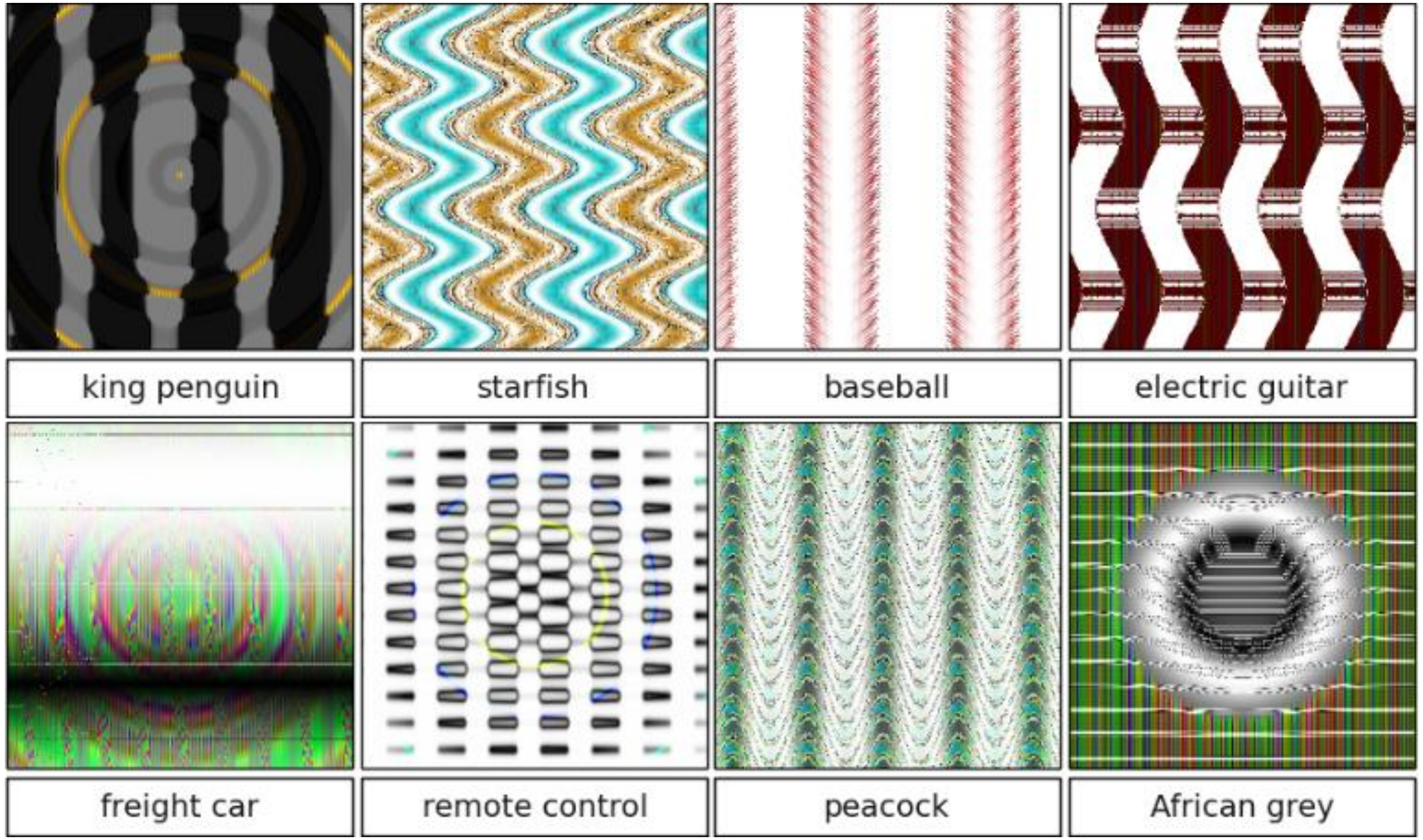
CS 6301.007

Machine Learning in Cyber Security

Wei Yang

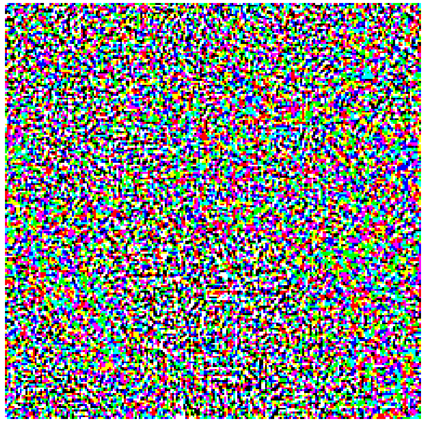
Department of Computer Science
University of Texas at Dallas

Adversarial examples



Adversarial examples

Classified as panda



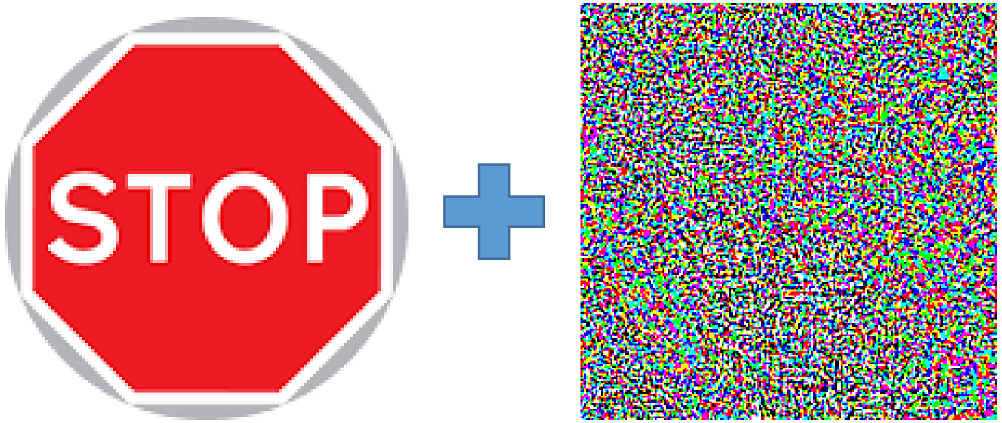
Small adversarial noise

Classified as gibbon

Who cares about panda?

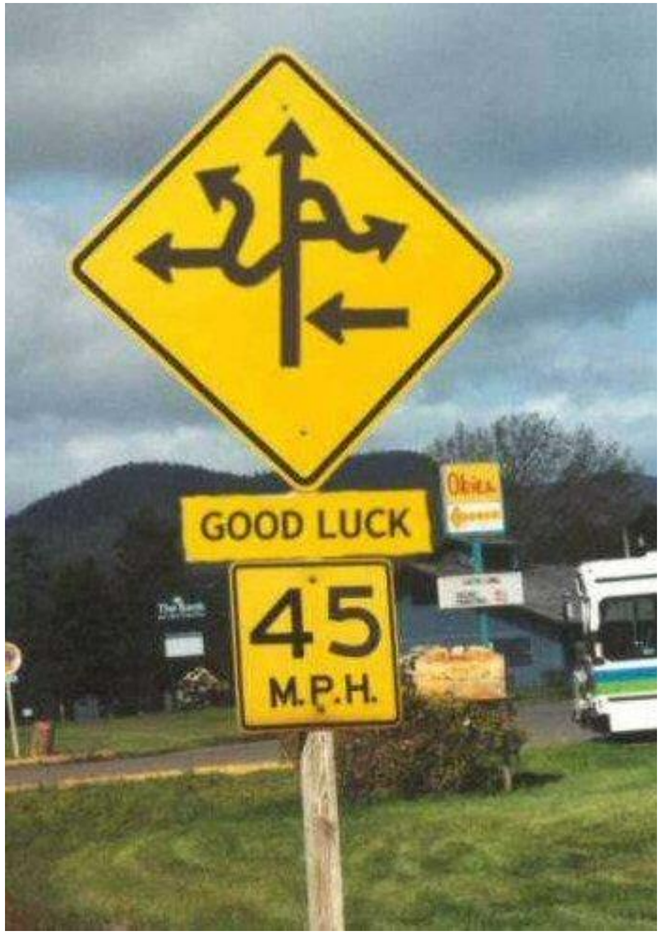
Suppose that the sign is

Add adversarial noise…



Small adversarial noise

and the ML in your self-driving car thinks it's



So, is that all there is to it?

NO

AI & Cybersecurity

- Cylance: "Leveraging *complex mathematical algorithms*, *predictive artificial intelligence* (AI) capabilities, and the power of *machine learning* techniques, CylancePROTECT has emerged as the most strategic new offering in the Forrester Wave report."

Adversarial ML

- Applications of ML in security
  - Fraud detection
  - Malware detection
  - Intrusion detection
  - Spam detection
- What do all of these have in common?
  - Detect bad “things” (actors, actions, objects)

Bad actors

- Key issue in AML: bad actors (who do bad things) have *objectives*
  - the main one is not getting detected
  - they can change their behavior to avoid detection
- This gives rise to *evasion attacks*
  - Attacks on ML, where malicious objects are deliberately transformed to evade detection (prediction by ML that these are malicious)

Data poisoning

- An entirely different class of attacks are *data poisoning attacks*
- In these, an adversary introduces malicious modifications to the data used for training
  - Can insert instances (for example, send specially crafted emails, either benign or malicious)
  - Can modify instances in the data (hack one of the servers used to store a part of the data)
  - Can selectively remove some instances

Evasion vs. poisoning

- The crucial distinction between these classes of attacks is
  - *Evasion* is an ***attack on the learned model*** (e.g., an actual classifier)
  - *Poisoning* is an ***attack on the algorithm*** (e.g., least-squares regression learning)
- "*adversarial examples*" in DNN are close to evasion attacks, as they attack the model

Adversarial Evasion

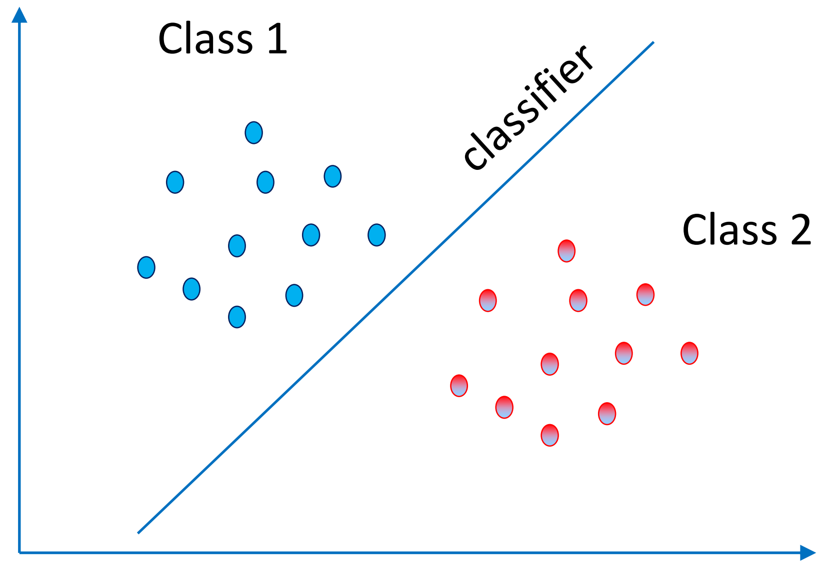
outline

- Evasion Attacks
- Evasion-robust Classification
- Validating evasion attack models

outline

- Evasion Attacks
  - Modeling evasion
  - White-box vs. black-box attacks
- Evasion-robust Classification
- Validating evasion attack models

Classification learning



Data set: $\{(x_1,y_1),\ldots,(x_n,y_n)\}$, $(x,y) \sim \mathbb{D}$

x: feature vectors

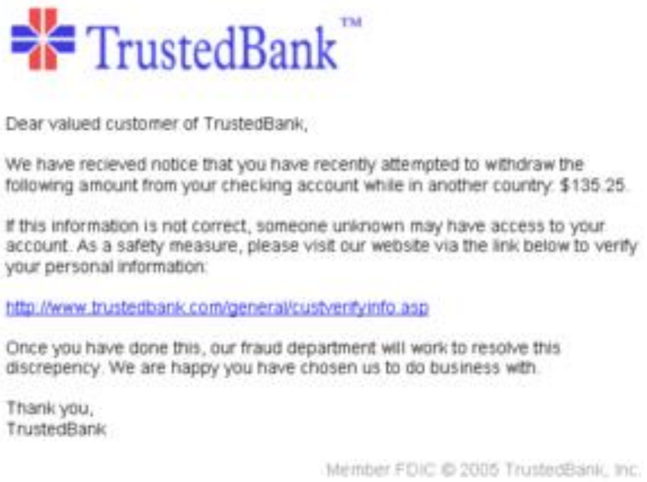
y: binary label in {-1, +1} representing which class x belongs to

Learning: train classifier f(x) on data

Prediction: use f(x) to predict label for arbitrary x

Classification in adversarial settings

- Often, classifiers are tasked with telling apart "good" from "bad"
  - Spam vs. non-spam (ham)
  - Benign vs. malicious software
  - Intrusion detection


TrustedBank™

Dear valued customer of TrustedBank,

We have recieved notice that you have recently attempted to withdraw the following amount from your checking account while in another country: $135.25.

If this information is not correct, someone unknown may have access to your account. As a safety measure, please visit our website via the link below to verify your personal information:

http://www.trustedbank.com/general/custverifyinfo.asp

Once you have done this, our fraud department will work to resolve this discrepency. We are happy you have chosen us to do business with.

Thank you,
TrustedBank

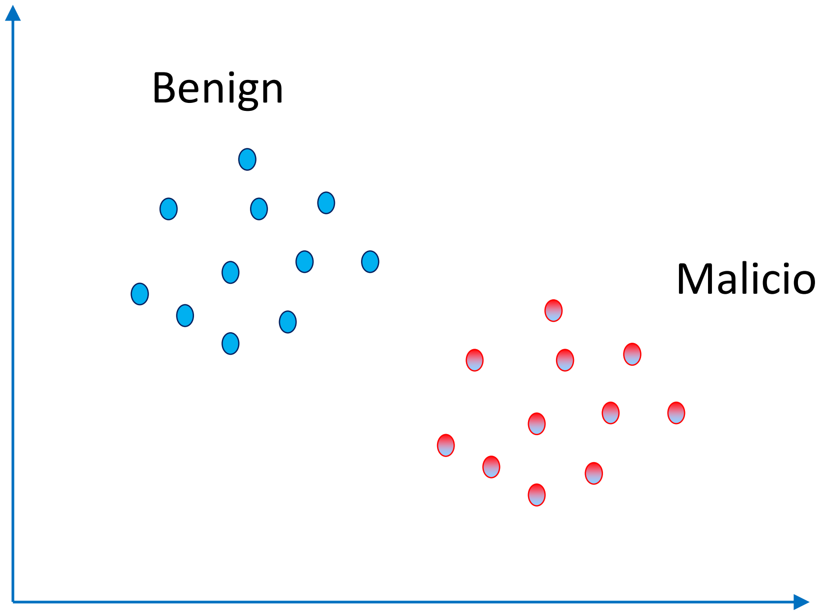
Member FDIC © 2005 TrustedBank, Inc.

Evasion attacks

- **Adversary who previously chose instance *x* (which is now classified as malicious) now chooses another instance *x'* which is classified as benign**

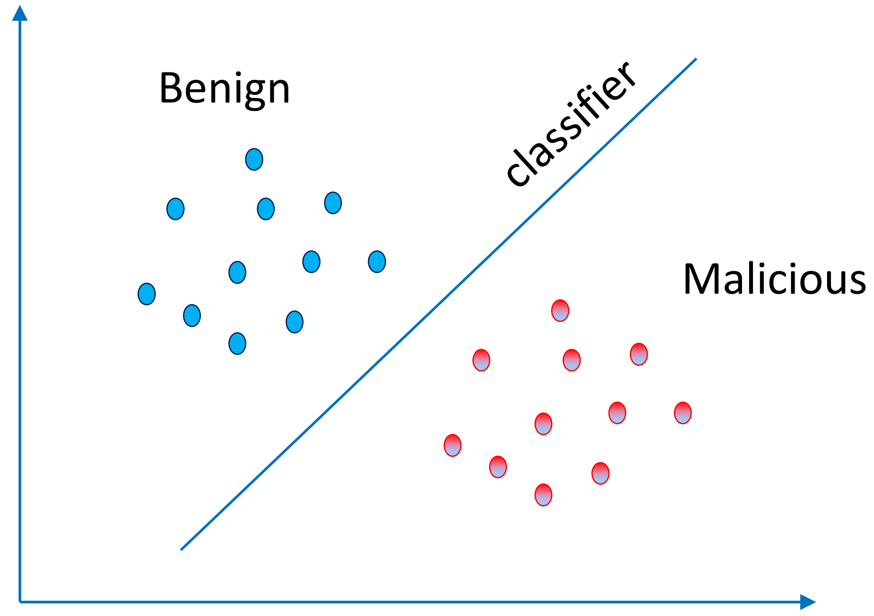
Evasion attacks

- **Adversary who previously chose instance *x* (which is now classified as malicious) now chooses another instance *x'* which is classified as benign**



Evasion attacks

- **Adversary who previously chose instance *x* (which is now classified as malicious) now chooses another instance *x'* which is classified as benign**



Example of Evasion

1. From: spammer@example.com
   Cheap mortgage now!!!

Feature Weights

2. cheap = 1.0
   mortgage = 1.5

3. Total score = 2.5 > 1.0 (threshold)

Spam

Example of Evasion

1\.
From: spammer@example.com
Cheap mortgage now!!!
Joy Oregon

Feature Weights

2\.
cheap = 1.0
mortgage = 1.5
Joy= -1.0
Oregon = -1.0

3\. Total score = 0.5 < 1.0 (threshold)

OK

Modeling evasion attacks

- Attacker has an "ideal" feature vector x_{ideal}
  - These are the original malicious feature vectors in training data
- Modifying *x* into another feature vector *x'* incurs a cost $c(x_{ideal}, x')$
- The attacker's goal is to appear "benign" to the classifier
- Observation: **feature space modeling**
  - Attacker can make arbitrary changes to features
  - Cost is meant to capture any constraints faced by the attacker in practice
  - **No actual attack instances are generated/validated** (this just produces a new feature vector rather than, say, another malicious PDF)

The lowd & meek model

- $\min_{x'} c(x_{ideal}, x')$ *s.t.:* x' classified as benign
  - Small modification: ensure $c(x_{ideal}, x') \le$ cost budget; otherwise, return x_{ideal}
- $c(x_{ideal}, x) = \sum_i \alpha_i |x_i - x_{ideal,i}|$
  - Note: can generalize this to a (weighted) l_p norm
  - $c(x_{ideal}, x) = \sum_i \alpha_i |x_i - x_{ideal,i}|^p$
    - Common special case: $c(x_{ideal}, x) = ||x - x_{ideal}||$

Norms

- **L-1 norm**

$$\|\bar{x}\|_1 = x_1 + x_2 + \cdots + x_d$$

- **L-2 norm**

$$\|\bar{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_d^2}$$

- **L-n norm**

$$\|\bar{x}\|_n = \sqrt[n]{x_1^n + x_2^n + \cdots + x_d^n}$$

- **L-∞ norm**

$$\|\bar{x}\|_\infty = \max(x_1, x_2, \ldots, x_n)$$

Other models

- Suppose that the classifier can be described as

  $$f(x) = \text{sgn}\{g(x)\}, \text{ for some } g(x) : X \rightarrow R$$

- Biggio et al. ECML '13:
  - $\min_x g(x)$ s.t.: $c(x_{ideal}, x') \leq$ cost budget

- Li and Vorobeychik, '16
  - $\min_x g(x) + \lambda\, c(x_{ideal}, x')$

Solving attacker optimization

- Commonly, these are hard to solve optimally
- Approaches depend on the nature of the feature space:
  - **Continuous** vs. **binary** (or discrete)

Continuous features

- Lowd & Meek model: easy to solve for linear classifiers
- Nelson et al. JMLR ‘12: can approximately solve for convex-inducing classifiers
- More generally, gradient descent approaches (Biggio et al., ECML ‘13)

Binary features

- Lowd & Meek KDD '05: 2-approximation algorithm for linear classifiers
  - Start with a benign instance
  - First, try to flip features in benign instance that are different from x_{ideal}
  - Then, try to flip pairs of features different from x_{ideal} and 1 feature that is the same (this also reduces cost by 1)
- Dalvi et al. KDD '04: minimum-cost camouflage (MCC); linear integer program (for Naïve Bayes classifier)

Is distance the right cost function?

Distance Based Cost Function Underestimates Adversary

~~C = 5~~

C = 1

Synonym

Spam

Hello!
Are you ready to become more active and attractive than ever before?
Our final product for losing weight is on clearance now.
Follow the link and you will find he cheapest way to gain your body back.
http://www.ebay.com/application_form

Ham

Hello!
Are you happy to become more active and attractive than ever before?
Our final merchandises for losing weigbt is on claerance now.
Follow the link and you will find he deapest way to gain your body back.
http://www.ebay.com/application_form

Letter substitution

An Alternative Cost Function

- Model the adversarial cost function
  - Traditional: Distance based cost function

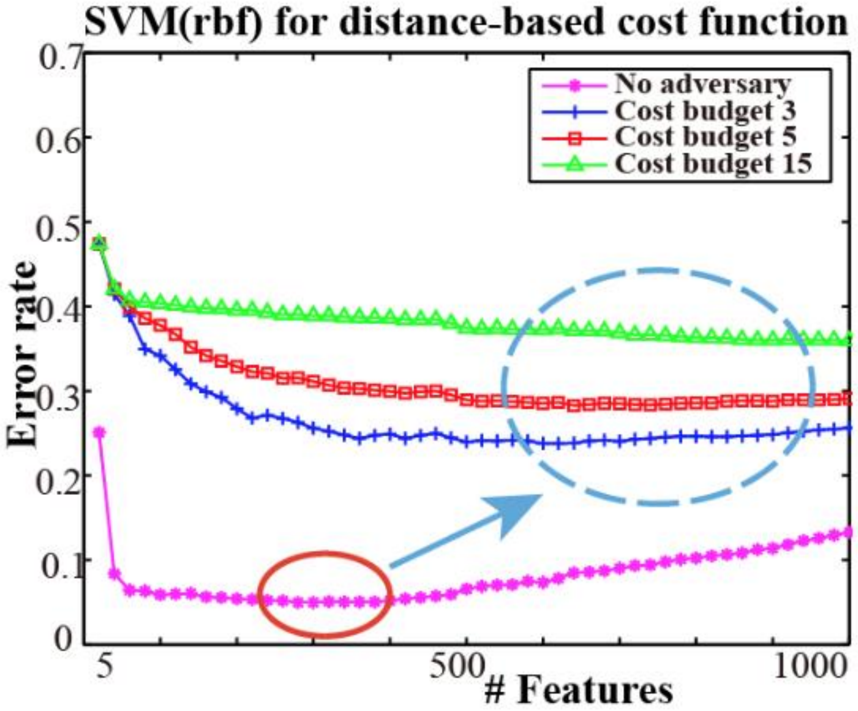
$$c(x', x^A) = \sum_i a_i |x'_i - x^A_i|$$

  - Equivalence based cost function

$$c(x', x^A) = \sum_i \min_{j \in \boxed{F_i} | x^A_j \oplus x'_j = 1} a_i |x'_j - x^A_i|$$

Feature Class

Perils of Dimension Reduction



No Adversary: Dimension Reduction = Good
With Adversary: Dimension Reduction = Vulnerable

White-box vs. black-box attacks

- In attack models, assumed that the attacker knows the classifier
  - Black-box attacks: attacker has a query access to the classifier; can get examples
- Lowd & Meek; Nelson et al.: minimizing the evasion cost through a sequence of queries to the classifier
  - NP-Hard in general (even for linear classifiers)
  - Poly-time approximations for linear and convex-inducing classifiers
  - **Is the black-box attack fundamentally hard?**

Black-box attacks

- ***Can the adversary approximate the classifier h used by the defender to (near)-arbitrary precision?***
  - Using only queries x to find out h(x)?
  - NP-Hard in general

everse engineering is easy “in practice”

- Previous results on black-box learnability are worst-case over an entire family of classifiers (linear, convex-inducing)
- **Observation**: these classifiers do not spontaneously appear; **they are learned from data**!
  - This fact implies a lot of structure: *since someone learned them to begin with, they should be learnable*
- **Theorem**: suppose a hypothesis class H is efficiently learnable, and h in H is learned (given data). Then h can be efficiently **reverse engineered**.
  - Reverse engineered: learned with arbitrarily small error
  - Follows directly from the fact that H is efficiently learnable and h is in H.
- *Consequence: theoretical reason why “black-box” attacks, e.g., with DNNs, work*

References (evasion modeling)

- Dalvi et al. Adversarial classification. KDD ‘04.
- Lowd and Meek. Adversarial learning. KDD ’05.
- Nelson et al. Query strategies for evading convex-inducing classifiers. JMLR ‘12.
- Biggio et al. Evasion attacks against machine learning at test time. ECML/PKDD ‘13.
- Li and Vorobeychik. Feature cross-substitution in adversarial classification. NIPS ‘14.
- Vorobeychik and Li. Optimal randomized classification in adversarial settings. AAMAS ‘14.
- Li, Vorobeychik, Chen. A general retraining framework for scalable adversarial classification. arxiv, 2016.

- ***Not exhaustive***
    - **Vorobeychik and Kantarcioglu, Adversarial Machine Learning book will have a more extensive bibliography**

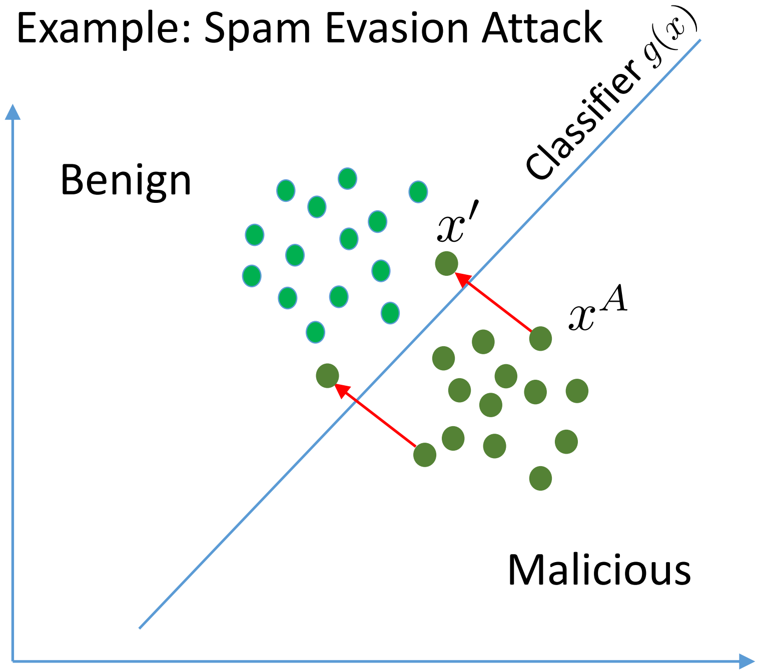
outline

- Evasion Attacks
- Evasion-robust Classification
  - Optimal evasion-robust classification
  - Scaling up with systematic retraining
- Validating evasion attack models

Stackelberg Game

- Learner: commits strategy S_d
- Adversary: best response based on S_d

Example: Spam Evasion Attack



x^A : adversarial instance

Learner commits to a strategy $\boxed{S_d}$ → $g(x)$

↓

Adversary solves $\boxed{x' = \mathcal{O}_{s_d}(x^A)}$ to generate x' from ideal x^A

Adversary Model

Designing evasion-robust classifiers

- If ”back-box” attacks are approximately “white-box” attacks, we focus on white-box evasion attacks

Robust learning through regularization

- Consider the following problem:

$$\min_{w} \sum_{j} l\left(w^T x_j\right)$$

Empirical risk minimization

Robust learning through regularization

- Its robust analog:

$$\min_{w} \sum_{j} \max_{x' \in C_j} l(w^T x')$$

"attacker" trying to maximize loss of a learned weight vector ***w***

Robust learning through regularization

- Robust empirical risk minimization:

$$\min_{w} \sum_{j} \max_{x' \in C_j} l(w^T x')$$

- Suppose $C_j = \{x \mid \left|\left|x - x_j\right|\right|_p \leq \lambda\}$

Robust learning through regularization

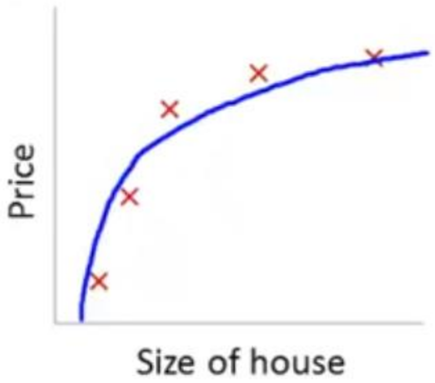
- Robust empirical risk minimization:

$$\min_{w} \sum_{j} \max_{x' \in C_j} l(w^T x')$$

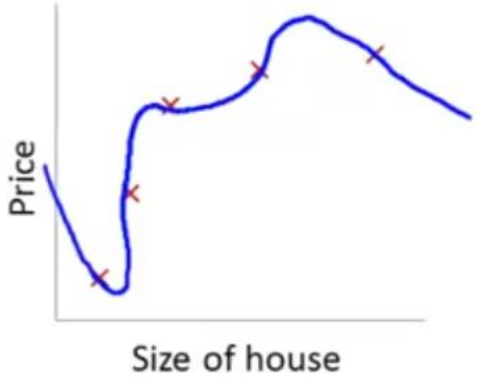
- Suppose $C_j = \{x \mid \left|\left|x - x_j\right|\right|_p \leq \lambda\}$
- If we use hinge (SVM) loss, this problem is equivalent to regularized SVM with l_q regularizer, where q is the dual norm of p

Regularization

Intuition



$\theta_0 + \theta_1 x + \theta_2 x^2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Suppose we penalize and make θ_3, θ_4 really small.

$$\rightarrow \quad \min_\theta \underline{\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2} + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

Small values for parameters $\boxed{\theta_0, \theta_1, \ldots, \theta_n} \leftarrow$

- "Simpler" hypothesis $\leftarrow$
- Less prone to overfitting $\leftarrow$

$\rightarrow \underline{\theta_3, \theta_4} \approx 0$

Robust learning through regularization

- Illustration:
  - Suppose that the attacker has an l_1 norm constraint: $C_j = \{x \mid \sum_k |x_k - x_{jk}| \leq \lambda\}$
  - Then you should use l_∞-regularized SVM
  - i.e., all features used, but have small magnitude
  - Attacker has to change a lot of features to be effective

Question?

But this is not quite our problem

- First, attacker is maximizing loss
  - In fact, attackers are interested in not being detected, which is not the same; may wish to consider alternative models of attacks
- Attackers may only correspond to malicious instances

Adversarial risk minimization

- Minimize l_1 regularized (hinge) risk, accounting for evasion
- Optimization problem:

$$\min_{w} \sum_{j|y_j=-1} l(-w^T x_j) + \sum_{j|y_j=+1} l(w^T A(x_j; w)) + \lambda \|w\|_1$$

  - $l(\ *\ )$: hinge loss
  - $A(x_j;w)$: adversarial decision model for an attacker who previously used a feature vector x_j

Adversarial risk minimization

- Can formulate the problem as a mixed integer linear program
- *In this formulation, we capture $A(x_j;w)$ for an optimizing attacker using constraints*
  - Assume that the attacker acts according to our evasion model earlier
- Scalability challenge: too many constraints
  - Each constraint corresponds to a malicious instance in the data and all possible alternative instances for the corresponding attacker
- Approach: clustering attacks + constraint generation

Limitations

UT DALLAS

- Scalability: formulation and solution approach still can only scale to dozens of features and relatively small data sets
- Classifier limitation: Limited to linear classifiers
- Attack model limitation: Attack model is limited to threats which are optimizers (minimizing evasion cost); what about other models (e.g., behavioral, data-driven, etc)?
- Simple solution: iterative retraining

Retraining

- Start with original data
- Use *any learning algorithm* to learn a model f
- For malicious instances, apply *any evasion method* to generate new instances x' to add to the dataset
- Repeat
- Stop when:
  - No new instances to add
  - Iteration limit
  - Classifier changes small between successive iterations

Retraining

- Start with original data
- Use *any learning algorithm* to learn a model *f*
- For malicious instances, apply *any evasion method* to generate new instances *x'* to add to the dataset
- Repeat
- Stop when:
  - No new instances to add
  - Iteration limit
  - Classifier changes small between successive iterations
- *RAD: Retraining with ADversarial examples*

Effectiveness of Retraining

- Theorem: if algorithm terminates when no new instances to add, the result is an *upper bound on the optimal adversarial risk*

Randomized intrusion detection

- Can improve robustness by randomizing classification decisions

Learning in a box

- Key idea: separate *learning* [about prediction based on current distribution of attacks] and *operational decisions* [using predictions made by learning, an adversary model, and operational constraints, to decide what to do]
  - Use learning as a black box to get *p(x)* = probability that *x* is generated by a malicious actor (e.g., use Naïve Bayes or logistic regression)
    - ***p(x) is a probability distribution over adversarial preferences*** *(i.e., probability that x is the **ideal instance** for the corresponding adversary)*
  - Optimize operational decisions based on *p(x)* and a *model of adversarial response (adversary's utility a function of how far they are from ideal instance, and probability the email is filtered)*
  - Operational decisions can be randomized; use instance-based randomization, **q(x)** (probability of "acting" on x)

Optimization Problem

$\max_q U_D(q,p,X)$

subject to:

$0 \leq q(x) \leq 1$

Represent attacker's response as a set of constraints

$v_A(x;q) \geq U_A(x,x'; q)$ *for all attacks* x'

operational budget constraint

$\boxed{\Sigma_x q(x) \leq c|X|}$

Linear program, but: *X* (set of all feature vectors) is extremely large (binary features: 2^n)

Scalability

- Idea: represent q(x) using basis functions

$$q(x) = \sum_j \alpha_j \phi_j(x)$$

- and optimize over α_j
  - Optimization program is linear in α
  - But: what should the basis be?

scalability

- Commonly, the input space X is Boolean
  - Spam/phish detection: presence of specific words/phrases in the text
- Fact: any function f on Boolean ({-1,+1}) vector space can be represented using a parity basis

$$f(x) = \sum_{S \subseteq [1..n]} \alpha_S \chi_S$$

$$\chi_S = \prod_{j \in S} x_j$$

- Idea: solve for q(x) using training data; choose a small parity basis to approximate q(x); use this basis in the full optimization problem

Solution approach

- Approximating the basis:
  - Can formulate an integer program to compute the parity function with the largest coefficient
  - Greedily add basis functions unti the largest remaining coefficient is below a threshold
- Dealing with constraints:
  - Constraint generation (iteratively computing attacker's best response)

$$\max_S \quad \frac{1}{K}\sum_{k=1}^{K} q(x^k) r_S^k$$

$$\begin{aligned} s.t.: \quad & S^T x^k = 2y^k + h^k \\ & r_S^k = 1 - 2h^k \\ & y^k \in Z, h^k \in \{0,1\}, S \in \{0,1\}^n \end{aligned}$$

Uses the fact that a coefficient of a basis can be computed using

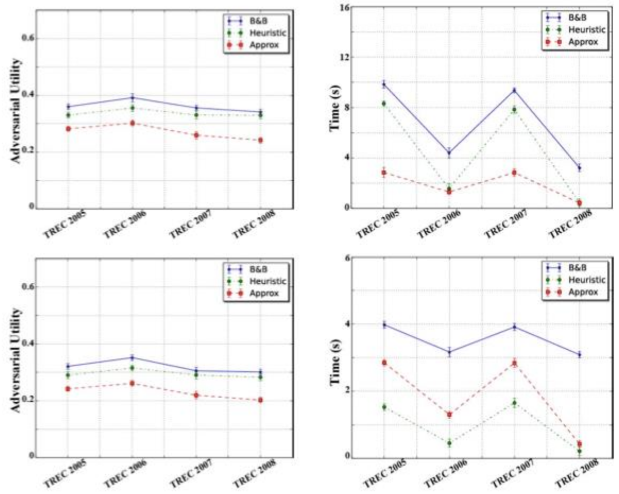
$$\alpha_S = \mathbf{E}_x[\chi_S(x) q(x)]$$

Attacker's best response

- Decision Problem version of the attacker's best response:

EVASION

$$\sum_j \alpha_j \phi_j(x') \le \lambda$$
$$\|x - x'\| \le k,$$



- Thm: **EVASION** is NP-Complete.
- Approaches:
  - Polynomial approximation algorithm. Suppose that c bounds the number of inputs in any basis. Algorithm approximates best response to a factor 1+ε in time poly(n,1/ε,2^c)
  - Complete branch-and-bound search
  - **Greedy heuristic**: near-optimal in practice (close to branch-and-bound); faster than alternatives)

References (evasion-robust learning)

- Xu et al. Robustness and regularization of Support Vector Machines. JMLR ’09.
- Teo et al. Convex learning with invariances. NIPS ‘07.
- Li and Vorobeychik. Feature cross-substitution in adversarial classification. NIPS ‘14.
- Li and Vorobeychik. Scalable optimization of randomized operational decisions in adversarial classification. AISTATS ‘15.
- Kantchelian et al. Evasion and hardening of tree ensemble classifiers. ICML ’16.
- Li, Vorobeychik, Chen. A general retraining framework for scalable adversarial classification. arxiv, 2016.
- Tong et al. Hardening classifiers against evasion: the good, the bad, and the ugly. arxiv, 2017.

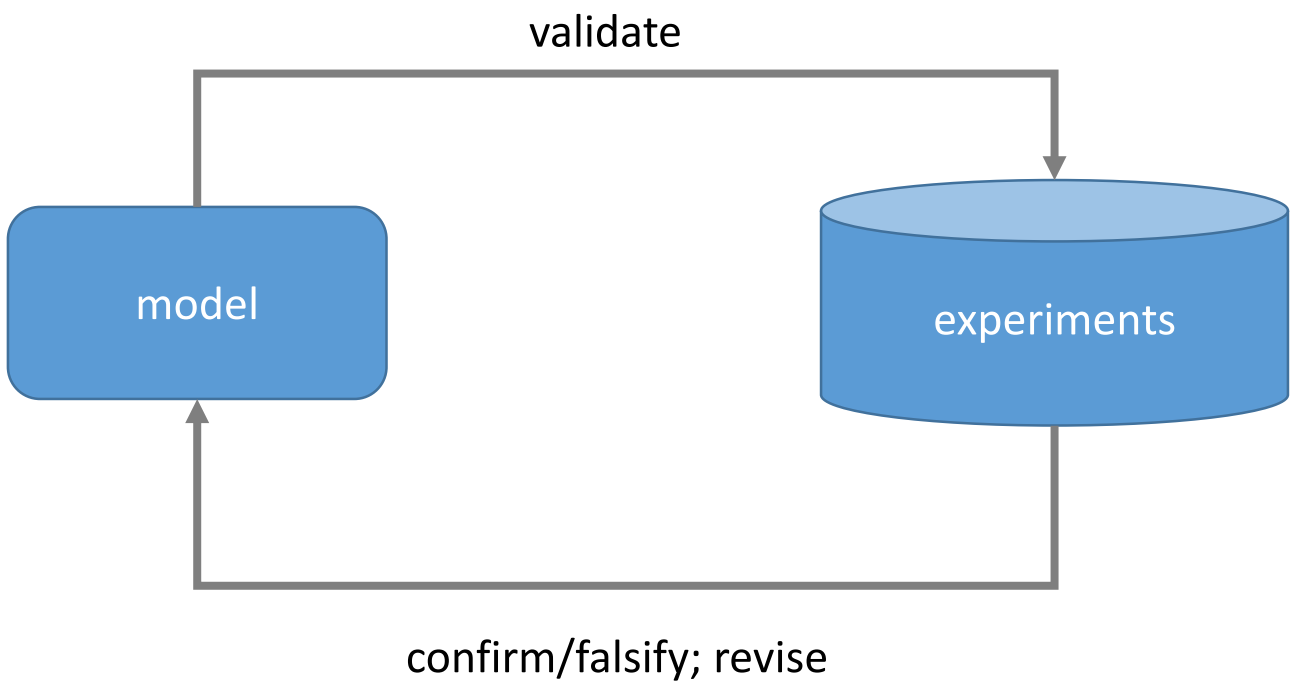
- ***Not exhaustive***
  - **Vorobeychik and Kantarcioglu, Adversarial Machine Learning book will have a more extensive bibliography**

outline

- Evasion Attacks
- Evasion-robust Classification
- Validating evasion attack models

Science and modeling

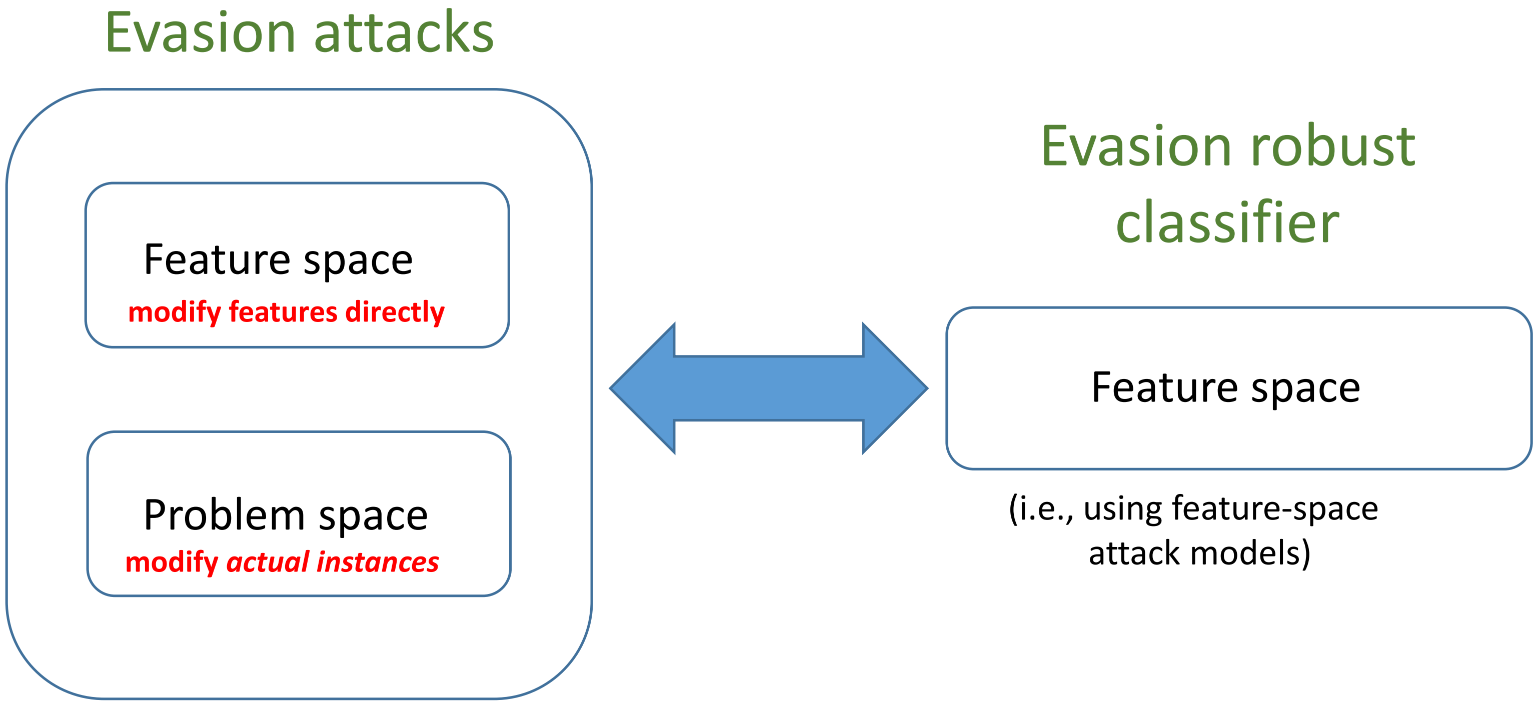
- In science, modeling is typically a process



Modeling in security

- Falsifying models:
  - All threat models are wrong; usually easy to falsify
  - But are they useful?
  - So how do you falsify a threat model in a security-meaningful way?
    - **A threat model is useful if it helps design a better defense (i.e. defense aiming to protect against this threat model)**
    - **"Better"**: against other (e.g., more concrete) attacks
  - Falsifying a threat model: showing that it is (relatively) ineffective in devising a defense

Feature space vs. problem space evasion attacks



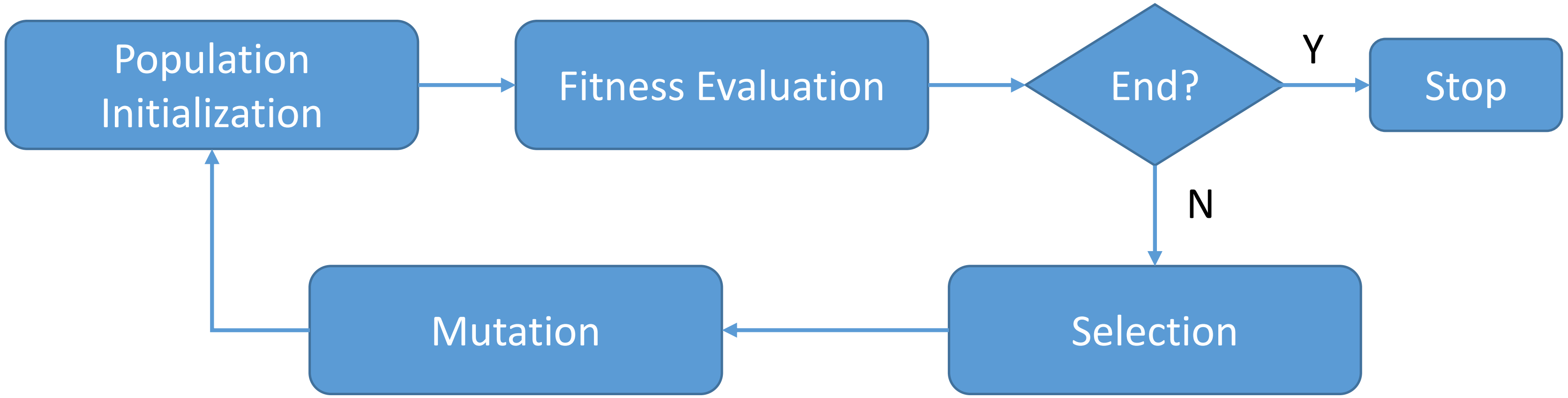
- How well do feature space evasion models represent actual attacks in problem space?

Distinction between feature space and problem space

- Problem space evasion attacks: modify actual malware source, and then check that it is still malicious ***using a sandbox***
    - Classifier then extracts features from the modified instance
    - Cannot have arbitrary feature modifications, but constraints on "feasible" attack space non-obvious and highly complex!

Pdf malware classifier
Evasion attacks in problem space

- Automated evasion in problem space (EvadeML-NDSS'16) using genetic programming + Cuckoo sandbox



Defense through retraining

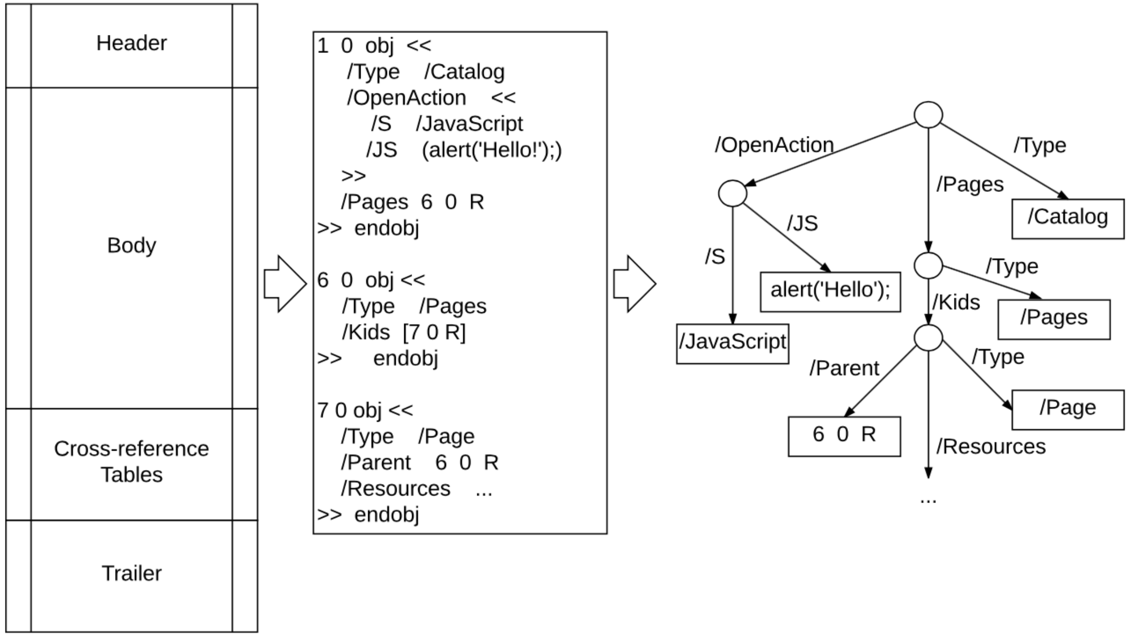
- Start with original data
- Use *any learning algorithm* to learn a model *f*
- For malicious instances, apply *any evasion method* to generate new instances *x'* to add to the dataset
- Repeat
- Stop when:
  - No new instances to add
  - Iteration limit
  - Classifier changes small between successive iterations

Experimental methodology

- **Problem space** retraining. Generating problem space adversarial instances (e.g. real-world malicious PDFs; e.g., using EvadeML), extract feature vectors, and add to the training data.
- **Feature space** retraining. Generating evasions by using mathematical evasion models in feature space (no actual malware is generated), and add the resulting feature vectors to the training data.

Case study using structure-based PDF malware classifiers

- Structure-based features using object paths within a PDF file

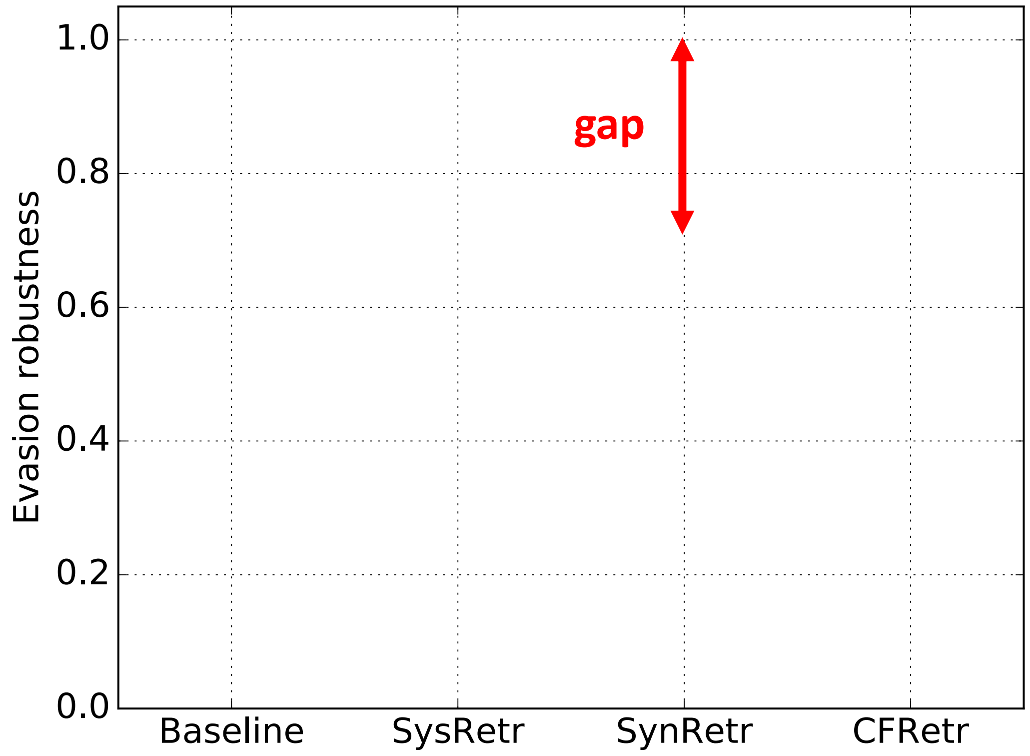


Features: existence of specific structural paths (binary)

PDF malware detector

- **Hidost**: PDF malware classifier with ~1000 structural features

Hidost



original

Problem space retraining

Feature space retraining

Limitations of feature space models

- Synthetically generated adversarial instances may in actuality NOT preserve malicious functionality. This introduce *noise* and *bias* into the retraining process.
- Realistic adversarial instances may not be produced as the evasion model may not abide by *realistic attack constraints*.
- How can we fix the model?

Classifying with conserved features

- Conserved features: features which are essentially invariant in problem space attacks.
- We identify a set of conserved features of Hidost by systematically manipulating each PDF object, checking impact on extracted features, and evaluating the corresponding maliciousness.
- This way we identified **7** conserved features, out of **1,000**

Retraining with conserved features

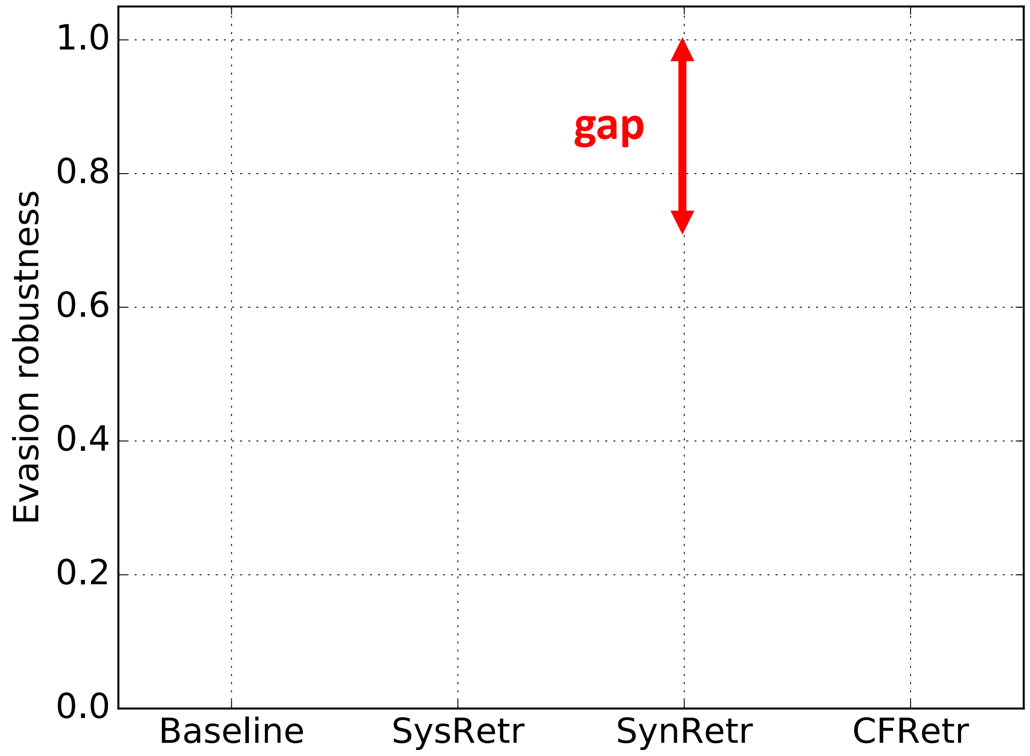
- Additional constraint: *conserved features* are preserved in evasive instances.

$$\min_{x} Q(x) = g(x) + \lambda c(x_{ideal}, x)$$

$$subject\ to \quad \boxed{x_i = x_{ideal,i}, \forall i \in S}$$

  - *S*: set of conserved features

Hidost, CF retraining



original

Problem space retraining

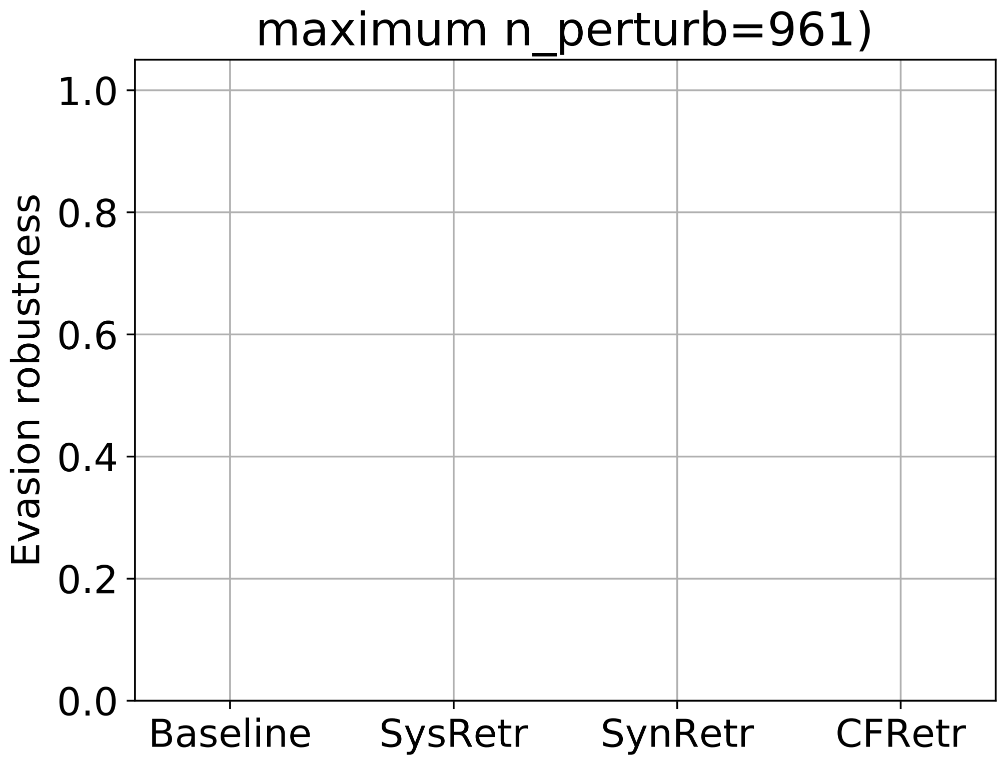
Feature space retraining

Feature space retraining with CF

What about other attacks?

- An alternative mimicry attack using Generative Adversarial Networks (MalGAN)

hidost



"Overfit" to EvadeML

Generalized robustness!

References (validation of evasion models)

- Tong et al. Hardening classifiers against evasion: the good, the bad, and the ugly. arxiv, 2017.

- **Exhaustive**