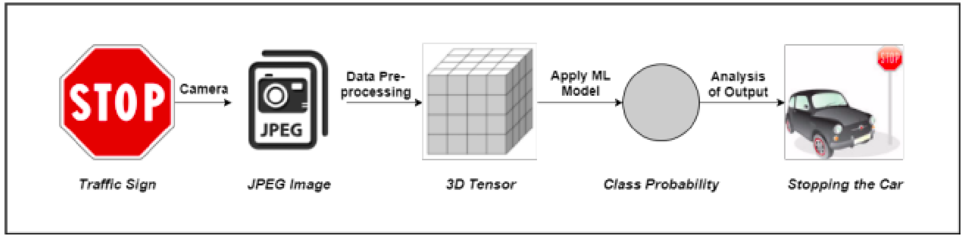
A primitive sequence of operations of the system at the testing time can be viewed as:

a) collection of input data from sensors or data repositories,

b) transferring the data in the digital domain,

c) processing of the transformed data by machine learning model to produce an output,

and finally, d) action taken based on the output.



The system collects sensor inputs (images using camera) from which model features (tensor of

pixel values) are extracted and used within the models.

It then interprets the meaning of the output probability of stop sign), and takes appropriate action (stopping the car).

The attack surface, in this case, can be defined with respect to the data processing pipeline.

An adversary can attempt to manipulate either the collection or the processing of data to corrupt the target model, thus tampering the original output.

Vulnerable Points

Attack models

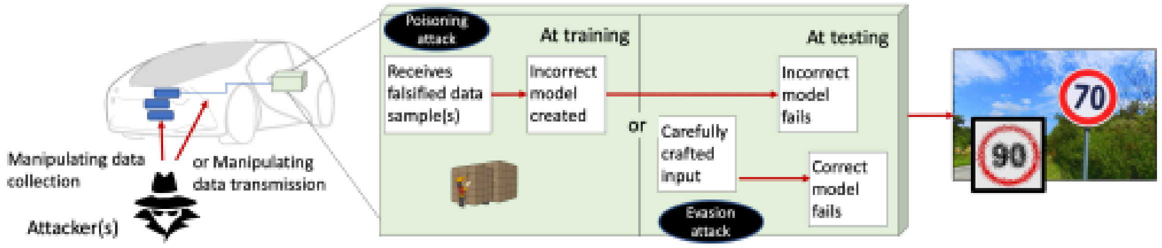


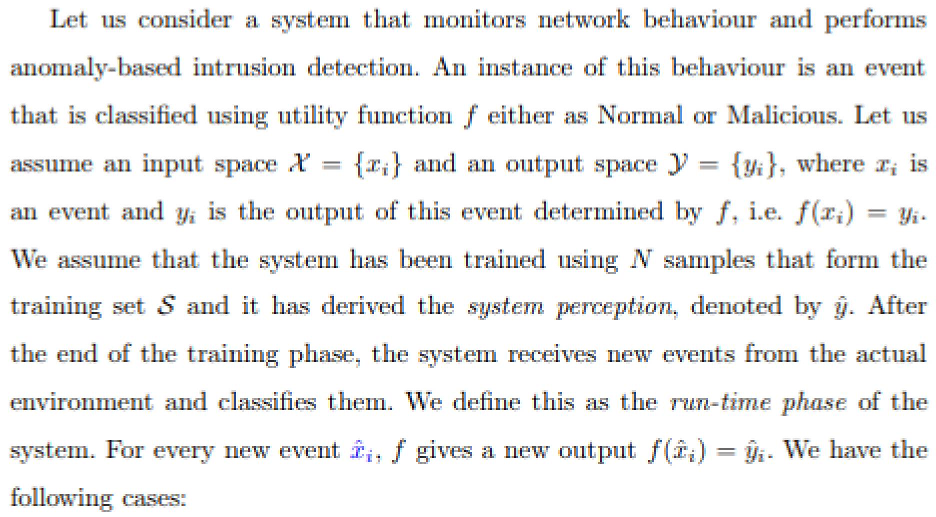
Figure : Different avenues for attacking a machine learning based system

Goals

- system that uses machine learning aims to find a hypothesis function f that maps observable events, into different classes.

- The high level goal of these models is to maximize the **generalization error** of the classification
    - For supervised learning applications in machine learning and statistical learning theory, generalization error (also known as the out-of-sample error or the risk) is **a measure of how accurately an algorithm is able to predict outcome values for previously unseen data**.
- mislead the decision making system towards desired malicious measurement values.

Example



Let us consider a system that monitors network behaviour and performs anomaly-based intrusion detection. An instance of this behaviour is an event that is classified using utility function f either as Normal or Malicious. Let us assume an input space $\mathcal{X} = \{x_i\}$ and an output space $\mathcal{Y} = \{y_i\}$, where x_i is an event and y_i is the output of this event determined by f, i.e. $f(x_i) = y_i$. We assume that the system has been trained using N samples that form the training set $\mathcal{S}$ and it has derived the *system perception*, denoted by $\hat{y}$. After the end of the training phase, the system receives new events from the actual environment and classifies them. We define this as the *run-time phase* of the system. For every new event $\hat{x}_i$, f gives a new output $f(\hat{x}_i) = \hat{y}_i$. We have the following cases:

• If $\hat{x}_i$ is malicious and the system does not recognize it as such (false negative)

there is a loss l caused to the system.

• If $\hat{x}_i$ is malicious and the system recognizes it as such (true positive) or it is not malicious then there is no loss to the system.

• If $\hat{x}_i$ is not malicious and the system recognizes it as such (false positive) then there is a loss λ.

- The aim of the attacker is to maximize the impact the attack has to the system by maximizing $|f(\hat{x}_i)-y_i|$.
- Thus, a challenge of the system that defends is to find a utility function that minimizes the losses, measured as the distance of $f(\hat{x}_i)$ to the real output y_i .
- This function can be linear or nonlinear and be more complex in formulation.

Types of Basic Attacks

Evasion attack

- The adversary can undertake an evasion attack against classification during the testing phase thus producing a wrong system perception.
- **In this case, the goal of the adversary is to achieve misclassification of some data towards, for example, remaining stealthy or mimicking some desirable behaviour.**
- With regards to network anomaly-based detection, an intrusion detection system (IDS) can be evaded by encoding the attack payload in such a way that the destination of the data is able to decode it but the IDS is not leading to a possible misclassification.
- Thus, the attacker can compromise the targeted system not being spotted out by the IDS.

Poisoning attacks

- The adversary can poison the **training dataset.**
- To achieve this, the adversary derives and injects a point to decrease the classification accuracy
- This attack has the ability to completely distort the classification function during its training thus allowing the attacker to define the classification of the system in any way she wishes.
- The magnitude of the classification error depends on the data the attacker has chosen to poison the training.

Poisoning Attacks Examples

- With regards to the aforesaid example:
    - The adversary may be able to create dataset of anomalous network-layer protocol behaviour and train an anomaly-based intrusion detection system with a labelled attack dataset as the groundtruth.
    - As a result, the detector will not be able to recognize cyber attacks against this network-layer protocol threatening the security of the underlying system.
    - This attack could be tailored to also have a significant impact to the quality of a signature-based intrusion detection system, which is responsible, for example, for detecting malware infecting a system or an infrastructure.

Trojan

- **A particularly insidious attack in this category is the backdoor or Trojan attack, where the adversary carefully poisons the model by inserting a backdoor key to ensure it will perform well on standard training data and validation samples, but misbehaves only when a backdoor key is present.**
  - Thus an attacker can selectively make a model misbehave by introducing backdoor keys once the model is deployed.
  - For instance, consider the case of assistive driving in autonomous vehicles:
    - a backdoor could cause the model to misclassify a stop sign as speed limit whenever a specific mark has been placed on the stop sign.
    - However, the model would perform as expected on stop signs without this mark, making the backdoor difficult to detect since users do not know the backdoor key a priori.

Types

(1) **Evasion Attack:** This is the most common type of attack in the adversarial setting. The adversary tries to **evade the system by adjusting malicious samples during testing phase.** This setting does not assume any influence over the training data.

(2) **Poisoning Attack:** This type of attack, known as contamination of the training data, takes place during the training time of the machine learning model. An adversary **tries to poison the training data by injecting carefully designed samples** to compromise the whole learning process eventually.

(3) **Exploratory Attack**: These attacks do not influence training dataset. Given **black box access to the model**, they try to gain as much knowledge as possible about the learning algorithm of the underlying system and pattern in training data.

Attack Steps

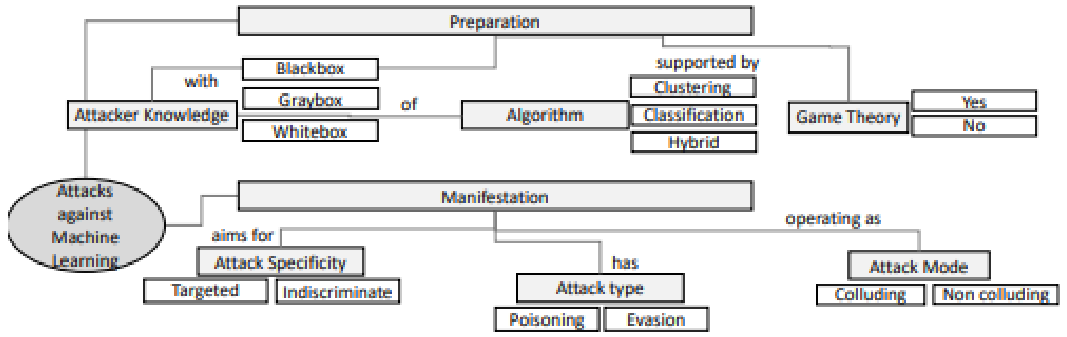


Figure A taxonomy of adversarial attacks on machine learning.

Preparation

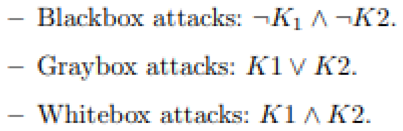
- In this phase, the attackers identify their resources and gather the intelligence required to prepare an attack plan.

- Here, what determines the characteristics of an adversarial machine learning approach is the **knowledge required by the attacker**, as well as the **type of machine learning** technique targeted and whether the attacker is **strategic**, i.e. they use game-theoretic techniques.

Features for Preparation

Attacker Knowledge

Here, we take the simplified view whereby the attacker may know (K1) the Ground truth, (K2) the learning algorithm, or both, leading to the following attacker knowledge categories:



- Blackbox attacks: $\neg K_1 \land \neg K2$.
- Graybox attacks: $K1 \lor K2$.
- Whitebox attacks: $K1 \land K2$.

The attacker knowledge may refer to (i) the training data, (ii) the feature set, (iii) the machine learning algorithm along with the objective function minimized during training and (iv) any trained parameters if applicable

Let us consider a target machine learning model f is trained over input pair (X,y) from the data distribution μ with a randomized training procedure train having randomness r (e.g., random weight initialization, dropout, etc.).

The model parameters θ are learned after the training procedure.
More formally, we can write: $\theta \leftarrow \text{train}(f\ ,X,y,r)$

Now, let us understand the capabilities of the white-box and black-box adversaries with respect to this definition.

White-Box Attacks

- In white-box attack on a machine learning model, an adversary has total **knowledge about the model (f)** used for classification (e.g., type of neural network along with number of layers).
- The attacker has information about the **algorithm (train) used in training** (e.g., gradient-descent optimization) and can access the training data distribution (μ).
- He **also knows the parameters (θ) of the fully trained model architecture**.
- The **adversary utilizes available information to identify the feature space where the model may be vulnerable, i.e, for which the model has a high error rate.**
- Then the model is exploited by altering an input using adversarial example
- crafting method, which we discuss later.
- **The access to internal model weights for a white-box attack corresponds to a very strong adversarial attack.**

Black-Box Attacks

- assumes no knowledge about the model
- uses information about the settings or past inputs to analyse the vulnerability of the model.
- [For example, in an oracle attack, the adversary exploits a model by providing a series of carefully crafted inputs and observing outputs.]
- Black Box attacks can be further classified.

Non-Adaptive Black-Box Attack: For a target model (f), a non-adaptive black-box adversary only gets access to the target model’s training data distribution (μ).

The adversary then chooses a procedure to train a model architecture f′ and trains a local model over samples from the data distribution μ to approximate the model learned by the target classifier.

Adaptive Black-Box Attack:

- For a target model (f), an adaptive black-box adversary does not have any information regarding the training process but can access the target model as an oracle.
- The adversary issues adaptive oracle queries to the target model and labels a carefully selected dataset, i.e.,for any arbitrarily chosen x the adversary obtains its label y by querying the target model f .
- The adversary then chooses a procedure to train and model architecture f to train a surrogate
- model over tuples (x,y) obtained from querying the target model.
- The surrogate model then produces adversarial samples by following white-box attack technique for forcing the target model to mis-classify malicious data.

Strict Black-Box Attack:

A black-box adversary sometimes may not contain the data distribution μ but has the ability to collect the input-output pairs (x,y) from the target classifier.

However, he can not change the inputs to observe the changes in output like an adaptive attack procedure.

Description	**Black box attack**	**White box attack**
Adversary Knowledge	Restricted knowledge from being able to only observe the networks output on some probed inputs.	Detailed knowledge of the network architecture and the parameters resulting from training.
Attack Strategy	Based on a greedy local search generating an implicit approximation to the actual gradient w.r.t the current output by observing changes in input.	Based on the gradient of the network loss function w.r.t to the input.

Adversarial Capabilities

- The term adversarial capabilities refer to the amount of information available to an adversary about the system, which also indicates the attack vector he may use on the threat surface.

- An internal adversary is one who have access to the model architecture and can use it to distinguish between different images and traffic signs.

- A weaker adversary is one who have access only to the dump of images fed to the model during testing time.
- Though both the adversaries are working on the same attack surface, the former adversary is assumed to have much more information and is thus strictly “stronger”.

Training Phase Capabilities

. Attacks during training time attempt to influence or corrupt the training dataset

(1) **Data Injection**: The adversary does not have any access to the training data as well as to the learning algorithm but has ability to augment a new data to the training set.

-can corrupt the target model by inserting adversarial samples into the training dataset.

(2) **Data Modification**: The adversary does not have access to the learning algorithm but has full access to the training data. He poisons the training data directly by modifying the data

before it is used for training the target model.

(3) **Logic Corruption**: The adversary has the ability to meddle with the learning algorithm.

Testing Phase Capabilities

Adversarial attacks at the testing time **do not tamper with the targeted model** but rather forces it to produce incorrect outputs.

The effectiveness of such attacks is determined mainly by the amount of information available to the adversary about the model.

Testing phase attacks can be broadly classified into either White-Box or Black-Box attacks.

Variety in Algorithm

- A large variety of machine learning techniques has been targeted in the literature:
  - DNNs and Convolutional Neural Networks (CNNs) are commonly addressed in the image recognition domain
  - Spam email detection, more common are Naive Bayes, Support Vector Machines (SVM) and Logistic Regression (LR).
  - K-Means, K-Nearest Neighbour (KNN), Linear Regression, Community Discovery and Singular Value Decomposition, are typically seen in the malware detection, biometric recognition and network failure and security breach detection domains.
- Classified the techniques in: i) clustering, ii) classification, or iii) hybrid fashion.

Manifestation:

- This is the phase where the adversary launches the attack against the machine learning system.
- Largely dependent on the intelligence gathered in the preparation phase

Features for Manifestation

Attack Specificity/ Error specificity

This refers to range of data points that are targeted by the attacker

- Targeted: The focus of the attack is on a particular sample (e.g., specific spam email misclassified as legitimate)
- Indiscriminate (a small set of samples): The adversary attacks a very general class of samples, such as “any false negative” (e.g., maximizing the percentage of spam emails misclassified as legitimate).

Attack Type

This refers to how the machine learning system is affected by an attack

- Poisoning: Poisoning attacks alter the training process through influence over the training data.
- Evasion: Evasion attacks exploit misclassifications but do not affect training (e.g. the learner or offline analysis, to discover information)

Attack Mode

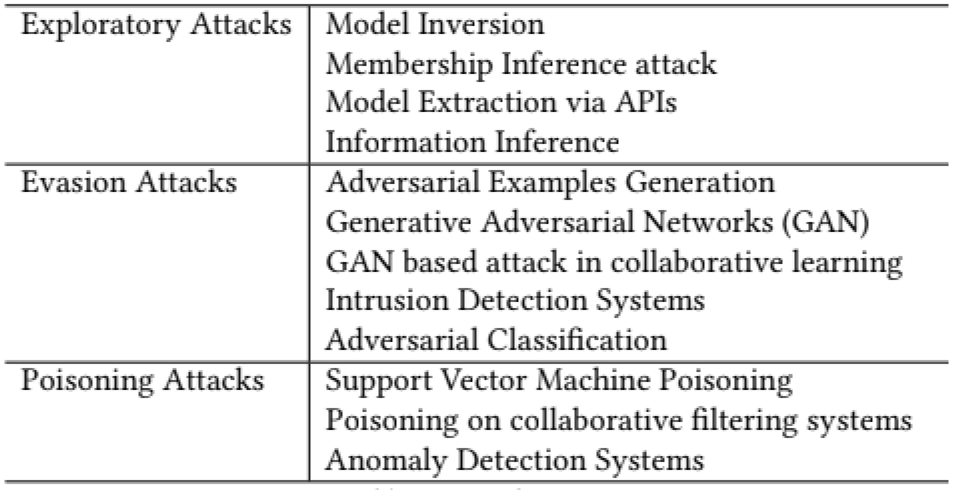
The original assumption of adversarial machine learning, which is still taken in most related literature, is that **attackers work on their own (non-colluding case)**.

The alternative is that different **colluding attackers can collaborate**, not only to cover their tracks but also to increase efficiency.

Attack Evaluation

- The output of an attack's manifestation is primarily characterized by the nature of its impact on the accuracy of a machine learning approach.

Attacks in Detail



Exploratory Attacks	Model Inversion
	Membership Inference attack
	Model Extraction via APIs
	Information Inference
Evasion Attacks	Adversarial Examples Generation
	Generative Adversarial Networks (GAN)
	GAN based attack in collaborative learning
	Intrusion Detection Systems
	Adversarial Classification
Poisoning Attacks	Support Vector Machine Poisoning
	Poisoning on collaborative filtering systems
	Anomaly Detection Systems

EVASION & POISONING ATTACKS

Evasion attacks are the most common attacks on machine learning systems. Malicious inputs are craftily modified so as to force the model to make a false prediction and evade detection.

Poisoning attack differs in that the inputs are modified during training and model is trained on contaminated inputs to obtain desired output.

Performance impact

- **The primary aim of adversarial machine learning is to reduce the performance of a classification or clustering process that is based on machine learning.**
- For classification problems: this can be interpreted as **increase in false positives, in false negatives, or in both.**
    - [spam detection, where there are two states (spam or normal), the aim of an attacker may be to make the targeted system falsely label many normal emails as spam emails. This would lead to the user missing emails. – False negatives:]
- For **clustering problems, the aim is generally to reduce accuracy. – False positives.**
    - Using the same example, if the attacker aims to increase the false negatives, then many spam emails would go through the user's filters. – Both false positives and false negatives: Here, the attacker aims to reduce the overall confidence of the user in their spam filtering system by letting spam emails go through and by filtering out normal emails
- Compared to classification, the accuracy of clustering is less straight-forward to evaluate.

Evasion Attack on SVM

Ref: https://arxiv.org/pdf/1401.7727.pdf

Algorithm 1 Gradient-descent attack procedure

Input: the initial attack point, $\mathbf{x}^0$; the step size, t; the trade-off parameter, λ; and $\varepsilon > 0$.
Output: $\mathbf{x}^*$, the final attack point.

```
 1: $k \leftarrow 0$.
 2: repeat
 3:     $k \leftarrow k + 1$
 4:     Set $\nabla E(\mathbf{x}^{k-1})$ to a unit vector aligned with $\nabla g(\mathbf{x}^{k-1}) - \lambda \nabla p(\mathbf{x}^{k-1} | f_{\mathbf{x}} = -1)$.
 5:     $\mathbf{x}^k \leftarrow \mathbf{x}^{k-1} - t \nabla E(\mathbf{x}^{k-1})$
 6:     if $d(\mathbf{x}^k, \mathbf{x}^0) > d_{\max}$ then
 7:         Project $\mathbf{x}^k$ onto the boundary of the feasible region (enforcing application-specific con-
            straints, if any).
 8:     end if
 9: until $E\left(\mathbf{x}^k\right) - E\left(\mathbf{x}^{k-1}\right) < \varepsilon$
10: return: $\mathbf{x}^* = \mathbf{x}^k$
```

- consider the problem of SVM evasion at test time; i.e., how to optimally manipulate samples at test time to avoid detection.
- evasion of kernel-based classifiers at test time can be realized with a straightforward gradient-descent-based approach.
- even if the adversary does not precisely know the classifier’s decision function, she can learn a surrogate classifier on a surrogate dataset and reliably evade the targeted classifier

Notation. We consider a classification algorithm $f : \mathscr{X} \mapsto \mathscr{Y}$ that assigns samples represented in some feature space $\mathbf{x} \in \mathscr{X}$ to a label in the set of predefined classes $y \in \mathscr{Y} = \{-1, +1\}$, where -1 ($+1$) represents the legitimate (malicious) class. The label $f_{\mathbf{x}} = f(\mathbf{x})$ given by a classifier is typically obtained by thresholding a continuous discriminant function $g : \mathscr{X} \mapsto \mathbb{R}$. Without loss of generality, we assume that $f(\mathbf{x}) = -1$ if $g(\mathbf{x}) < 0$, and $+1$ otherwise. Further, note that we use $f_{\mathbf{x}}$ to refer to a label assigned by the classifier for the point $\mathbf{x}$ (rather than the true label y of that point) and the shorthand f_i for the label assigned to the i^{th} training point, $\mathbf{x}_i$.

Adversary’s Knowledge:

first, the adversary has perfect knowledge (PK) of the targeted classifier; i.e., she knows the feature space and function g(x).

In the second, the adversary is assumed to have limited knowledge (LK) of the classifier: knows the feature representation and the learning algorithm, but that she does not know the learned classifier g(x).

In both cases, we assume the attacker does not have knowledge of the training set (k.i).

Surrogate classifier

LK scenario: the adversary does not know the true discriminant function g(x) but may approximate it by learning.

[may sniff network traffic or collect legitimate and spam emails from an alternate source.]

Two sub-cases: which depend on whether the adversary can query the classifier.

- If so, the adversary can build the training set by submitting a set of nq queries xi to the targeted classifier to obtain their classification labels, yi = f(xi).

- requires to have access to classifier feedback;

[e.g., by having an email account protected by the targeted filter (for public email providers, the adversary can reasonably obtain such accounts). If not, the adversary may use the true class labels for the surrogate data, although this may not correctly approximate the targeted classifier.]

Adversary’s Capability

- the adversary can only manipulate testing data, no way to influence training data.

- The adversary’s capability of manipulating the features of each sample should be defined based on application-specific constraints.
- However, at a **more general level we can bound the attack point to lie within some maximum distance from the original attack sample, dmax, which then is a parameter of our evaluation**.

Attack Strategy

Consider malicious sample x^0 (the adversary's true objective).

an optimal attack strategy finds a sample x^∗ to minimize g or its estimate g^, subject to a bound on its modification distance from x^0:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \hat{g}(\mathbf{x}) \quad \text{s.t.} \quad d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max}$$

For several classifiers, minimizing g(x) is equivalent to maximizing the estimated posterior $p(f_{\mathbf{x}} = -1|\mathbf{x})$

Limitations

1. Generally, this is a non-linear optimization, which one may optimize with many well-known techniques (e.g., gradient descent, Newton’s method, or BFGS)

[An optimization problem is nonlinear **if the objective function f(x) or any of the inequality constraints $c_i(x) \leq 0$, i = 1, 2, …, m, or equality constraints $d_j(x) = 0$, j = 1, 2, …, n, are nonlinear functions of the vector of variables x**.]

1. if $\hat{g}(x)$ is not convex, descent approaches may not find a global optima:
   - Instead, the descent path may lead to a flat region (local minimum) outside of the samples’ support where $p(x) \approx 0$
     - the classification behavior of g is unspecified and may stymie evasion attempts

Remove False Evasion Points

- introduce an additional component into the formulation of attack objective
- That estimates p(x| fx = −1) using density-estimation techniques.
- This second component acts as a penalizer for x in low density regions and is weighted by a parameter λ ≥ 0 yielding the following modified optimization problem:

$$\arg\min_{\mathbf{x}} \; E(\mathbf{x}) = \hat{g}(\mathbf{x}) - \frac{\lambda}{n} \sum_{i|f_i=-1} k\left(\tfrac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$

$$\text{s.t.} \;\; d(\mathbf{x},\mathbf{x}^0) \leq d_{\max} \;\; ,$$

$$\arg\min_{\mathbf{x}} \; E(\mathbf{x}) = \hat{g}(\mathbf{x}) - \frac{\lambda}{n} \sum_{i|f_i=-1} k\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$
$$\text{s.t.} \;\; d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max} \;\; ,$$

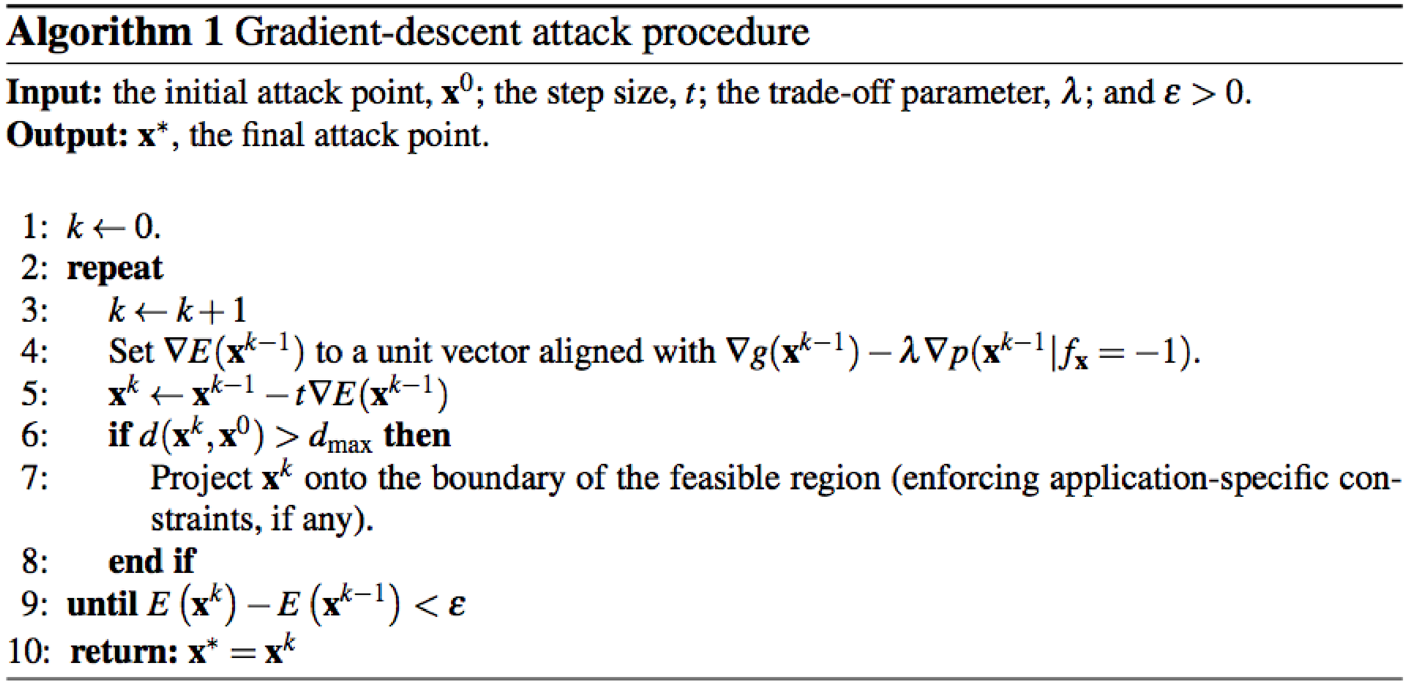
where h is a bandwidth parameter for a kernel density estimator (KDE),

and n is the number of benign samples (fx = −1) available to the adversary.

[non-parametric method to estimate the probability density function of a random variable based on kernels as weights.]

alternate objective trades off between minimizing g^(x) (or p(fx = −1|x)) and maximizing the estimated density p(x| fx = −1).

- **The extra component favors attack points to imitate features of known samples classified as legitimate.**
- **In doing so, it reshapes the objective function and thereby biases the resulting density augmented gradient descent towards regions where the negative class is concentrated.**
- **this behavior may lead our technique to disregard attack patterns within unsupported regions ($p(x) \approx 0$) for which $g(x) < 0$.**



Algorithm 1 Gradient-descent attack procedure

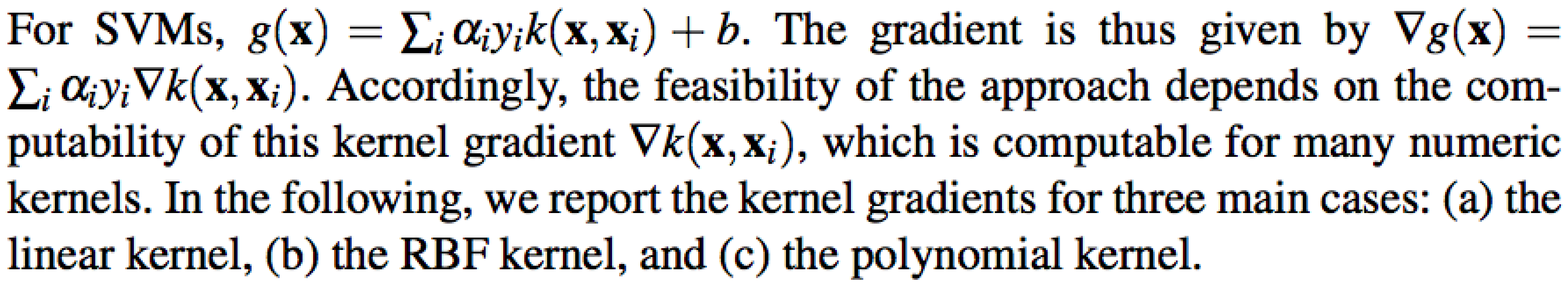
Input: the initial attack point, $\mathbf{x}^0$; the step size, t; the trade-off parameter, λ; and $\varepsilon > 0$.
Output: $\mathbf{x}^*$, the final attack point.

```
 1: $k \leftarrow 0$.
 2: repeat
 3:     $k \leftarrow k+1$
 4:     Set $\nabla E(\mathbf{x}^{k-1})$ to a unit vector aligned with $\nabla g(\mathbf{x}^{k-1}) - \lambda \nabla p(\mathbf{x}^{k-1} | f_{\mathbf{x}} = -1)$.
 5:     $\mathbf{x}^k \leftarrow \mathbf{x}^{k-1} - t \nabla E(\mathbf{x}^{k-1})$
 6:     if $d(\mathbf{x}^k, \mathbf{x}^0) > d_{\max}$ then
 7:         Project $\mathbf{x}^k$ onto the boundary of the feasible region (enforcing application-specific con-
            straints, if any).
 8:     end if
 9: until $E\left(\mathbf{x}^k\right) - E\left(\mathbf{x}^{k-1}\right) < \varepsilon$
10: return: $\mathbf{x}^* = \mathbf{x}^k$
```

- iteratively modifies the attack point x in the feature space as x 0 ← x−t∇E
- ∇E is a unit vector aligned with the gradient of our objective function, and t is the step size

How to project the gradient ∇E?

Gradient of Support Vector Machines:



For SVMs, $g(\mathbf{x}) = \sum_i \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b$. The gradient is thus given by $\nabla g(\mathbf{x}) = \sum_i \alpha_i y_i \nabla k(\mathbf{x}, \mathbf{x}_i)$. Accordingly, the feasibility of the approach depends on the computability of this kernel gradient $\nabla k(\mathbf{x}, \mathbf{x}_i)$, which is computable for many numeric kernels. In the following, we report the kernel gradients for three main cases: (a) the linear kernel, (b) the RBF kernel, and (c) the polynomial kernel.

Primal and Dual Form Ref: https://www.robots.ox.ac.uk/~az/lectures/ml/lect3.pdf

(a) Linear kernel. In this case, the kernel is simply given by $k(\mathbf{x}, \mathbf{x}_i) = \langle \mathbf{x}, \mathbf{x}_i \rangle$. Accordingly, $\nabla k(\mathbf{x}, \mathbf{x}_i) = \mathbf{x}_i$ (we remind the reader that the gradient has to be computed with respect to the current attack sample $\mathbf{x}$), and $\nabla g(\mathbf{x}) = \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$.

(b) RBF kernel. For this kernel, $k(\mathbf{x}, \mathbf{x}_i) = \exp\{-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\}$. The gradient is thus given by $\nabla k(\mathbf{x}, \mathbf{x}_i) = -2\gamma \exp\{-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\}(\mathbf{x} - \mathbf{x}_i)$.

(c) Polynomial kernel. In this final case, $k(\mathbf{x}, \mathbf{x}_i) = (\langle \mathbf{x}, \mathbf{x}_i \rangle + c)^p$. The gradient is thus given by $\nabla k(\mathbf{x}, \mathbf{x}_i) = p(\langle \mathbf{x}, \mathbf{x}_i \rangle + c)^{p-1} \mathbf{x}_i$.

Class Assignment

Verify for:

i) RBF Kernel

ii) Polynomial Kernel

Gradient Descent Attack in Discrete Spaces

In discrete spaces, gradient approaches **may lead to a path through infeasible portions of the feature space**.

In such cases, we need to **find feasible neighbors x** that yield a steepest descent; i.e., maximally decreasing E(x).

$$\mathbf{x}' \leftarrow \arg\max_{\mathbf{z}\in\mathcal{N}(\mathbf{x})} \frac{(\mathbf{z}-\mathbf{x})}{\|\mathbf{z}-\mathbf{x}\|}^{\top} \nabla E(\mathbf{x})$$