Title: Adversarial machine learning

URL: https://en.wikipedia.org/wiki/Adversarial_machine_learning

PageID: 45049676

Categories: Category:AI safety, Category:Computer security, Category:Machine learning

-----

Supervised learning

Unsupervised learning

Semi-supervised learning

Self-supervised learning

Reinforcement learning

Meta-learning

Online learning

Batch learning

Curriculum learning

Rule-based learning

Neuro-symbolic AI

Neuromorphic engineering

Quantum machine learning

Classification

Generative modeling

Regression

Clustering

Dimensionality reduction

Density estimation

Anomaly detection

Data cleaning

AutoML

Association rules

Semantic analysis

Structured prediction

Feature engineering

Feature learning

Learning to rank

Grammar induction

Ontology learning

Multimodal learning

Apprenticeship learning

Decision trees

Ensembles Bagging Boosting Random forest

Bagging

Boosting

Random forest

k -NN

Linear regression

Naive Bayes

Artificial neural networks

Logistic regression

Perceptron

Relevance vector machine (RVM)

Support vector machine (SVM)

BIRCH

CURE

Hierarchical

k -means

Fuzzy

Expectation–maximization (EM)

DBSCAN

OPTICS

Mean shift

Factor analysis

CCA

ICA

LDA

NMF

PCA

PGD

t-SNE

SDL

Graphical models Bayes net Conditional random field Hidden Markov

Bayes net

Conditional random field

Hidden Markov

RANSAC

k -NN

Local outlier factor

Isolation forest

Autoencoder

Deep learning

Feedforward neural network

Recurrent neural network LSTM GRU ESN reservoir computing

LSTM

GRU

ESN

reservoir computing

Boltzmann machine Restricted

Restricted

GAN

Diffusion model

SOM

Convolutional neural network U-Net LeNet AlexNet DeepDream

U-Net

LeNet

AlexNet

DeepDream

Neural field Neural radiance field Physics-informed neural networks

Neural radiance field

Physics-informed neural networks

Transformer Vision

Vision

Mamba

Spiking neural network

Memtransistor

Electrochemical RAM (ECRAM)

Q-learning

Policy gradient

SARSA

Temporal difference (TD)

Multi-agent Self-play

Self-play

Active learning

Crowdsourcing

Human-in-the-loop

Mechanistic interpretability

RLHF

Coefficient of determination

Confusion matrix

Learning curve

ROC curve

Kernel machines

Bias–variance tradeoff

Computational learning theory

Empirical risk minimization

Occam learning

PAC learning

Statistical learning

VC theory

Topological deep learning

AAAI

ECML PKDD

NeurIPS

ICML

ICLR

IJCAI

ML

JMLR

Glossary of artificial intelligence

List of datasets for machine-learning research List of datasets in computer vision and image processing

List of datasets in computer vision and image processing

Outline of machine learning

v

t

e

Adversarial machine learning is the study of the attacks on machine learning algorithms, and of the defenses against such attacks. A survey from May 2020 revealed practitioners' common feeling for better protection of machine learning systems in industrial applications.

Machine learning techniques are mostly designed to work on specific problem sets, under the assumption that the training and test data are generated from the same statistical distribution (IID). However, this assumption is often dangerously violated in practical high-stake applications, where users may intentionally supply fabricated data that violates the statistical assumption.

Most common attacks in adversarial machine learning include evasion attacks , data poisoning attacks , Byzantine attacks and model extraction.

History

At the MIT Spam Conference in January 2004, John Graham-Cumming showed that a machine-learning spam filter could be used to defeat another machine-learning spam filter by automatically learning which words to add to a spam email to get the email classified as not spam.

In 2004, Nilesh Dalvi and others noted that linear classifiers used in spam filters could be defeated by simple " evasion attacks" as spammers inserted "good words" into their spam emails. (Around 2007, some spammers added random noise to fuzz words within "image spam" in order to defeat OCR -based filters.) In 2006, Marco Barreno and others published "Can Machine Learning Be Secure?", outlining a broad taxonomy of attacks. As late as 2013 many researchers continued to hope that non-linear classifiers (such as support vector machines and neural networks) might be robust to adversaries, until Battista Biggio and others demonstrated the first gradient-based attacks on such machine-learning models (2012 –2013). In 2012, deep neural networks began to dominate computer vision problems; starting in 2014, Christian Szegedy and others demonstrated that deep neural networks could be fooled by adversaries, again using a gradient-based attack to craft adversarial perturbations.

Recently, it was observed that adversarial attacks are harder to produce in the practical world due to the different environmental constraints that cancel out the effect of noise. For example, any small rotation or slight illumination on an adversarial image can destroy the adversariality. In addition, researchers such as Google Brain's Nick Frosst point out that it is much easier to make self-driving cars miss stop signs by physically removing the sign itself, rather than creating adversarial examples. Frosst also believes that the adversarial machine learning community incorrectly assumes models trained on a certain data distribution will also perform well on a completely different data distribution. He suggests that a new approach to machine learning should be explored, and is currently working on a unique neural network that has characteristics more similar to human perception than state-of-the-art approaches.

While adversarial machine learning continues to be heavily rooted in academia, large tech companies such as Google, Microsoft, and IBM have begun curating documentation and open source code bases to allow others to concretely assess the robustness of machine learning models and minimize the risk of adversarial attacks.

Examples

Examples include attacks in spam filtering , where spam messages are obfuscated through the misspelling of "bad" words or the insertion of "good" words; attacks in computer security , such as obfuscating malware code within network packets or modifying the characteristics of a network flow to mislead intrusion detection; attacks in biometric recognition where fake biometric traits may be exploited to impersonate a legitimate user; or to compromise users' template galleries that adapt to updated traits over time.

Researchers showed that by changing only one-pixel it was possible to fool deep learning algorithms. Others 3-D printed a toy turtle with a texture engineered to make Google's object detection AI classify it as a rifle regardless of the angle from which the turtle was viewed. Creating the turtle required only low-cost commercially available 3-D printing technology.

A machine-tweaked image of a dog was shown to look like a cat to both computers and humans. A 2019 study reported that humans can guess how machines will classify adversarial images. Researchers discovered methods for perturbing the appearance of a stop sign such that an autonomous vehicle classified it as a merge or speed limit sign.

A data poisoning filter called Nightshade was released in 2023 by researchers at the University of Chicago . It was created for use by visual artists to put on their artwork to corrupt the data set of text-to-image models , which usually scrape their data from the internet without the consent of the image creator.

McAfee attacked Tesla 's former Mobileye system, fooling it into driving 50 mph over the speed limit, simply by adding a two-inch strip of black tape to a speed limit sign.

Adversarial patterns on glasses or clothing designed to deceive facial-recognition systems or license-plate readers, have led to a niche industry of "stealth streetwear".

An adversarial attack on a neural network can allow an attacker to inject algorithms into the target system. Researchers can also create adversarial audio inputs to disguise commands to intelligent assistants in benign-seeming audio; a parallel literature explores human perception of such stimuli.

Clustering algorithms are used in security applications. Malware and computer virus analysis aims to identify malware families, and to generate specific detection signatures.

Attack modalities

Taxonomy

Attacks against (supervised) machine learning algorithms have been categorized along three primary axes: influence on the classifier, the security violation and their specificity.

Classifier influence: An attack can influence the classifier by disrupting the classification phase. This may be preceded by an exploration phase to identify vulnerabilities. The attacker's capabilities might be restricted by the presence of data manipulation constraints.

Security violation: An attack can supply malicious data that gets classified as legitimate. Malicious data supplied during training can cause legitimate data to be rejected after training.

Specificity: A targeted attack attempts to allow a specific intrusion/disruption. Alternatively, an indiscriminate attack creates general mayhem.

This taxonomy has been extended into a more comprehensive threat model that allows explicit assumptions about the adversary's goal, knowledge of the attacked system, capability of manipulating the input data/system components, and on attack strategy. This taxonomy has further been extended to include dimensions for defense strategies against adversarial attacks.

Strategies

Below are some of the most commonly encountered attack scenarios.

Data poisoning

Poisoning consists of contaminating the training dataset with data designed to increase errors in the output. Given that learning algorithms are shaped by their training datasets, poisoning can effectively reprogram algorithms with potentially malicious intent. Concerns have been raised especially for user-generated training data, e.g. for content recommendation or natural language models. The ubiquity of fake accounts offers many opportunities for poisoning. Facebook reportedly removes around 7 billion fake accounts per year. Poisoning has been reported as the leading concern for industrial applications.

On social medias, disinformation campaigns attempt to bias recommendation and moderation algorithms, to push certain content over others.

A particular case of data poisoning is the backdoor attack, which aims to teach a specific behavior for inputs with a given trigger, e.g. a small defect on images, sounds, videos or texts.

For instance, intrusion detection systems are often trained using collected data. An attacker may poison this data by injecting malicious samples during operation that subsequently disrupt retraining.

Data poisoning techniques can also be applied to text-to-image models to alter their output, which is used by artists to defend their copyrighted works or their artistic style against imitation.

Data poisoning can also happen unintentionally through model collapse , where models are trained on synthetic data.

Byzantine attacks

As machine learning is scaled, it often relies on multiple computing machines. In federated learning , for instance, edge devices collaborate with a central server, typically by sending gradients or model parameters. However, some of these devices may deviate from their expected behavior, e.g.

to harm the central server's model or to bias algorithms towards certain behaviors (e.g., amplifying the recommendation of disinformation content). On the other hand, if the training is performed on a single machine, then the model is very vulnerable to a failure of the machine, or an attack on the machine; the machine is a single point of failure . In fact, the machine owner may themselves insert provably undetectable backdoors .

The current leading solutions to make (distributed) learning algorithms provably resilient to a minority of malicious (a.k.a. Byzantine) participants are based on robust gradient aggregation rules. The robust aggregation rules do not always work especially when the data across participants has a non-iid distribution. Nevertheless, in the context of heterogeneous honest participants, such as users with different consumption habits for recommendation algorithms or writing styles for language models, there are provable impossibility theorems on what any robust learning algorithm can guarantee.

Evasion

Evasion attacks consist of exploiting the imperfection of a trained model. For instance, spammers and hackers often attempt to evade detection by obfuscating the content of spam emails and malware . Samples are modified to evade detection; that is, to be classified as legitimate. This does not involve influence over the training data. A clear example of evasion is image-based spam in which the spam content is embedded within an attached image to evade textual analysis by anti-spam filters. Another example of evasion is given by spoofing attacks against biometric verification systems.

Evasion attacks can be generally split into two different categories: black box attacks and white box attacks .

Model extraction

Model extraction involves an adversary probing a black box machine learning system in order to extract the data it was trained on. This can cause issues when either the training data or the model itself is sensitive and confidential. For example, model extraction could be used to extract a proprietary stock trading model which the adversary could then use for their own financial benefit.

In the extreme case, model extraction can lead to model stealing , which corresponds to extracting a sufficient amount of data from the model to enable the complete reconstruction of the model.

On the other hand, membership inference is a targeted model extraction attack, which infers the owner of a data point, often by leveraging the overfitting resulting from poor machine learning practices. Concerningly, this is sometimes achievable even without knowledge or access to a target model's parameters, raising security concerns for models trained on sensitive data, including but not limited to medical records and/or personally identifiable information. With the emergence of transfer learning and public accessibility of many state of the art machine learning models, tech companies are increasingly drawn to create models based on public ones, giving attackers freely accessible information to the structure and type of model being used.

Categories

Adversarial attacks and training in linear models

There is a growing literature about adversarial attacks in

linear models. Indeed, since the seminal work from Goodfellow at al. studying these models in linear models has been an important tool to understand how adversarial attacks affect machine learning models.

The analysis of these models is simplified because the computation of adversarial attacks can be simplified in linear regression and classification problems. Moreover, adversarial training is convex in this case.

Linear models allow for analytical analysis while still reproducing phenomena observed in state-of-the-art models.

One prime example of that is how this model can be used to explain the trade-off between robustness and accuracy. Diverse work indeed provides analysis of adversarial attacks in linear models, including asymptotic analysis for classification and for linear regression. And, finite-sample analysis based on Rademacher complexity.

A result from studying adversarial attacks in linear models is that it closely relates to regularization . Under certain conditions, it has been shown that

adversarial training of a linear regression model with input perturbations restricted by the infinity-norm closely resembles Lasso regression, and that

adversarial training of a linear regression model with input perturbations restricted by the 2-norm closely resembles Ridge regression .

Adversarial deep reinforcement learning

Adversarial deep reinforcement learning is an active area of research in reinforcement learning focusing on vulnerabilities of learned policies. In this research area, some studies initially showed that reinforcement learning policies are susceptible to imperceptible adversarial manipulations. While some methods have been proposed to overcome these susceptibilities, in the most recent studies it has been shown that these proposed solutions are far from providing an accurate representation of current vulnerabilities of deep reinforcement learning policies.

Adversarial natural language processing

Adversarial attacks on speech recognition have been introduced for speech-to-text applications, in particular for Mozilla's implementation of DeepSpeech.

Specific attack types

There are a large variety of different adversarial attacks that can be used against machine learning systems. Many of these work on both deep learning systems as well as traditional machine learning models such as SVMs and linear regression . A high level sample of these attack types include:

Adversarial Examples

Trojan Attacks / Backdoor Attacks

Model Inversion

Membership Inference

Adversarial examples

An adversarial example refers to specially crafted input that is designed to look "normal" to humans but causes misclassification to a machine learning model. Often, a form of specially designed "noise" is used to elicit the misclassifications. Below are some current techniques for generating adversarial examples in the literature (by no means an exhaustive list).

Gradient-based evasion attack

Fast Gradient Sign Method (FGSM)

Projected Gradient Descent (PGD)

Carlini and Wagner (C&W;) attack

Adversarial patch attack

Black box attacks

Black box attacks in adversarial machine learning assume that the adversary can only get outputs for provided inputs and has no knowledge of the model structure or parameters. In this case, the adversarial example is generated either using a model created from scratch, or without any model at all (excluding the ability to query the original model). In either case, the objective of these attacks is to create adversarial examples that are able to transfer to the black box model in question.

Simple Black-box Adversarial Attacks is a query-efficient way to attack black-box image classifiers.

Take a random orthonormal basis v 1 , v 2 , … , v d {\displaystyle v_{1},v_{2},\dots ,v_{d}} in R d {\displaystyle \mathbb {R} ^{d}} . The authors suggested the discrete cosine transform of the standard basis (the pixels).

For a correctly classified image x {\displaystyle x} , try x + ■ v 1 , x − ■ v 1 {\displaystyle x+\epsilon v_{1},x-\epsilon v_{1}} , and compare the amount of error in the classifier upon x + ■ v 1 , x , x − ■ v 1 {\displaystyle x+\epsilon v_{1},x,x-\epsilon v_{1}} . Pick the one that causes the largest amount of error.

Repeat this for v 2 , v 3 , … {\displaystyle v_{2},v_{3},\dots } until the desired level of error in the classifier is reached.

It was discovered when the authors designed a simple baseline to compare with a previous black-box adversarial attack algorithm based on gaussian processes , and were surprised that the baseline worked even better.

The Square Attack was introduced in 2020 as a black box evasion adversarial attack based on querying classification scores without the need of gradient information. As a score based black box attack, this adversarial approach is able to query probability distributions across model output classes, but has no other access to the model itself. According to the paper's authors, the proposed Square Attack required fewer queries than when compared to state-of-the-art score-based black box attacks at the time.

To describe the function objective, the attack defines the classifier as f : [0 , 1] d → R K {\textstyle f:[0,1]^{d}\rightarrow \mathbb {R} ^{K}} , with d {\textstyle d} representing the dimensions of the input and K {\textstyle K} as the total number of output classes. f k (x) {\textstyle f_{k}(x)} returns the score (or a probability between 0 and 1) that the input x {\textstyle x} belongs to class k {\textstyle k} , which allows the classifier's class output for any input x {\textstyle x} to be defined as argmax k = 1 , . . . , K f k (x) {\textstyle {\text{argmax}}_{k=1,...,K}f_{k}(x)} . The goal of this attack is as follows:

argmax k = 1 , . . . , K f k (x ^) ≠ y , | | x ^ − x | | p ≤ ■ and x ^ ∈ [0 , 1] d {\displaystyle {\text{argmax}}_{k=1,...,K}f_{k}({\hat {x}})\neq y,||{\hat {x}}-x||_{p}\leq \epsilon {\text{ and }}{\hat {x}}\in [0,1]^{d}}

In other words, finding some perturbed adversarial example x ^ {\textstyle {\hat {x}}} such that the classifier incorrectly classifies it to some other class under the constraint that x ^ {\textstyle {\hat {x}}} and x {\textstyle x} are similar. The paper then defines loss L {\textstyle L} as L (f (x ^) , y) = f y (x ^) − max k ≠ y f k (x ^) {\textstyle L(f({\hat {x}}),y)=f_{y}({\hat {x}})-\max _{k\neq y}f_{k}({\hat {x}})} and proposes the solution to finding adversarial example x ^ {\textstyle {\hat {x}}} as solving the below constrained optimization problem :

min x ^ ∈ [0 , 1] d L (f (x ^) , y) , s.t. | | x ^ − x | | p ≤ ■ {\displaystyle \min _{{\hat {x}}\in [0,1]^{d}}L(f({\hat {x}}),y),{\text{ s.t. }}||{\hat {x}}-x||_{p}\leq \epsilon }

The result in theory is an adversarial example that is highly confident in the incorrect class but is also very similar to the original image. To find such example, Square Attack utilizes the iterative random search technique to randomly perturb the image in hopes of improving the objective function. In each step, the algorithm perturbs only a small square section of pixels, hence the name Square Attack, which terminates as soon as an adversarial example is found in order to improve query efficiency. Finally, since the attack algorithm uses scores and not gradient information, the authors of the paper indicate that this approach is not affected by gradient masking, a common technique formerly used to prevent evasion attacks.

This black box attack was also proposed as a query efficient attack, but one that relies solely on access to any input's predicted output class. In other words, the HopSkipJump attack does not require the ability to calculate gradients or access to score values like the Square Attack, and will require just the model's class prediction output (for any given input). The proposed attack is split into two different settings, targeted and untargeted, but both are built from the general idea of adding minimal perturbations that leads to a different model output. In the targeted setting, the goal is to cause the model to misclassify the perturbed image to a specific target label (that is not the original label). In the untargeted setting, the goal is to cause the model to misclassify the perturbed image to any label that is not the original label. The attack objectives for both are as follows where x {\textstyle x} is the original image, x ′ {\textstyle x^{\prime }} is the adversarial image, d {\textstyle d} is a distance function between images, c ∗ {\textstyle c^{*}} is the target label, and C {\textstyle C} is the model's classification class label function:

Targeted: min x ′ d (x ′ , x) subject to C (x ′) = c ∗ {\displaystyle {\textbf {Targeted:}}\min _{x^{\prime }}d(x^{\prime },x){\text{ subject to }}C(x^{\prime })=c^{*}}

Untargeted: min x ′ d (x ′ , x) subject to C (x ′) ≠ C (x) {\displaystyle {\textbf {Untargeted:}}\min _{x^{\prime }}d(x^{\prime },x){\text{ subject to }}C(x^{\prime })\neq C(x)}

To solve this problem, the attack proposes the following boundary function S {\textstyle S} for both the untargeted and targeted setting:

S (x ′) := { max c ≠ C (x) F (x ′) c − F (x ′) C (x) , (Untargeted) F (x ′) c ∗ − max c ≠ c ∗ F (x ′) c , (Targeted) {\displaystyle S(x^{\prime }):={\begin{cases}\max _{c\neq C(x)}{F(x^{\prime })_{c}}-F(x^{\prime })_{C(x)},&{\text{(Untargeted)}}\\F(x^{\prime })_{c^{*}}-\max _{c\neq c^{*}}{F(x^{\prime })_{c}},&{\text{(Targeted)}}\end{cases}}}

This can be further simplified to better visualize the boundary between different potential adversarial examples:

S (x ′) > 0 ■ { a r g m a x c F (x ′) ≠ C (x) , (Untargeted) a r g m a x c F (x ′) = c ∗ , (Targeted) {\displaystyle S(x^{\prime })>0\iff {\begin{cases}argmax_{c}F(x^{\prime })\neq C(x),&{\text{(Untargeted)}}\\argmax_{c}F(x^{\prime })=c^{*},&{\text{(Targeted)}}\end{cases}}}

With this boundary function, the attack then follows an iterative algorithm to find adversarial examples x ′ {\textstyle x^{\prime }} for a given image x {\textstyle x} that satisfies the attack objectives.

Initialize x {\textstyle x} to some point where S (x) > 0 {\textstyle S(x)>0}

Iterate below Boundary search Gradient update Compute the gradient Find the step size

Boundary search

Gradient update Compute the gradient Find the step size

Compute the gradient

Find the step size

Boundary search uses a modified binary search to find the point in which the boundary (as defined by S {\textstyle S}) intersects with the line between x {\textstyle x} and x ′ {\textstyle x^{\prime }} . The next step involves calculating the gradient for x {\textstyle x} , and update the original x {\textstyle x} using this gradient and a pre-chosen step size. HopSkipJump authors prove that this iterative algorithm will converge, leading x {\textstyle x} to a point right along the boundary that is very close in distance to the original image.

However, since HopSkipJump is a proposed black box attack and the iterative algorithm above requires the calculation of a gradient in the second iterative step (which black box attacks do not have access to), the authors propose a solution to gradient calculation that requires only the model's output predictions alone. By generating many random vectors in all directions, denoted as u b {\textstyle u_{b}} , an approximation of the gradient can be calculated using the average of these random vectors weighted by the sign of the boundary function on the image x ′ + δ u b {\textstyle x^{\prime }+\delta _{u_{b}}} , where δ u b {\textstyle \delta _{u_{b}}} is the size of the random vector perturbation:

∇ S (x ′ , δ) ≈ 1 B ∑ b = 1 B φ (x ′ + δ u b) u b {\displaystyle \nabla S(x^{\prime },\delta)\approx {\frac {1}{B}}\sum _{b=1}^{B}\phi (x^{\prime }+\delta _{u_{b}})u_{b}}

The result of the equation above gives a close approximation of the gradient required in step 2 of the iterative algorithm, completing HopSkipJump as a black box attack.

White box attacks

White box attacks assumes that the adversary has access to model parameters on top of being able to get labels for provided inputs.

One of the first proposed attacks for generating adversarial examples was proposed by Google researchers Ian J. Goodfellow , Jonathon Shlens, and Christian Szegedy. The attack was called fast gradient sign method (FGSM), and it consists of adding a linear amount of in-perceivable noise to the image and causing a model to incorrectly classify it. This noise is calculated by multiplying the sign of the gradient with respect to the image we want to perturb by a small constant epsilon. As epsilon increases, the model is more likely to be fooled, but the perturbations become easier to identify as well. Shown below is the equation to generate an adversarial example where x {\textstyle x} is the original image, ■ {\textstyle \epsilon } is a very small number, Δ x {\textstyle \Delta _{x}} is the gradient function, J {\textstyle J} is the loss function, θ {\textstyle \theta } is the model weights, and y {\textstyle y} is the true label.

a d v x = x + ■ · s i g n (Δ x J (θ , x , y)) {\displaystyle adv_{x}=x+\epsilon \cdot sign(\Delta _{x}J(\theta ,x,y))}

One important property of this equation is that the gradient is calculated with respect to the input image since the goal is to generate an image that maximizes the loss for the original image of true label y {\textstyle y} . In traditional gradient descent (for model training), the gradient is used to update the weights of the model since the goal is to minimize the loss for the model on a ground truth dataset. The Fast Gradient Sign Method was proposed as a fast way to generate adversarial examples to evade the model, based on the hypothesis that neural networks cannot resist even linear amounts of perturbation to the input. FGSM has shown to be effective in adversarial attacks for image classification and skeletal action recognition.

In an effort to analyze existing adversarial attacks and defenses, researchers at the University of California, Berkeley, Nicholas Carlini and David Wagner in 2016 propose a faster and more robust method to generate adversarial examples.

The attack proposed by Carlini and Wagner begins with trying to solve a difficult non-linear optimization equation:

min (| | δ | | p) subject to C (x + δ) = t , x + δ ∈ [0 , 1] n {\displaystyle \min(||\delta ||_{p}){\text{ subject to }}C(x+\delta)=t,x+\delta \in [0,1]^{n}}

Here the objective is to minimize the noise (δ {\textstyle \delta }), added to the original input x {\textstyle x} , such that the machine learning algorithm (C {\textstyle C}) predicts the original input with delta (or x + δ {\textstyle x+\delta }) as some other class t {\textstyle t} . However instead of directly the above equation, Carlini and Wagner propose using a new function f {\textstyle f} such that:

C (x + δ) = t ■ f (x + δ) ≤ 0 {\displaystyle C(x+\delta)=t\iff f(x+\delta)\leq 0}

This condenses the first equation to the problem below:

min (| | δ | | p) subject to f (x + δ) ≤ 0 , x + δ ∈ [0 , 1] n {\displaystyle \min(||\delta ||_{p}){\text{ subject to }}f(x+\delta)\leq 0,x+\delta \in [0,1]^{n}}

and even more to the equation below:

min (| | δ | | p + c · f (x + δ)) , x + δ ∈ [0 , 1] n {\displaystyle \min(||\delta ||_{p}+c\cdot f(x+\delta)),x+\delta \in [0,1]^{n}}

Carlini and Wagner then propose the use of the below function in place of f {\textstyle f} using Z {\textstyle Z} , a function that determines class probabilities for given input x {\textstyle x} . When substituted in, this equation can be thought of as finding a target class that is more confident than the next likeliest class by some constant amount:

f (x) = ([max i ≠ t Z (x) i] − Z (x) t) + {\displaystyle f(x)=([\max _{i\neq t}Z(x)_{i}]-Z(x)_{t})^{+}}

When solved using gradient descent, this equation is able to produce stronger adversarial examples when compared to fast gradient sign method that is also able to bypass defensive distillation, a defense that was once proposed to be effective against adversarial examples.

Defenses

Researchers have proposed a multi-step approach to protecting machine learning.

Threat modeling – Formalize the attackers goals and capabilities with respect to the target system.

Attack simulation – Formalize the optimization problem the attacker tries to solve according to possible attack strategies.

Attack impact evaluation

Countermeasure design

Noise detection (For evasion based attack)

Information laundering – Alter the information received by adversaries (for model stealing attacks)

Mechanisms

A number of defense mechanisms against evasion, poisoning, and privacy attacks have been proposed, including:

Secure learning algorithms

Byzantine-resilient algorithms

Multiple classifier systems

AI-written algorithms.

AIs that explore the training environment; for example, in image recognition, actively navigating a 3D environment rather than passively scanning a fixed set of 2D images.

Privacy-preserving learning

Ladder algorithm for Kaggle -style competitions

Game theoretic models

Sanitizing training data

Adversarial training

Backdoor detection algorithms

Gradient masking/obfuscation techniques: to prevent the adversary exploiting the gradient in white-box attacks. This family of defenses is deemed unreliable as these models are still vulnerable to black-box attacks or can be circumvented in other ways.

Ensembles of models have been proposed in literature, which have shown to be ineffective against evasion attacks but effective against data poisoning attacks.

See also

Pattern recognition

Fawkes (image cloaking software)

Generative adversarial network

References

External links

MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems

NIST 8269 Draft: A Taxonomy and Terminology of Adversarial Machine Learning

NIPS 2007 Workshop on Machine Learning in Adversarial Environments for Computer Security

AlfaSVMLib Archived 2020-09-24 at the Wayback Machine – Adversarial Label Flip Attacks against Support Vector Machines

Laskov, Pavel; Lippmann, Richard (2010). "Machine learning in adversarial environments". Machine Learning . 81 (2): 115– 119. doi : 10.1007/s10994-010-5207-6 . S2CID 12567278 .

Dagstuhl Perspectives Workshop on " Machine Learning Methods for Computer Security "

Workshop on Artificial Intelligence and Security , (AISec) Series

v

t

e

History timeline

timeline

Companies

Projects

Parameter Hyperparameter

Hyperparameter

Loss functions

Regression Bias–variance tradeoff Double descent Overfitting

Bias–variance tradeoff

Double descent

Overfitting

Clustering

Gradient descent SGD Quasi-Newton method Conjugate gradient method

SGD

Quasi-Newton method

Conjugate gradient method

Backpropagation

Attention

Convolution

Normalization Batchnorm

Batchnorm

Activation Softmax Sigmoid Rectifier

Softmax

Sigmoid

Rectifier

Gating

Weight initialization

Regularization

Datasets Augmentation

Augmentation

Prompt engineering

Reinforcement learning Q-learning SARSA Imitation Policy gradient

Q-learning

SARSA

Imitation

Policy gradient

Diffusion

Latent diffusion model

Autoregression

Adversary

RAG

Uncanny valley

RLHF

Self-supervised learning

Reflection

Recursive self-improvement

Hallucination

Word embedding

Vibe coding

Machine learning In-context learning

In-context learning

Artificial neural network Deep learning

Deep learning

Language model Large language model NMT

Large language model

NMT

Reasoning language model

Model Context Protocol

Intelligent agent

Artificial human companion

Humanity's Last Exam

Artificial general intelligence (AGI)

AlexNet

WaveNet

Human image synthesis

HWR

OCR

Computer vision

Speech synthesis 15.ai ElevenLabs

15.ai

ElevenLabs

Speech recognition Whisper

Whisper

Facial recognition

AlphaFold

Text-to-image models Aurora DALL-E Firefly Flux Ideogram Imagen Midjourney Recraft Stable Diffusion

Aurora

DALL-E

Firefly

Flux

Ideogram

Imagen

Midjourney

Recraft

Stable Diffusion

Text-to-video models Dream Machine Runway Gen Hailuo AI Kling Sora Veo

Dream Machine

Runway Gen

Hailuo AI

Kling

Sora

Veo

Music generation Riffusion Suno AI Udio

Riffusion

Suno AI

Udio

Word2vec

Seq2seq

GloVe

BERT

T5

Llama

Chinchilla AI

PaLM

GPT 1 2 3 J ChatGPT 4 4o o1 o3 4.5 4.1 o4-mini 5

1

2

3

J

ChatGPT

4

4o

o1

o3

4.5

4.1

o4-mini

5

Claude

Gemini Gemini (language model) Gemma

Gemini (language model)

Gemma

Grok

LaMDA

BLOOM

DBRX

Project Debater

IBM Watson

IBM Watsonx

Granite

PanGu-Σ

DeepSeek

Qwen

AlphaGo

AlphaZero

OpenAI Five

Self-driving car

MuZero

Action selection AutoGPT

AutoGPT

Robot control

Alan Turing

Warren Sturgis McCulloch

Walter Pitts

John von Neumann
Claude Shannon
Shun'ichi Amari
Kunihiko Fukushima
Takeo Kanade
Marvin Minsky
John McCarthy
Nathaniel Rochester
Allen Newell
Cliff Shaw
Herbert A. Simon
Oliver Selfridge
Frank Rosenblatt
Bernard Widrow
Joseph Weizenbaum
Seymour Papert
Seppo Linnainmaa
Paul Werbos
Geoffrey Hinton
John Hopfield
Jürgen Schmidhuber
Yann LeCun
Yoshua Bengio
Lotfi A. Zadeh
Stephen Grossberg
Alex Graves
James Goodnight
Andrew Ng
Fei-Fei Li
Alex Krizhevsky
Ilya Sutskever
Oriol Vinyals
Quoc V. Le
Ian Goodfellow
Demis Hassabis
David Silver
Andrej Karpathy
Ashish Vaswani

Noam Shazeer

Aidan Gomez

John Schulman

Mustafa Suleyman

Jan Leike

Daniel Kokotajlo

François Chollet

Neural Turing machine

Differentiable neural computer

Transformer Vision transformer (ViT)

Vision transformer (ViT)

Recurrent neural network (RNN)

Long short-term memory (LSTM)

Gated recurrent unit (GRU)

Echo state network

Multilayer perceptron (MLP)

Convolutional neural network (CNN)

Residual neural network (RNN)

Highway network

Mamba

Autoencoder

Variational autoencoder (VAE)

Generative adversarial network (GAN)

Graph neural network (GNN)

Category