Adversarial Data Mining for Cyber Security

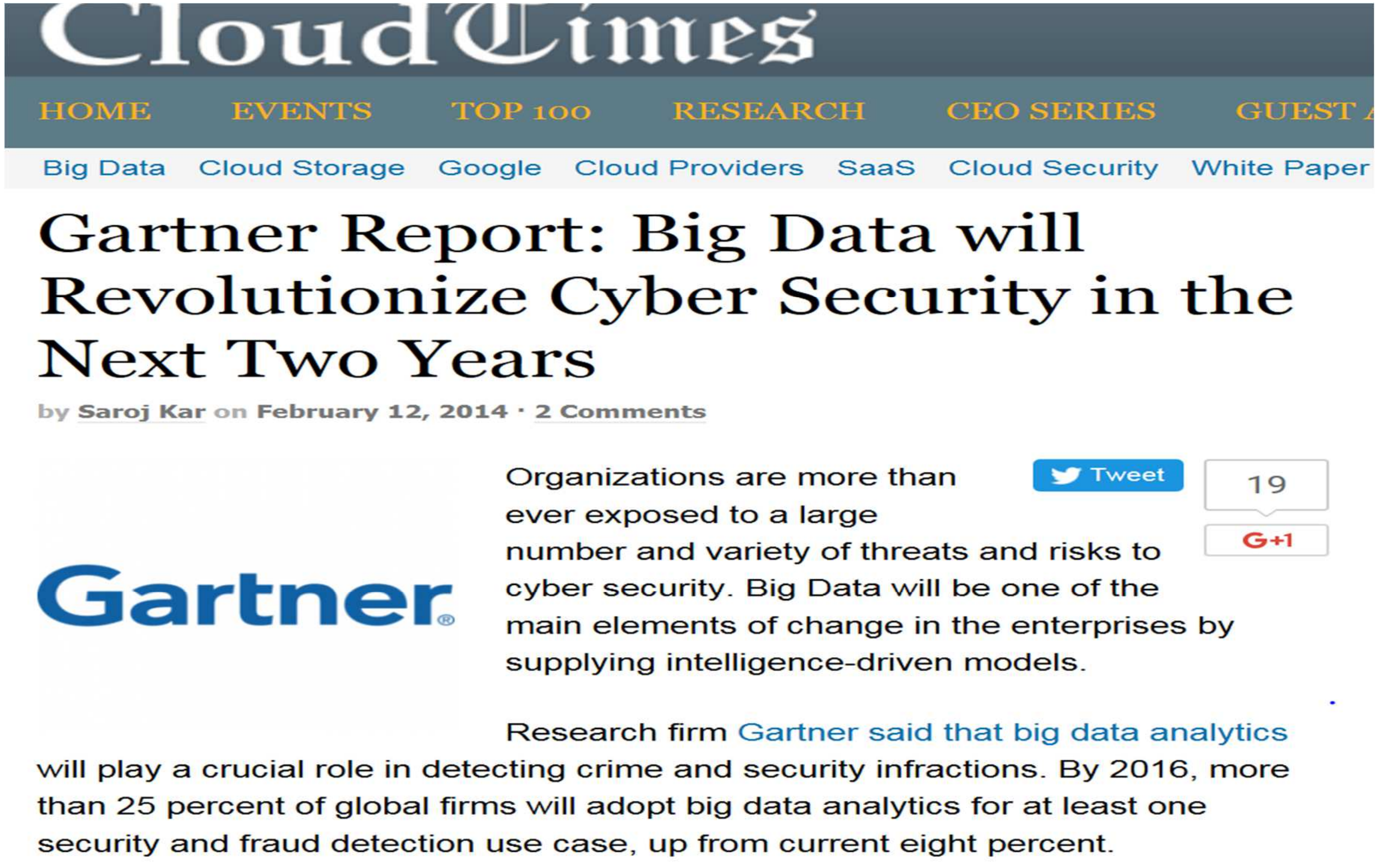
Murat Kantarcioglu*,

- Computer Science, University of Texas at Dallas

*

Special thanks to Bowei Xi (Statistics, Purdue University) and Yan Zhou (University of Texas at Dallas) for slide preparation help

Big data and Cyber Security



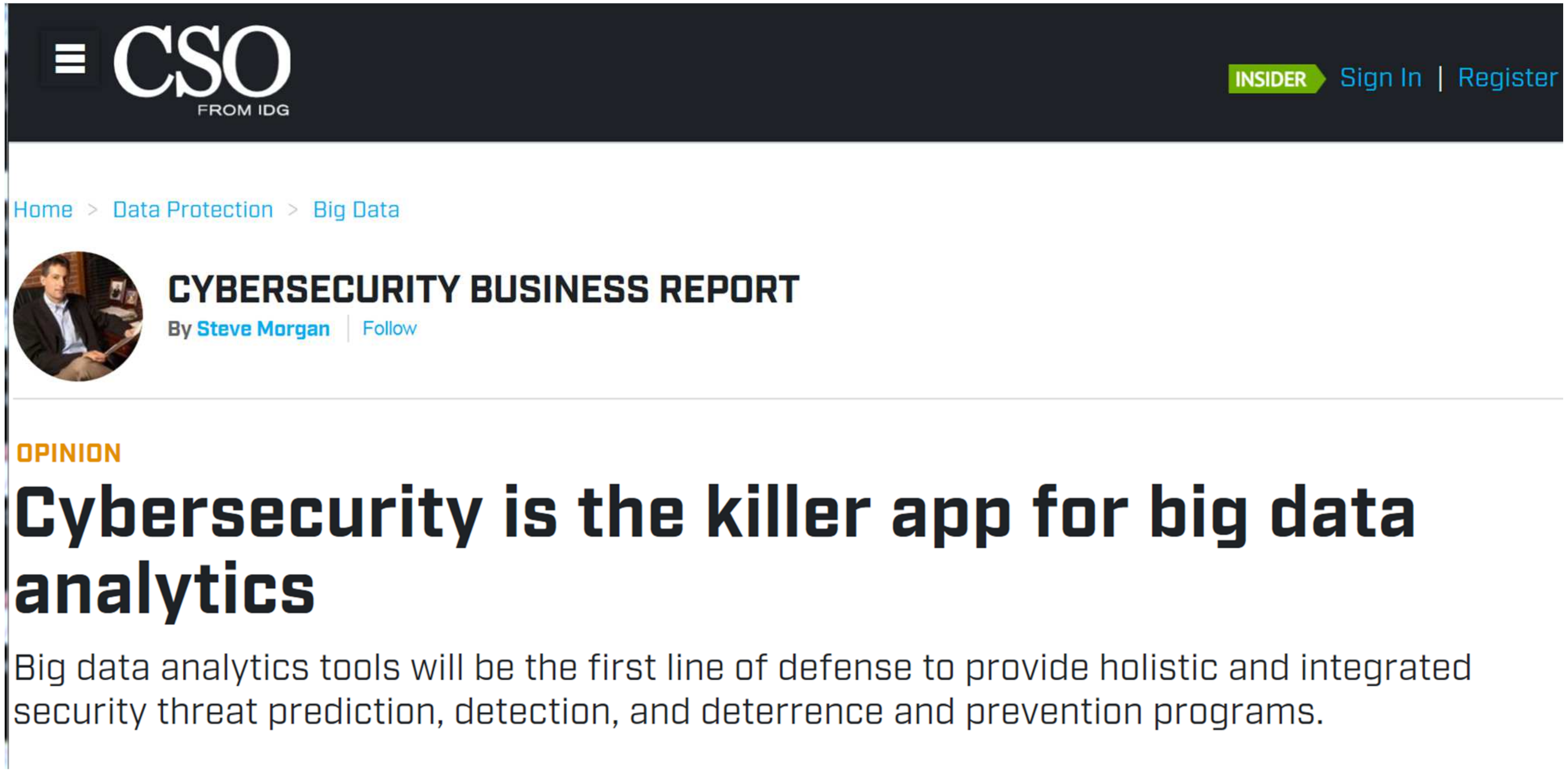
Gartner Report: Big Data will Revolutionize Cyber Security in the Next Two Years

by Saroj Kar on February 12, 2014 · 2 Comments

Organizations are more than ever exposed to a large number and variety of threats and risks to cyber security. Big Data will be one of the main elements of change in the enterprises by supplying intelligence-driven models.

Research firm Gartner said that big data analytics will play a crucial role in detecting crime and security infractions. By 2016, more than 25 percent of global firms will adopt big data analytics for at least one security and fraud detection use case, up from current eight percent.

Already many start-ups in the field.



CYBERSECURITY BUSINESS REPORT

By Steve Morgan | Follow

OPINION

Cybersecurity is the killer app for big data analytics

Big data analytics tools will be the first line of defense to provide holistic and integrated security threat prediction, detection, and deterrence and prevention programs.

More data is available for cyber security

- Malware samples
- System Logs
- Firewall Logs
- Sensor data
- …

Cyber Security is Different

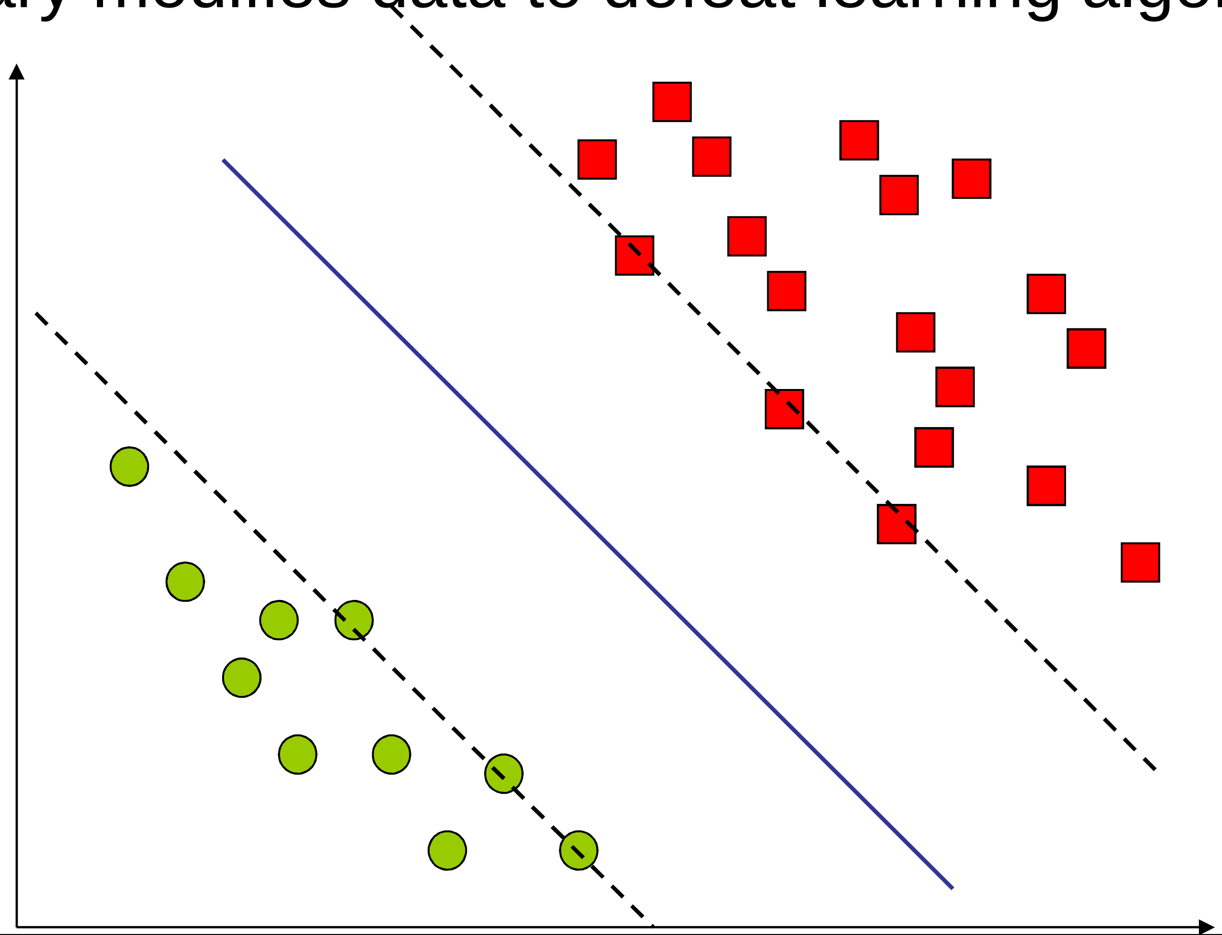
- Many adversarial learning problems in practice
  - Intrusion Detection
  - Fraud Detection
  - Spam Detection
  - Malware Detection

- Adversary adapts to avoid being detected.

- New solutions are explored to address this problem

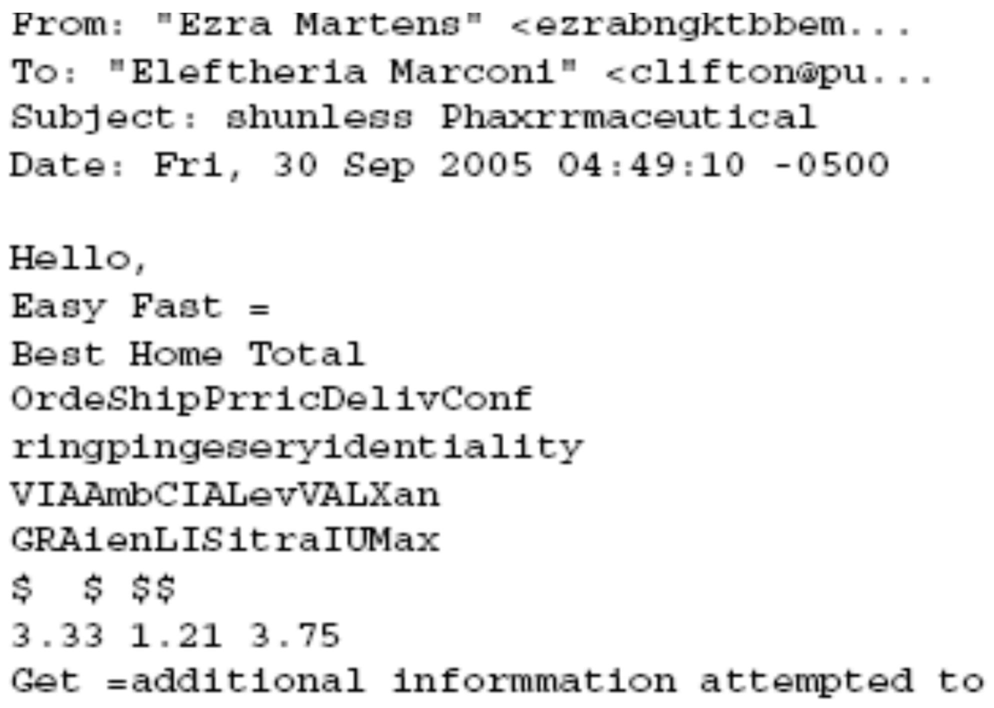
The Problem

- Violation of standard *i.i.d.* assumption
- Adversary modifies data to defeat learning algorithms



Example: Spam Filtering

- Millions way to write Viagra



```
From: "Ezra Martens" <ezrabngktbbem...
To: "Eleftheria Marconi" <clifton@pu...
Subject: shunless Phaxrrmaceutical
Date: Fri, 30 Sep 2005 04:49:10 -0500

Hello,
Easy Fast =
Best Home Total
OrdeShipPrricDelivConf
ringpingeseryidentiality
VIAAmbCIALevVALXan
GRAienLISitraIUMax
$  $ $$
3.33 1.21 3.75
Get =additional informmation attempted to
```

Understanding Adversarial Learning

- It is not concept drift
- It is not online learning
- Adversary adapts to avoid being detected
  - During training time (i.e., data poisoning)
  - During test time (i.e., modifying features when data mining is deployed)
- There is game between the data miner and the adversary

Solution Ideas

- Constantly adapt your classifier to changing adversary behavior.

- Questions??
  - How to model this game?
  - Does this game ever end?
  - Is there an equilibrium point in the game?

Agenda

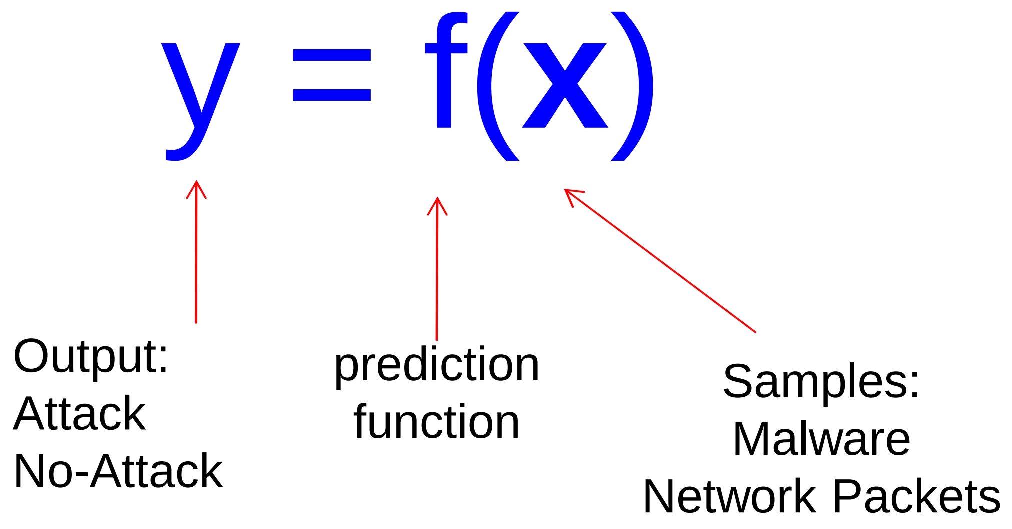
- Summary of foundational results/models to reason about learning in the presence of an active adversary
  - No proofs/ Summary of the models
  - Not all the good work could be covered ☹
- Modified techniques resistant to adversarial behavior
- Some applications of data mining for cyber security/practical attacks
- Summary/Suggestions

Foundations

Machine Learning Problems

	Supervised Learning	***Unsupervised Learning***
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

The machine learning framework

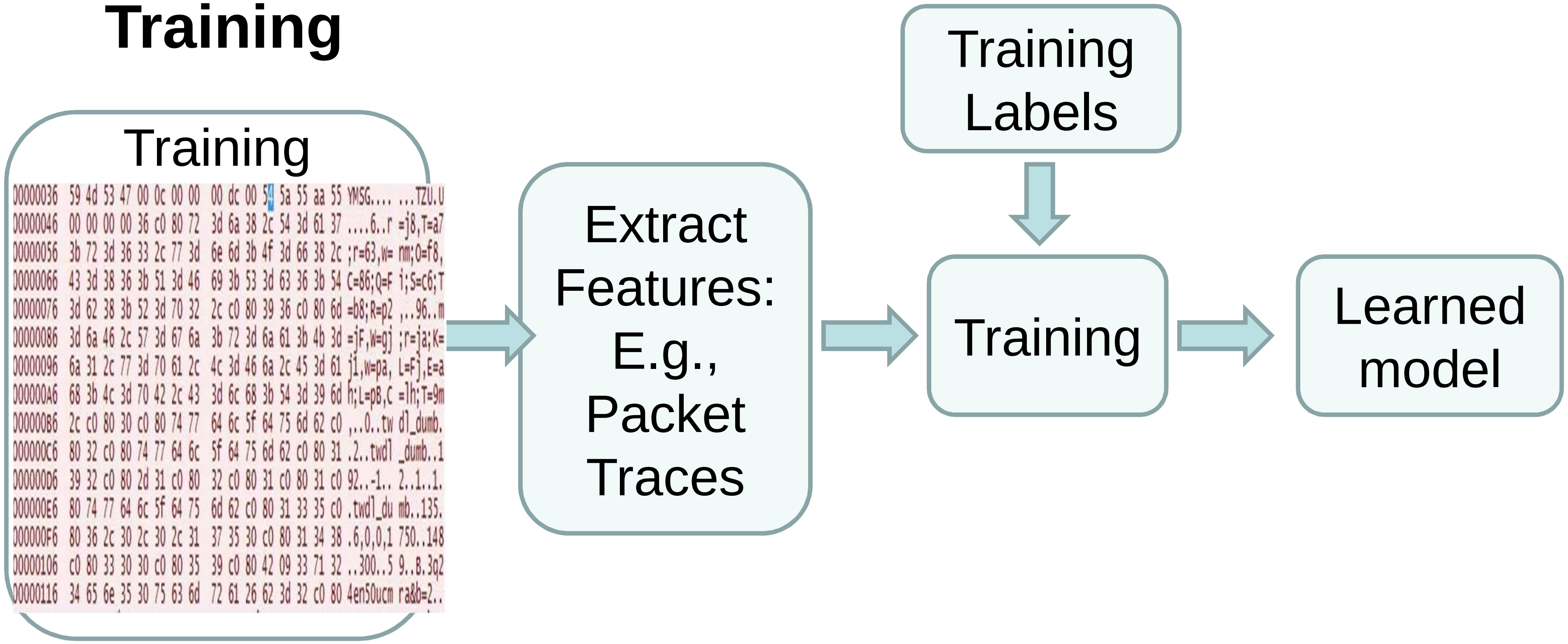


- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing some criteria on the training set
- **Testing:** apply f to a *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Slide credit: L. Lazebnik

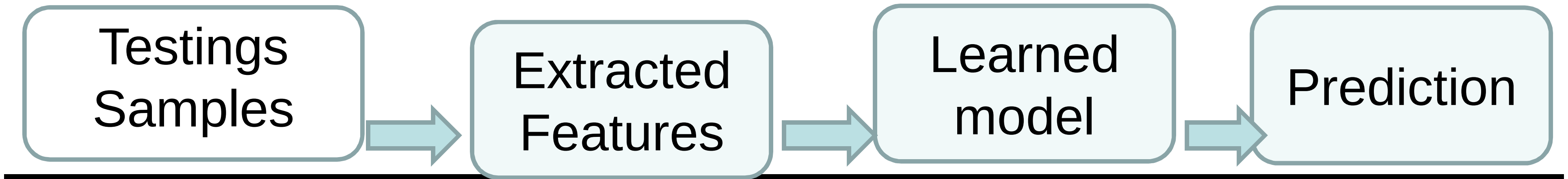
Steps

Training



```
00000036  59 4d 53 47 00 0c 00 00  00 dc 00 54 5a 55 aa 55 YMSG.... ...TZU.U
00000046  00 00 00 00 36 c0 80 72  3d 6a 38 2c 54 3d 61 37 ....6..r =j8,T=a7
00000056  3b 72 3d 36 33 2c 77 3d  6e 6d 3b 4f 3d 66 38 2c ;r=63,w= nm;O=f8,
00000066  43 3d 38 36 3b 51 3d 46  69 3b 53 3d 63 36 3b 54 C=86;Q=F i;S=c6;T
00000076  3d 62 38 3b 52 3d 70 32  2c c0 80 39 36 c0 80 6d =b8;R=p2 ,..96..m
00000086  3d 6a 46 2c 57 3d 67 6a  3b 72 3d 6a 61 3b 4b 3d =jF,W=gj ;r=ja;K=
00000096  6a 31 2c 77 3d 70 61 2c  4c 3d 46 6a 2c 45 3d 61 j1,w=pa, L=Fj,E=a
000000A6  68 3b 4c 3d 70 42 2c 43  3d 6c 68 3b 54 3d 39 6d h;L=pB,C =lh;T=9m
000000B6  2c c0 80 30 c0 80 74 77  64 6c 5f 64 75 6d 62 c0 ,..0..tw dl_dumb.
000000C6  80 32 c0 80 74 77 64 6c  5f 64 75 6d 62 c0 80 31 .2..twdl _dumb..1
000000D6  39 32 c0 80 2d 31 c0 80  32 c0 80 31 c0 80 31 c0 92..-1.. 2..1..1.
000000E6  80 74 77 64 6c 5f 64 75  6d 62 c0 80 31 33 35 c0 .twdl_du mb..135.
000000F6  80 36 2c 30 2c 30 2c 31  37 35 30 c0 80 31 34 38 .6,0,0,1 750..148
00000106  c0 80 33 30 30 c0 80 35  39 c0 80 42 09 33 71 32 ..300..5 9..B.3q2
00000116  34 65 6e 35 30 75 63 6d  72 61 26 62 3d 32 c0 80 4en50ucm ra&b=2..
```

Testing



Slide credit: D. Hoiem and L. Lazebnik

Threat Models

- Training Time Attacks:
  - Poison/ modify the training data
  - Some attacks are tailored for specific f()
- Test time/ Deployment Time Attacks
  - Attacker modifies x to x'
    - E.g., modify packet length by adding dummy bytes
    - Add good word to spam e-mail
    - Add noise to an image
  - Could be specific to f()

Learning in the Presence of Malicious Errors [1]

- Training data contains malicious noise.
- The adversary has
  - unbounded computational resource
  - knowledge of target concept, target distributions, internal states of the learning algorithm
- With probability β ($0 \leq \beta < 1/2$), the adversary gets to generate malicious errors.
- The adversary's goal is to foil the learning algorithm.

Optimal Malicious Error Rates

- Optimal malicious error rates for a class C:
  - $E_M(C)$:
    - the largest value of β that can be tolerated by any learning algorithm for C
    - Upper bound
  - $E_M^P(C)$:
    - the largest rate of malicious error that can be tolerated by a polynomial-time learning algorithm for C
    - Lower bound

Upper Bound of Malicious Error Rate

Theorem 1 *For a distinct concept class C,*

$$E_M(\mathcal{C}) < \frac{\epsilon}{1+\epsilon}$$

– To learn an ε-good hypothesis (type 1 and type 2 error rates are less than ε), a learning algorithm can only handle $\beta < \varepsilon/(1+ \varepsilon)$.

– The bound holds regardless of the time or sample complexity of the learning algorithms for C.

– The bound holds even for algorithms with unbounded computational resources.

Lower Bound of Malicious Error Rate for Polynomial-time Learning Algorithms

Theorem 2. *Let C be a polynomial time learnable concept class in the error-free model by algorithm A with sample complexity* $s_A(\varepsilon,\delta)$ (learns a ε–hypothesis with prob. at 1- δ *using* $s_A(\varepsilon,\delta)$ samples) *and let* $s = s_A(\varepsilon/8, ½)$. *We can learn C in polynomial time with an error rate of* $\beta = \Omega(\min(\frac{\epsilon}{8}, \frac{\ln s}{s}))$

– *A* is a β-tolerant Occam algorithm for *C* if it is consistent with at least $1-\varepsilon/2$ of the samples received from the faulty oracles.

Summary of [1]

- Error tolerance against data poisoning need not come at the expense of efficiency or simplicity if you have large enough data and attacker capabilities are bounded.
- Better tolerable error rates (upper bound) can be achieved using both types (+/-) of examples than positive-only or negative-only learning.
- Strong ties exist between learning with errors and data poisoning attacks.

Adversarial Classification [2]

- Data is manipulated by an adversary to increase false negatives.
  - Spam detection
  - Intrusion detection
  - Fraud detection

- Classification is considered as a game between the classifier and the adversary.
  - Both are cost-sensitive

Cost and Utility for Classifier & Adversary

- Given a training set S and a test set T,
  - CLASSIFIER
    - learn from S a classification function $y_C = C(x)$
    - V_i: cost of measuring the i^{th} feature X_i
    - $U_C(y_C, y)$: utility of classifying an instance as y_C with true class y
      - $U_C(+, -) < 0$, $U_C(-, +) < 0$, $U_C(+,+) > 0$, $U_C(-,-) > 0$
  - ADVERSARY
    - modify a *positive* instance in T from x to $x' = A(x)$
    - $W_i(x_i, x'_i)$: cost of changing the i^{th} feature from x_i to x'_i
    - $U_A(y_C, y)$: ADVERSARY's utility when the classifier classifies an instance as y_C with true class y
    - $U_A(-,+) > 0$, $U_A(+,+) < 0$ and $U_A(-,-) = U_A(+,-) = 0$

A Single-step Two Players' Game

- For computational tractability, the adversarial classification game only considers one move by each of the players.
- It also assumes that all parameters of both players are known to each other.
- Classifier is Naïve Bayes:
  - an instance x is classified positive if the expected utility of doing so exceeds that of classifying it as negative

$$\frac{P(+|x)}{P(-|x)} > \frac{U_C(-,-) - U_C(+,-)}{U_C(+,+) - U_C(-,+)}$$

Adversary’s Strategy

- Adversary’s optimal strategy:
  - Two assumptions:
    - complete Information
    - CLASSIFIER is unaware of its presence.
  - Modify features such that
    - The transformation cost is less than the expected utility.
    - The new instances is classified as negative.
  - Solve an integer LP

Classifier's Strategy

- Classifier's optimal strategy:
  - Three assumptions:
    - Adversary uses optimal strategy.
    - Training set is not tampered by Adversary.
    - The transformation cost $W_i(x_i, x'_i)$ is a semi-metric.
  - Make prediction y_C that Maximizes conditional utility:

$$U(y_C|x) = \sum_{y \in \mathcal{Y}} P(y|x) U_C(y_C, y)$$

with a post-adversary conditional probability

$$P_A(x'|+) = \sum_{x \in \mathcal{X}} P(x|+) P_A(x'|x, +)$$

Classifier Evaluation and Attribute Selection against Active Adversaries [3]

- Consider cases where the classifier is modified after observing adversaries action.
  - Spam filter rules.

- Stackelberg Games
  - Adversary chooses an action a_1
  - After observing a_1 , data miner chooses action a_2
  - Game ends with payoffs to each player

$$u_1(a_1, a_2), u_2(a_1, a_2)$$

Adversarial Stackelberg Game Formulation

- Two class problem
  - Good class, Bad class
- Mixture model

$$x = (x_1, x_2, x_3, \ldots, x_n)$$

$$p_1 + p_2 = 1$$

$$f(x) = p_1 f_1(x) + p_2 f_2(x)$$

- Adversary applies a transformation T to modify

bad class (i.e $f_2(x) \xrightarrow{T} f_2^T(x)$)

Adversarial Stackelberg Game Formulation Cont.

- After observing transformation, data miner chooses an updated classifier h
- We define the payoff function for the data miner

$$f(x) = p_1 f_1(x) + p_2 f_2^T(x)$$

$$c(T,h) = \int_{L_1^h} c_{11} p_1 f_1(x) + c_{12} p_2 f_2^T(x) dx + \int_{L_2^h} c_{21} p_1 f_1(x) + c_{22} p_2 f_2^T(x) dx$$

$$u_2(T,h) = -c(T,h)$$

- C_{ij} is the cost for classifying x to class i to given that it is in class j
- Data miner tries to minimize c(T,h)

Adversarial Stackelberg Game Formulation Cont.

- Transformation has a cost for the adversary
  - Reduced effectiveness for spam e-mails
- Let $g^T(x)$ be the gain of an element after transformation
- Adversary gains for the "bad" instances that are classified as "good"

$$u_1(T, h) = \int_{L_1^h} g^T(x) f_2^T(x)\, dx$$

Adversarial Stackelberg Game Formulation Cont.

- Given the transformation T, we can find the best response classifier(R(T)) h that minimizes the c(T,h)

$$h_T(x) = \begin{cases} \pi_1, (c_{12} - c_{22})p_2 f_2^T(x) \leq (c_{21} - c_{11})p_1 f_1(x) \\ \pi_2, \text{otherwise} \end{cases}$$

- For Adversarial Stackelberg game, subgame perfect equilibrium is:

$$T^* = \arg\max_{T \in S} (u_1(T, R(T)))$$

$$(T^*, R(T^*))$$

Adversarial Stackelberg Game Formulation Cont.

$$\begin{aligned} g_e(T) &= u_1(T, R_2(T)) \\ &= \int_{L_1^{h_T}} (g^T(x) f_2^T(x))dx \\ &= E_{f_2^T}(I_{\{L_1^{h_T}\}}(x) \times g^T(x)) \end{aligned}$$

$$T^* = \arg\max_{T \in S}(g_e(T))$$

- If the game is repeated finitely many times, after an equilibrium is reached, each party does not have incentive change their actions.

Summary [3]: Attribute Selection for Adversarial Learning

- How to choose attributes for Adversarial Learning?
  - Choose the most predictive attribute
  - Choose the attribute that is hardest to change
- Example:

Attribute	π_1	π_2	Penalty	Equilibrium Bayes Error
X_1	N(1,1)	N(3,1)	a = 1	0.16
X_2	N(1,1)	N(3.5,1)	a = 0.45	0.13
X_3	N(1,1)	N(4,1)	a = 0	0.23

- Not so good ideas!!

Stackelberg Games for Adversarial Prediction Problems [4]

- Unlike the previous research, Bruckner & Scheffer consider Stackelberg games where the *classifier* is the leader and the *adversary* is the follower.
  - *Data miner* chooses an action a_1
  - After observing a_1 , the adversary chooses action a_2
  - Game ends with payoffs to each player

$$u_1(a_1, a_2), u_2(a_1, a_2)$$

Cost Definition

- Two-players game between learner (-1) and adversary (+1).
- The costs of the two players are defined as follows:

$$\hat{\theta}_{-1}(\mathbf{w}, \dot{D}) = \sum_{i=1}^{n} c_{-1,i}\ell_{-1}(f_{\mathbf{w}}(\dot{x}_i), y_i) + \rho_{-1}\hat{\Omega}_{-1}(\mathbf{w}),$$

$$\hat{\theta}_{+1}(\mathbf{w}, \dot{D}) = \sum_{i=1}^{n} c_{+1,i}\ell_{+1}(f_{\mathbf{w}}(\dot{x}_i), y_i) + \rho_{+1}\hat{\Omega}_{+1}(D, \dot{D})$$

Stackelberg Games

1. Learner decides on w.
2. Adversary observes w and changes the data distribution.
3. Adversary minimizes its loss given w by searching for a sample D_w that leads to the global minimum of the loss

$$\mathcal{D}_{\mathbf{w}} = \left\{ \{(\dot{x}_i, y_i)\}_{i=1}^{n} : \{\dot{x}_i\}_{i=1}^{n} \in \underset{\dot{x}'_1, \ldots, \dot{x}'_n \in \mathcal{X}}{\operatorname{argmin}} \ \hat{\theta}_{+1}\left(\mathbf{w}, \{(\dot{x}'_i, y_i)\}_{i=1}^{n}\right) \right\}$$

Stackelberg Equilibrium

- Assuming that the adversary will decide for any $D \in D_w$, the learner has to choose model parameters $w*$ that minimize the learner's cost function θ_{-1} for any of the possible reactions $D \in D_w$ that are optimal for the adversary:

$$\mathbf{w}^* \in \underset{\mathbf{w} \in \mathbb{R}^m}{\operatorname{argmin}} \max_{\dot{D} \in \dot{\mathcal{D}}_{\mathbf{w}}} \hat{\theta}_{-1}(\mathbf{w}, \dot{D})$$

- An action $w*$ that minimizes the learner's costs and a corresponding optimal action $D \in D_{w*}$ of the adversary are called a Stackelberg equilibrium.

Find Stackelberg Equilibrium [4]

Finding Stackelberg Equilibrium

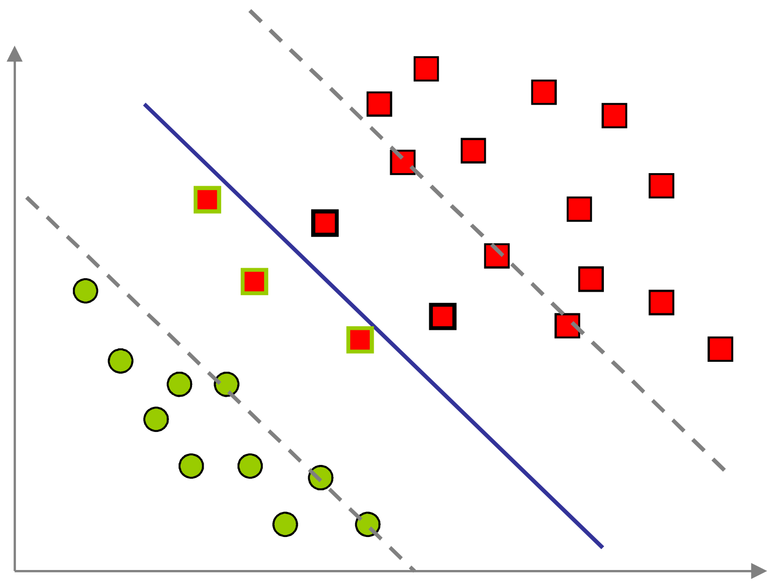
$$\min_{\mathbf{w}\in\mathbb{R}^m} \max_{\forall i:\dot{x}_i\in\mathcal{X}} \hat{\theta}_{-1}(\mathbf{w}, \{(\dot{x}_i, y_i)\}_{i=1}^n)$$
$$\text{s.t.} \quad \{\dot{x}_i\}_{i=1}^n \in \operatorname*{argmin}_{\dot{x}'_1,\dots,\dot{x}'_n\in\mathcal{X}} \hat{\theta}_{+1}(\mathbf{w}, \{(\dot{x}'_i, y_i)\}_{i=1}^n)$$

Stackelberg equilibrium is applicable when
(1.) the adversary is rational;
(2.) the predictive model is known to the adversary.

TECHNIQUES

Adversarial support vector machine learning [5]

- Support Vector machines try to find the hyperplane that has the highest possible separation margin.



Adversarial Attack Models

- Free-range attack
  - Adversary can move malicious data anywhere in the domain

$$c_f(x_{.j}^{\min} - x_{ij}) \le \delta_{ij} \le C_f(x_{.j}^{\max} - x_{ij})$$

- Targeted attack
  - Adversary can move malicious data closer to a target point

$$0 \le (x_{ij}^t - x_{ij})\delta_{ij} \le C_\xi \left(1 - C_\delta \frac{\left|x_{ij}^t - x_{ij}\right|}{\left|x_{ij}\right| + \left|x_{ij}^t\right|}\right)(x_{ij}^t - x_{ij})^2$$

Adversarial SVM Risk Minimization Model

SVM risk minimization model: free-range attack

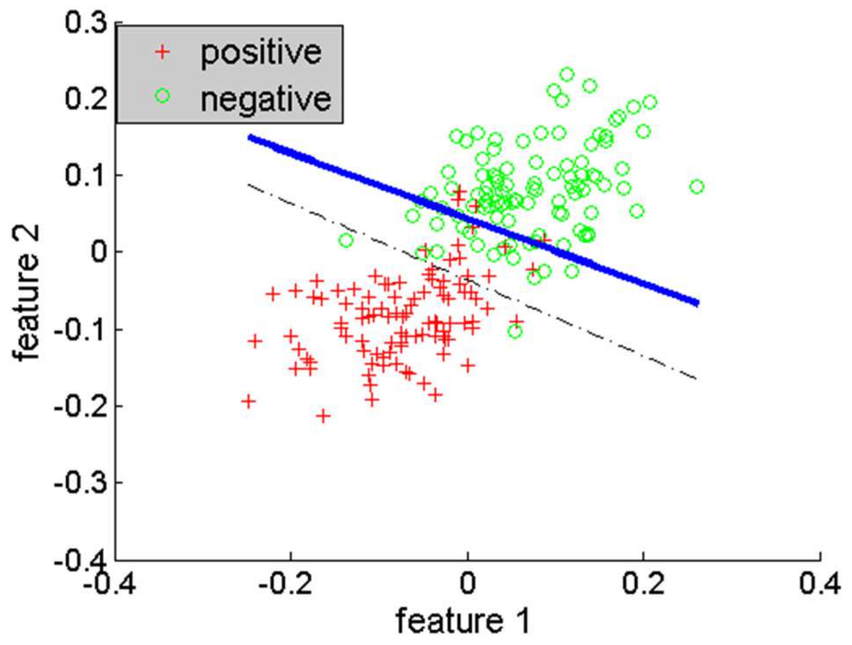
$$
\begin{array}{ll}
\underset{w,b,\xi_i,t_i,u_i,v_i}{\operatorname{argmin}} & \frac{1}{2}||w||^2 + C\sum_i \xi_i \\
s.t. & \xi_i \geq 0 \\
 & \xi_i \geq 1 - y_i \cdot (w \cdot x_i + b) + t_i \\
 & t_i \geq \sum_j C_f \left(v_{ij}(x_j^{max} - x_{ij}) - u_{ij}(x_j^{min} - x_{ij})\right) \\
 & u_i - v_i = \frac{1}{2}(1 + y_i)w \\
 & u_i \succeq 0 \\
 & v_i \succeq 0
\end{array}
$$

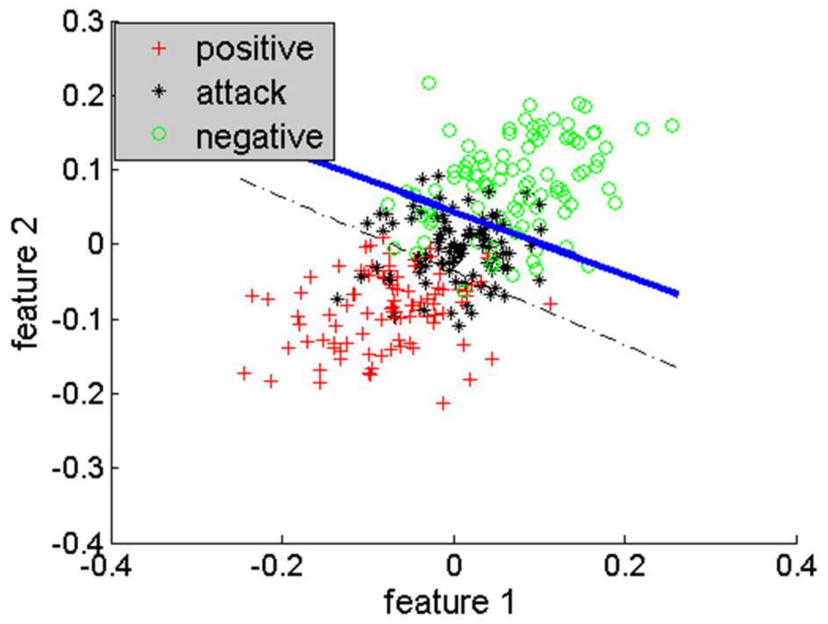
Adversarial SVM Risk Minimization Model cont.

SVM risk minimization model: targeted attack

$$
\begin{array}{ll}
\underset{w,b,\xi_i,t_i,u_i,v_i}{\text{argmin}} & \frac{1}{2}||w||^2 + C\sum_i \xi_i \\
s.t. & \xi_i \geq 0 \\
 & \xi_i \geq 1 - y_i \cdot (w \cdot x_i + b) + t_i \\
 & t_i \geq \sum_j e_{ij} u_{ij} \\
 & (-u_i + v_i) \circ (x_i^t - x_i) = \frac{1}{2}(1 + y_i)w \\
 & u_i \succeq 0 \\
 & v_i \succeq 0
\end{array}
$$

AD-SVM Example:





black dashed line is the standard SVM classification boundary, and
the blue line is the Adversarial SVM (ADV-SVM) classification boundary

Summary of [5]

- AD-SVM solves a convex optimization problem where the constraints are tied to adversarial attack models
- AD-SVM is more resilient to modest attacks than other SVM learning algorithms

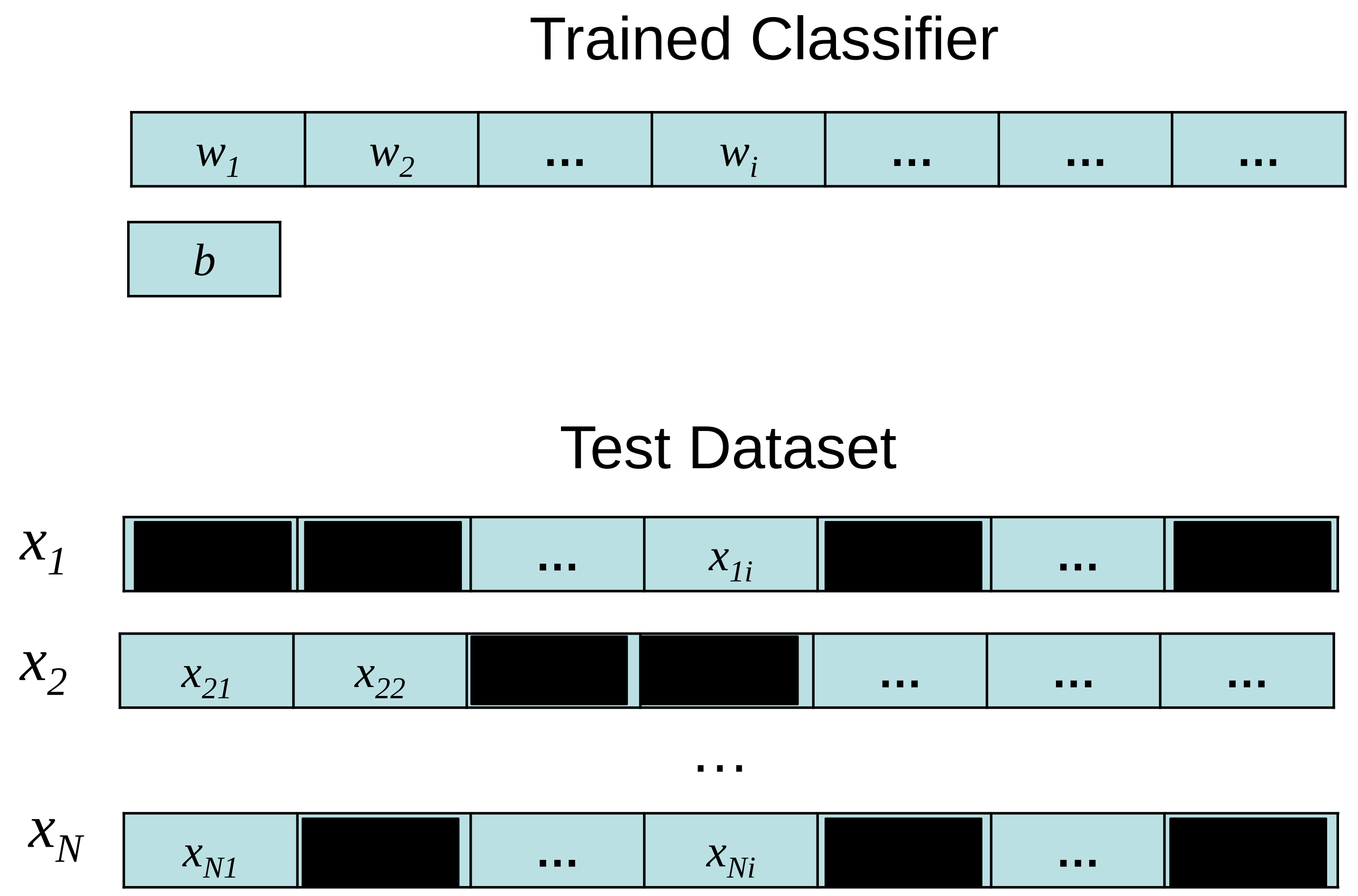
Learning to Classify with Missing and Corrupted Features [7]

- Goal: devising classifiers which are robust to classification phase noise
  - Instances drawn *i.i.d.* from some

$$\mathcal{X} \times \{\pm 1\}, \quad \mathcal{X} \subseteq \mathbb{R}^n$$

  - Linear margin-based classifiers
  - A clean, uncorrupted training data is available for learning a classifier $<w,b>$

Problem Setting



Problem Setting cont.

- Adversary's power must be reasonably bounded for learning to be possible.
- Suppose each feature j has a fixed feature value $v_j \geq 0$.
- Assumption: the adversary must leave intact a subset J in $\{1, \ldots, n\}$ of features such that

$$V(J) := \sum_{j \in J} v_j \geq P$$

where P specifies noise tolerance.

LP Formulation

- Worst-case "empirical risk" on training set:

$$\frac{1}{m}\sum_{i=1}^{m}\left[\!\left[\min_{J : V([n]\setminus J)\leq N} y_i\left(b+\textstyle\sum_{j\in J} w_j x_{i,j}\right) \leq 0 \right]\!\right]$$

- An SVM-like formulation:

$$\begin{aligned}
\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{m\gamma}\textstyle\sum_{i=1}^{m}\xi_i \\
\text{s.t.} \quad & \forall\, i\in[m] \quad \forall J : V([n]\setminus J)\leq N \\
& \qquad y_i\left(b+\textstyle\sum_{j\in J} w_j x_{i,j}\right) \geq \frac{\gamma V(J)}{P} - \xi_i \\
& \forall\, i\in[m]\ \xi_i \geq 0\,, \quad \|\mathbf{w}\|_\infty \leq C\ .
\end{aligned}$$

Problem: exponential growth of the constraint set

A Compact LP Formulation

- With a duality transform, a compact $O(mn)$ constraint set is obtained:

$$
\begin{aligned}
\min \quad & \frac{1}{m\gamma}\sum_{i=1}^{m}\xi_i \\
\text{s.t. } \forall\, i\in[m] \quad & P\lambda_i - \sum_{j=1}^{n}\alpha_{i,j} + y_i b \ \geq -\xi_i \\
\forall\, i\in[m]\ \forall\, j\in[n] \quad & y_i w_j x_{i,j} - \frac{\gamma v_j}{P} \ \geq \ \lambda_i v_j - \alpha_{i,j}\ , \\
\forall\, i\in[m]\ \forall\, j\in[n] \quad & \alpha_{i,j} \geq 0\ , \\
\forall\, i\in[m] \quad & \lambda_i \geq 0 \ \text{ and } \ \xi_i \geq 0\ , \\
& \|\mathbf{w}\|_\infty \leq C\ ,
\end{aligned}
$$

An online-to-batch algorithm is developed to learn the average hypothesis.

Adversarial Learning: Practice and Theory [8]

Problem: Content-based spam filtering

- Practice: good word attacks
  - Passive attacks
  - Active attacks
- Theory: ACRE learning

Passive Attacks

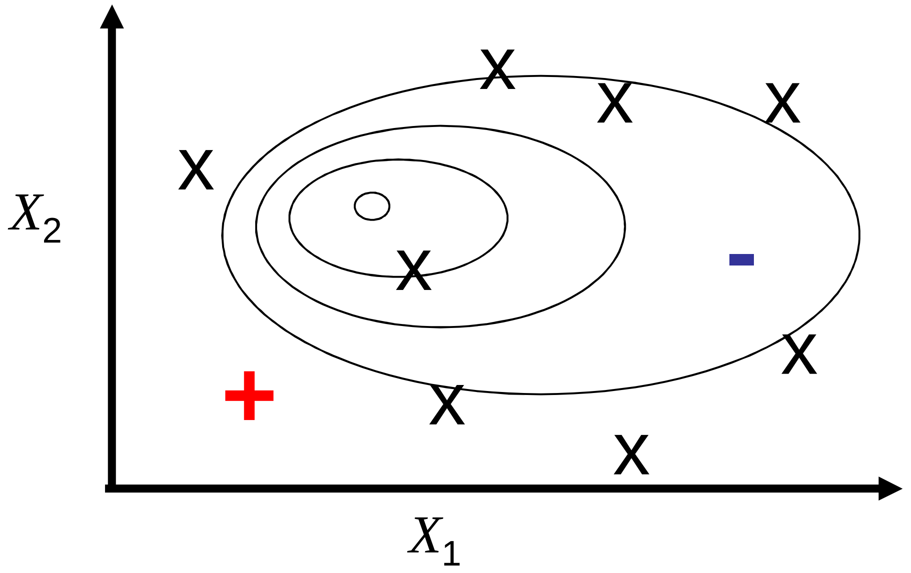
- Common heuristics
  - Random dictionary words
  - Most frequent English words
  - Highest ratio: English frequency/spam frequency

Active Attacks

- Learn which words are best by querying the spam filter.
- First-N: Find n good words using as few queries as possible
- Best-N: Find the best n words

Adversarial Classifier Reverse Engineering (ACRE)

- **ACRE** k-learnable algo.**:** minimize $a(\mathbf{x})$ subject to $c(\mathbf{x}) = -1$ within a factor of k, given:



  - –the adversarial cost function a(**x**)
  - –One positive and one negative example, $\mathbf{x}^+$ and $\mathbf{x}^-$
  - –A polynomial number of membership queries

ACRE *k*-learnability of Linear Classifiers

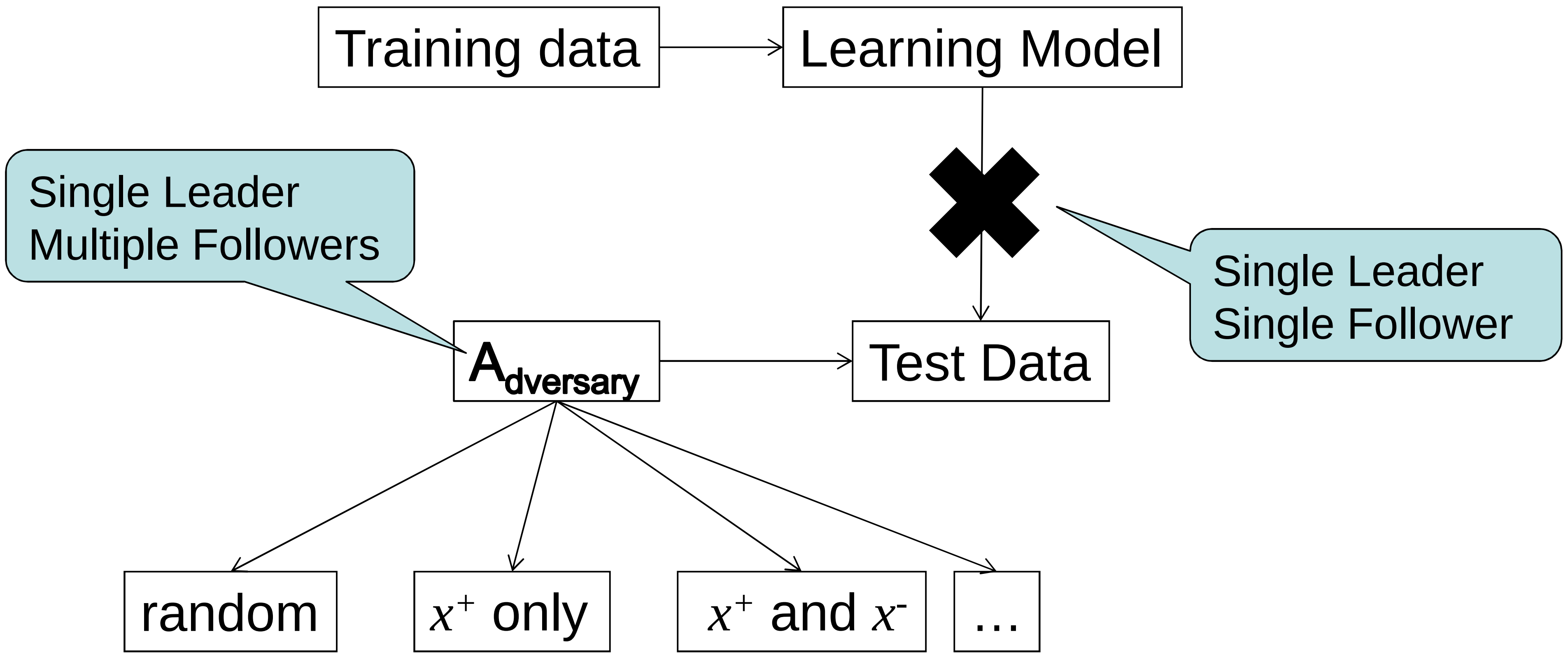
- Linear classifiers with *continuous* features are ACRE (1+ε)-learnable under linear cost functions.

- Linear classifiers with *boolean* features are ACRE 2-learnable under uniform linear cost functions

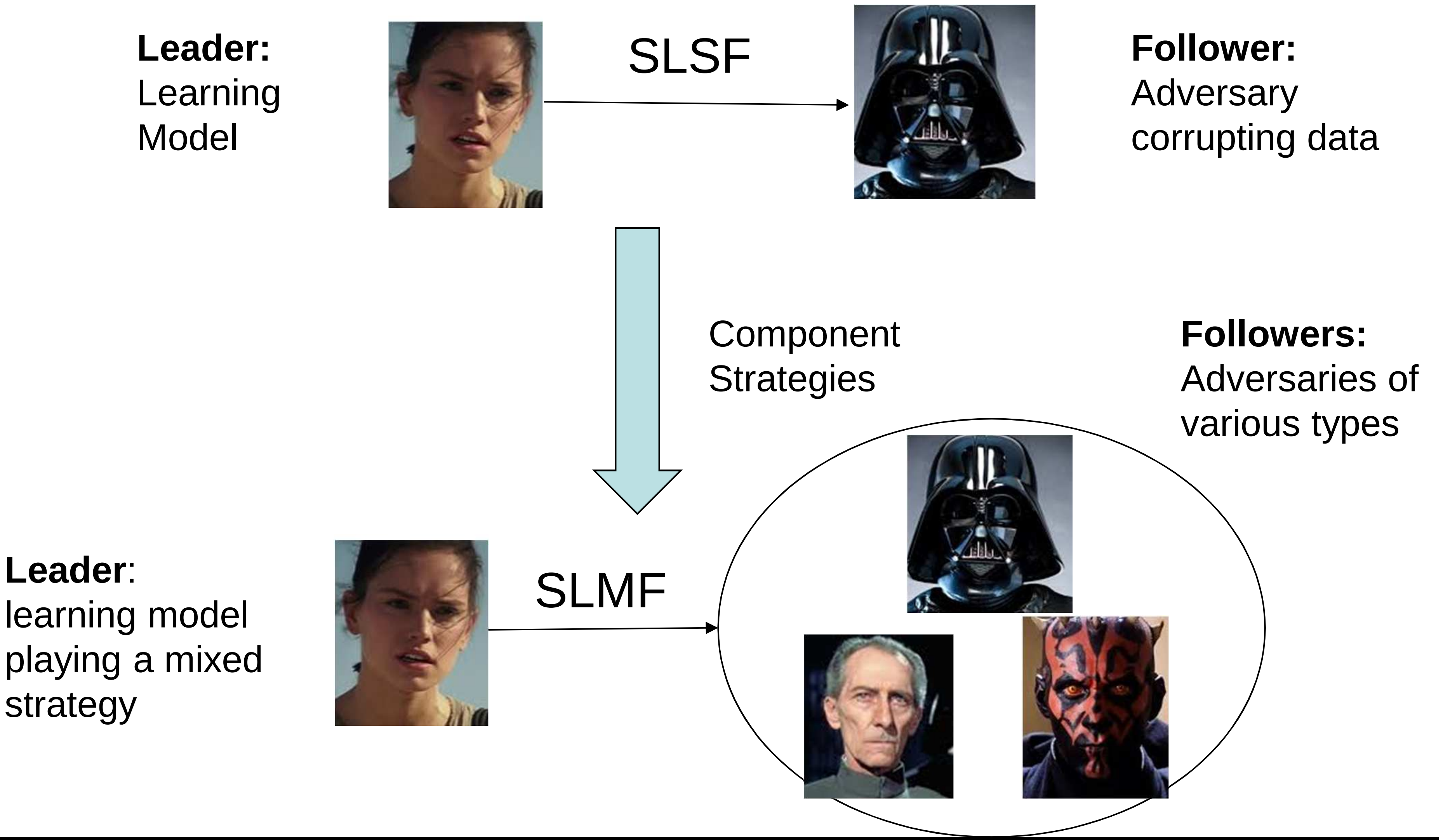
Modeling Adversarial Learning as Nested Stackelberg Games [6]

- Existing adversarial learning approaches
  - A two-player game
    - Zero-sum, Nash, Stackelberg
  - AD-SVM, AD-RVM, AD-HME
  - handle a single adversary of one type
- A more challenging problem:
  - Multiple adversaries of various types

Adversarial Learning: Multiple Adversaries of Various Types



Nested Stackelberg Game Framework


Leader: Learning Model
SLSF
Follower: Adversary corrupting data
Component Strategies
Followers: Adversaries of various types
Leader: learning model playing a mixed strategy
SLMF

Single Leader Single Follower Stackelberg Game

- Learner commits its strategy that is observable to the adversary
- Adversary plays its optimal strategy
  - Maximize learner's loss
  - Minimize adversary's loss

$$\boxed{\begin{array}{ll} \underset{w^*}{\operatorname{argmin}}\ \underset{\delta_x^*}{\operatorname{argmax}} & L_l(w,x,\delta_x) \\ s.t. & \delta_x^* \in \underset{\delta_x}{\arg\min}\, L_f(w,x,\delta_x) \end{array}}$$

SLMF Bayesian Stackelberg Game

Problem Definition:

Given the payoff matrices R^l and R^f of the leader and the m followers of n different types, find the leader's optimal mixed strategy.

•All followers know the leader's strategy when optimizing their rewards.

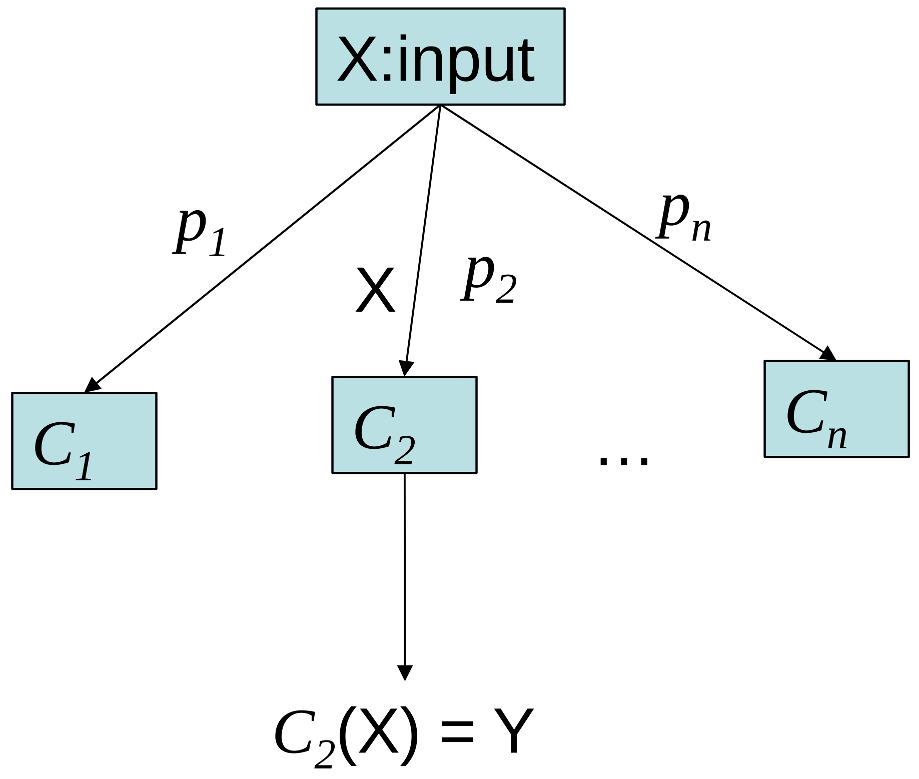
•The leader's pure strategies consist of a set of generalized linear learning models $\langle\varphi(x),w\rangle$.

•The followers' pure strategies include a set of vectors performing data transformation $x \longrightarrow x + \Delta x$.

SLMF Bayesian Stackelberg Game

- The leader makes its decision prior to the followers' decisions.
- The leader does not know the exact type of the adversary while solving its optimization problem.
- The followers play their optimal responses to maximize the payoffs given the leader's strategy.

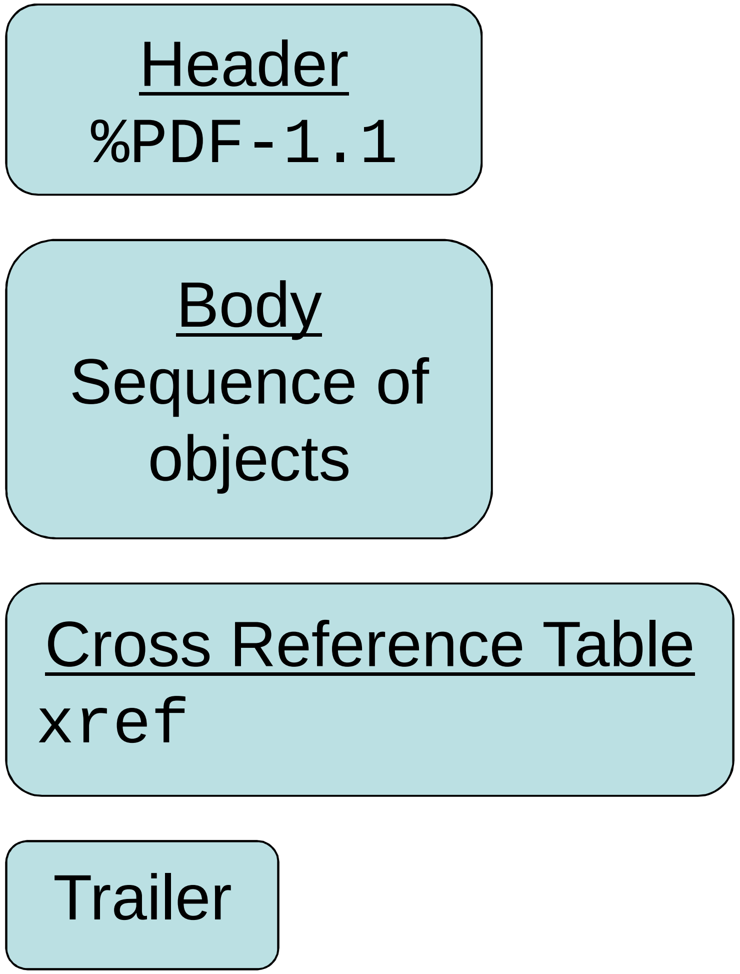
Mixed Strategy Classification



APPLICATIONS

Malicious PDF Detection using Metadata and Structural Features [10]

- PDF Basics
  - Tree structure
  - Root note: /Catalog
  - Other valid elements are In the downward path of /Catalog



Malicious PDF Exploitation

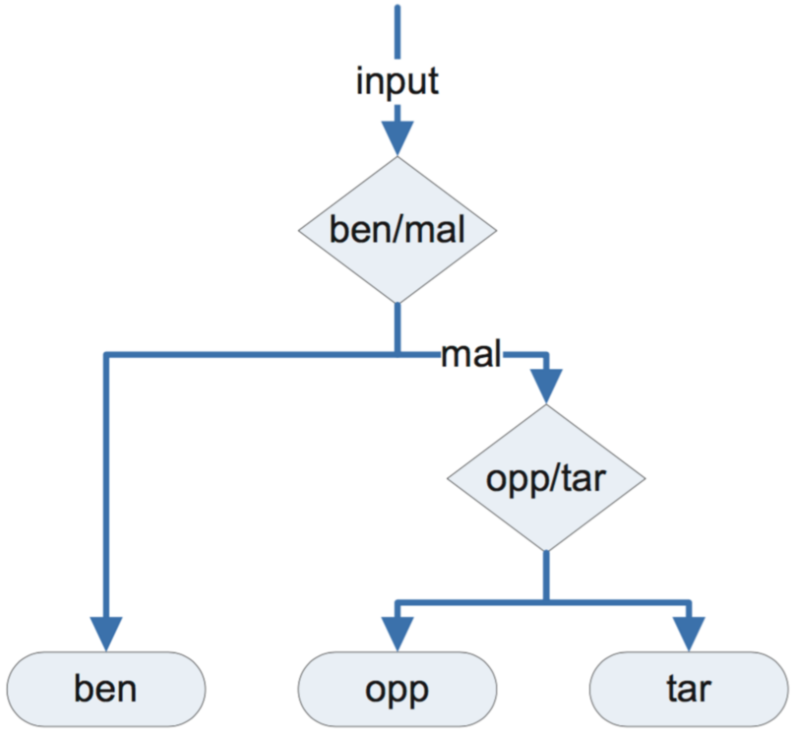
- PDF Document Exploitation
  - Malicious PDFs may contain the complete malware payload or small size code for downloading other malware components
  - Suspicious elements
    - Javascript
    - Embedded PDFs
    - Malformed objects
    - Malicious patterns
    - Encryption
    - Suspicious actions: /Actions, /OpenAction, /Names, et. al.

Feature Extraction

- Static analysis on features based on document structure and metadata.
  - works well even on encrypted documents each object/stream is encrypted individually in PDF, leaving structure and metadata to be extracted the same as normal documents.

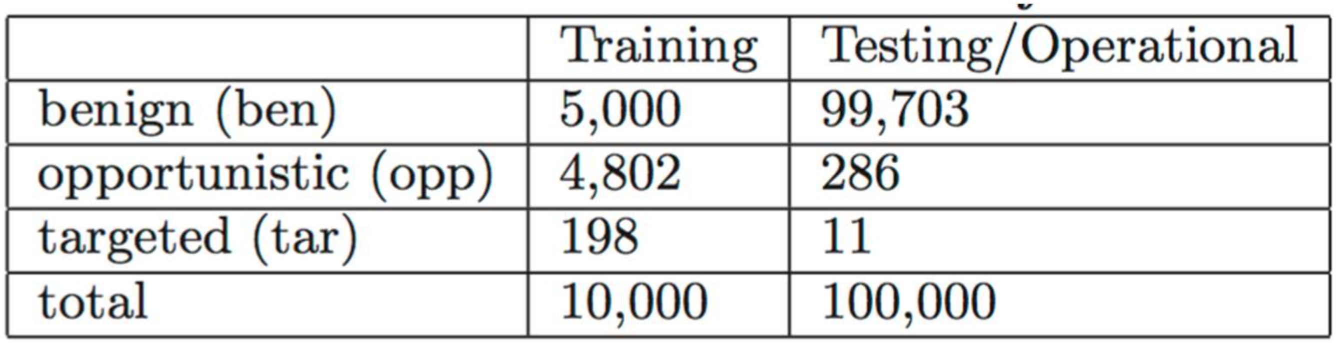
Dual Classifier

- Dual classifier
  - One differentiates *benign* from *malicious.*
  - The other classifier differentiates *opportunistic* from *targeted* malicious documents.



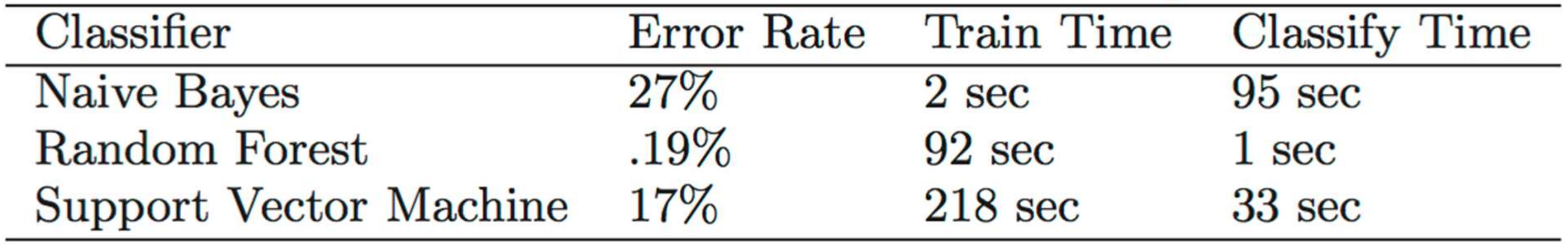
Data & Classifier Performance

- *Contagio* & *Operational* datasets
  - Subsample of *Contagio* for training
  - Test classifier on *Operational*



	Training	Testing/Operational
benign (ben)	5,000	99,703
opportunistic (opp)	4,802	286
targeted (tar)	198	11
total	10,000	100,000

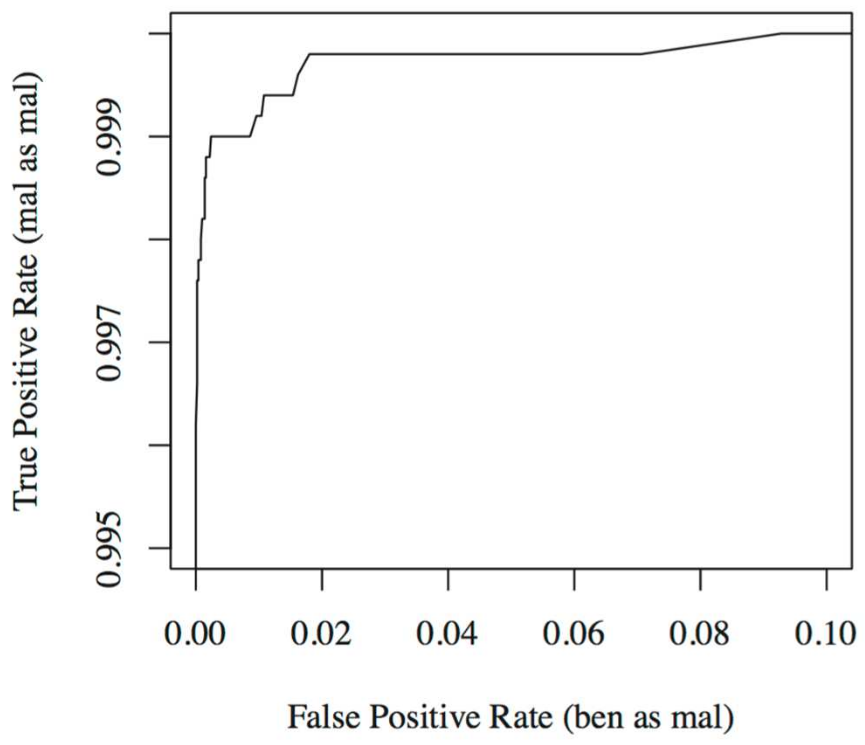
- 10-fold cross-validation on training data



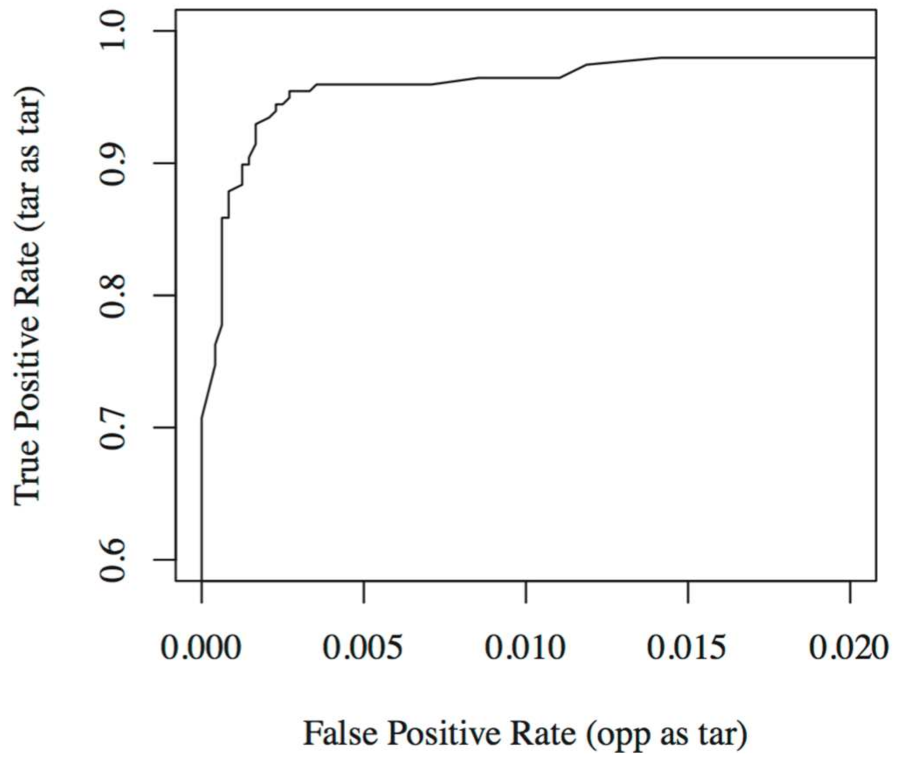
Classifier	Error Rate	Train Time	Classify Time
Naive Bayes	27%	2 sec	95 sec
Random Forest	.19%	92 sec	1 sec
Support Vector Machine	17%	218 sec	33 sec

Classification & Detection Performance

ROC for Training Set (ben/mal)



ROC for Training Set (opp/tar)

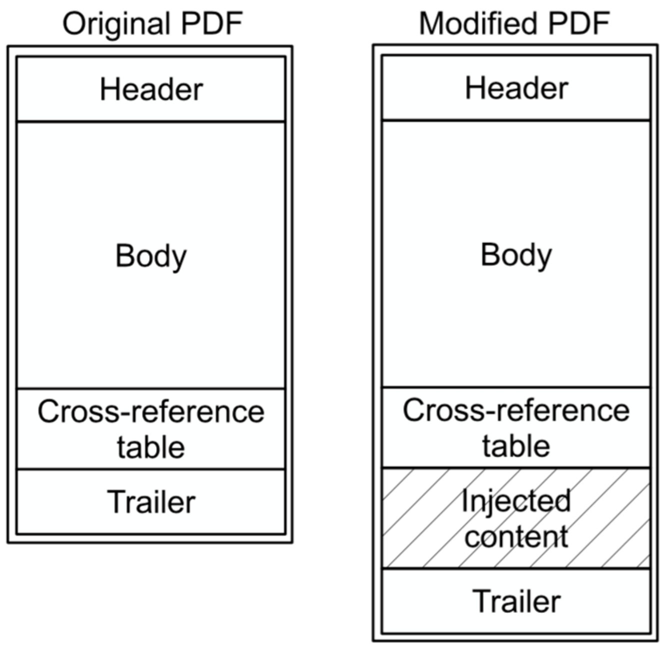


Practical Evasion of a Learning-Based Classifier: A Case Study [11]

Attack on previous work!

- Investigate a real learning-based system—PDFRATE
  - Random Forest classifier
    - Not resilient to malicious noise
    - Periodic retraining is not implemented in the system.
  - The attacker modifies the submitted PDF file, with its malicious functionality intact, and decrease the probabilistic score returned by PDFRATE.
    - Insert dummy content into PDF files

Modification of PDF Docs



PDF readers jump from *Trailer* directly to the *Cross-reference table*, skipping injected content completely.

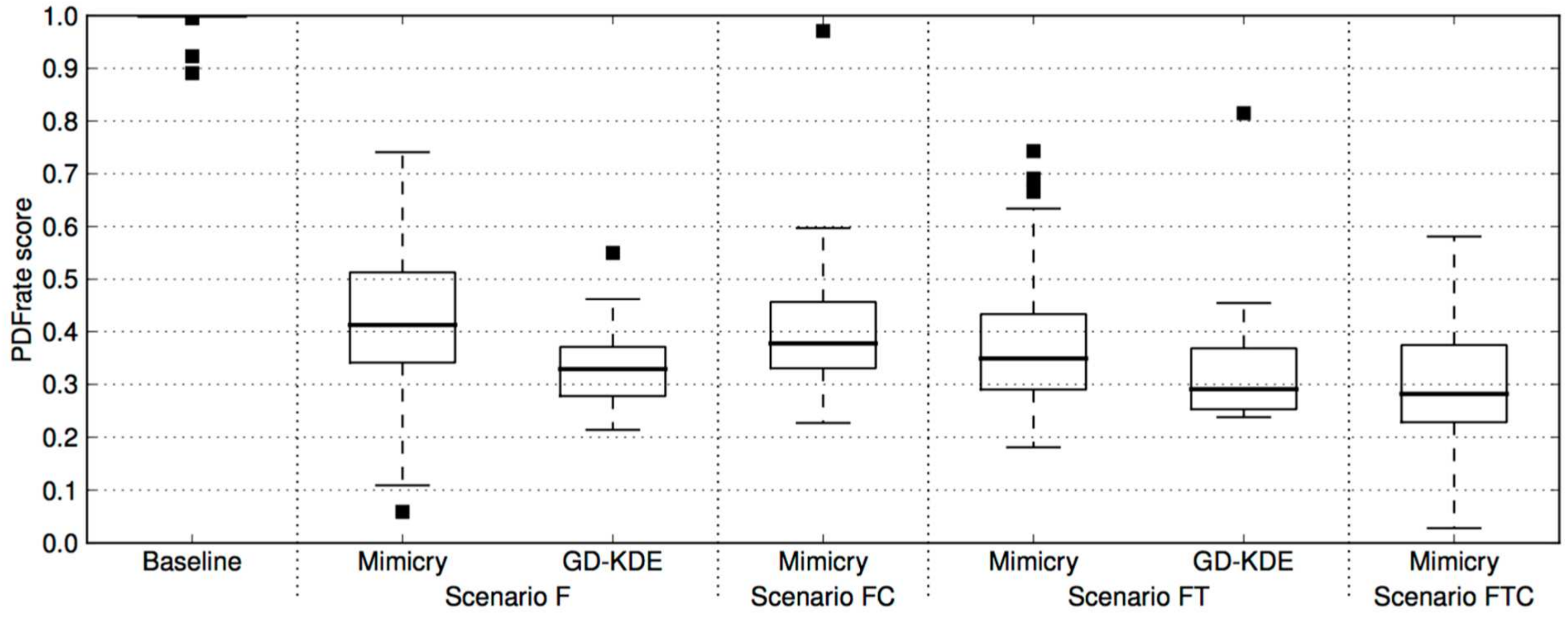
Attacks

- Mimicry Attack
  - transform a malicious sample so that it mimics a chosen benign sample as much as possible.

- Gradient Descent and Kernel Density Estimation (GD- KDE) Attack
  - require the knowledge of a specific learned model and a set of benign samples
  - only applicable to differentiable classifiers, such as SVM, artificial neural network

Results

- Baseline: results before attacks, all but 3 receives 100% PDF Score.
- Except for mimicry F, 75% of the attacks would be classified as benign if a 50% threshold over classification scores were used for decision making.



Defensive Measures

- Vaccination defense
  - modify a fraction of malicious samples in the training dataset in such a way that they are more similar to *expected* attack samples.
  - Only effective against *correctly anticipated* attacks

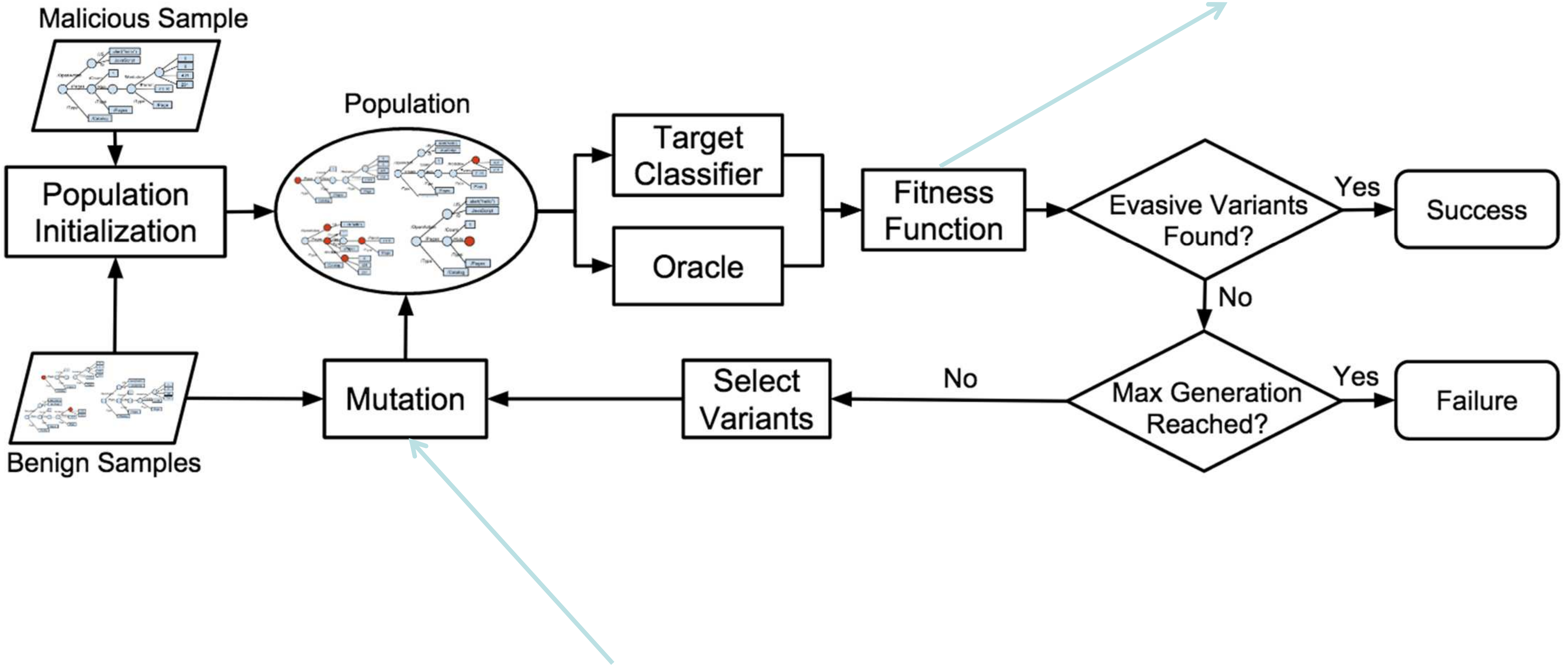
Automatically Evading Classifiers [15]

- The key idea of this work is to find an evasive variant of a malicious sample that preserves the malicious behavior but is classified as benign by the classifier.
  - Do not assume the adversary has any detailed knowledge of the classifier or the features it uses, or can use targeted expert knowledge to manually direct the search for an evasive sample.
  - Query classifiers to get the classification score for variants, assuming the classifiers do not adapt to submitted variants.

- Results are reported from experiments against two PDF malware classifiers, PDFrate (random forest based) and Hidost (SVM specifically targeting evasive attacks).

Finding Evasive Samples

$$fitness_{pdfrate}(x) = \begin{cases} 0.5 - pdfrate(x) & oracle(x) = 1 \\ LOW\_SCORE & oracle(x) = 0 \end{cases}$$

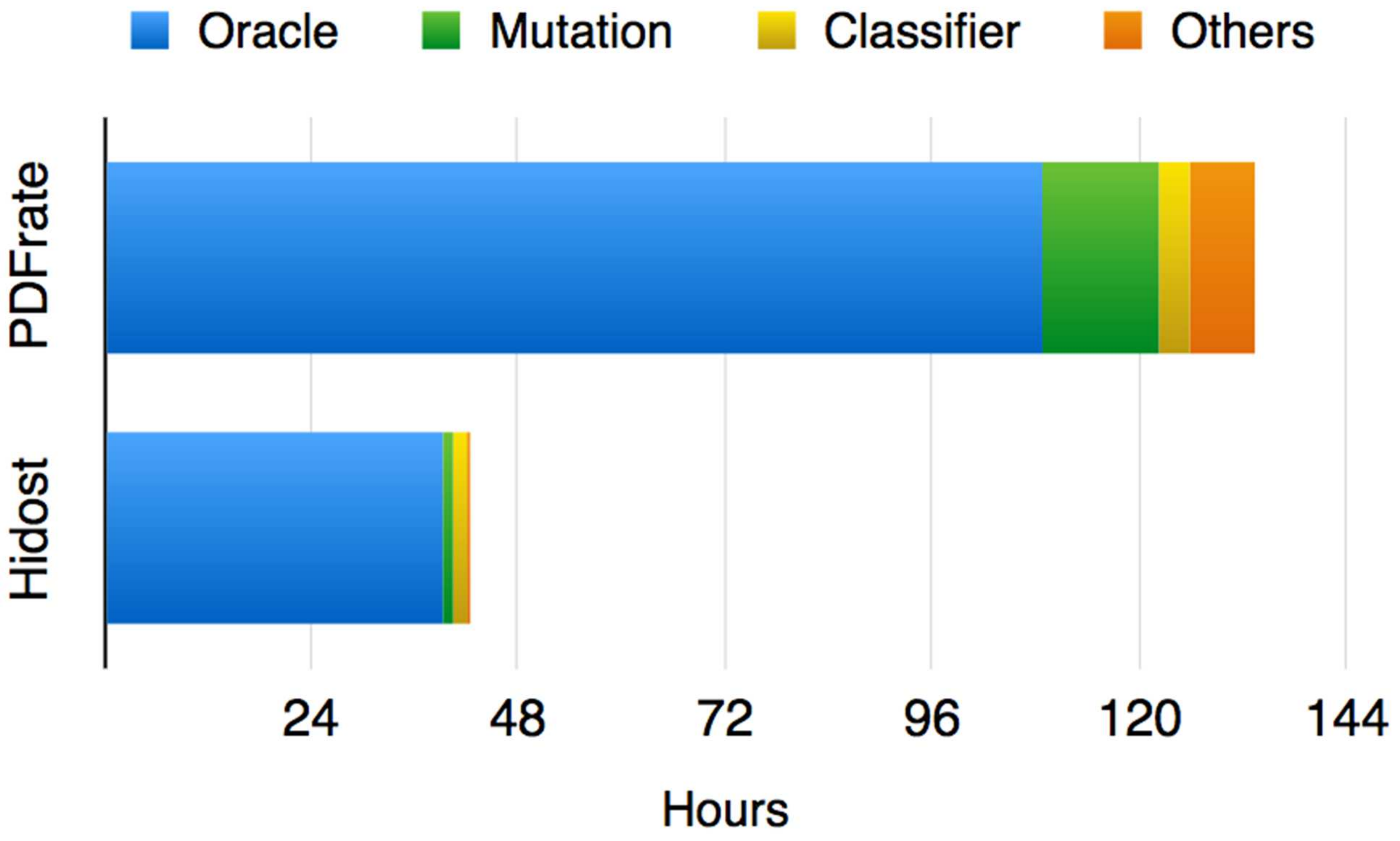


Insertion, deletion, or replacement of an object

Evaluation

- 500 malicious samples and 179 benign samples.
- After approximately one week of execution, the algorithm found 72 effective mutation traces that generated 16,985 total evasive variants for the 500 malware seeds (34.0 evasive variants per seed in average), achieving 100% evasion rate in attacking PDFrate.
- The experiment of evading Hidost took around two days to execute. Although Hidost was designed specifically to resist evasion attempts, 100% evasion rate was achieved, generating 2,859 evasive samples in total for 500 seeds (5.7 evasive samples per seed in average).

Evaluation (cont.)



Defenses

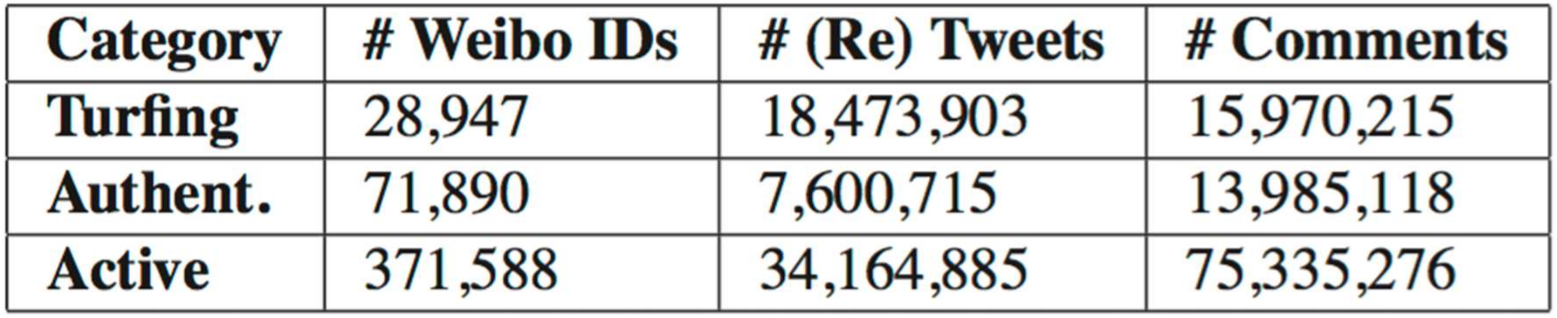
- Information Hiding and Randomization
  - hide the classification scores from the users or adding random noise to the scores
- Adapting to Evasive Variants
- Defeating Overfitting
- Selecting Robust Features

Practical Adversarial Detection of Malicious Crowdsourcing Workers [14]

- Investigating robustness of machine learning based approaches to detecting adversaries in crowdsourcing
  - Envision attack
  - Poisoning attack

Experimental Setup

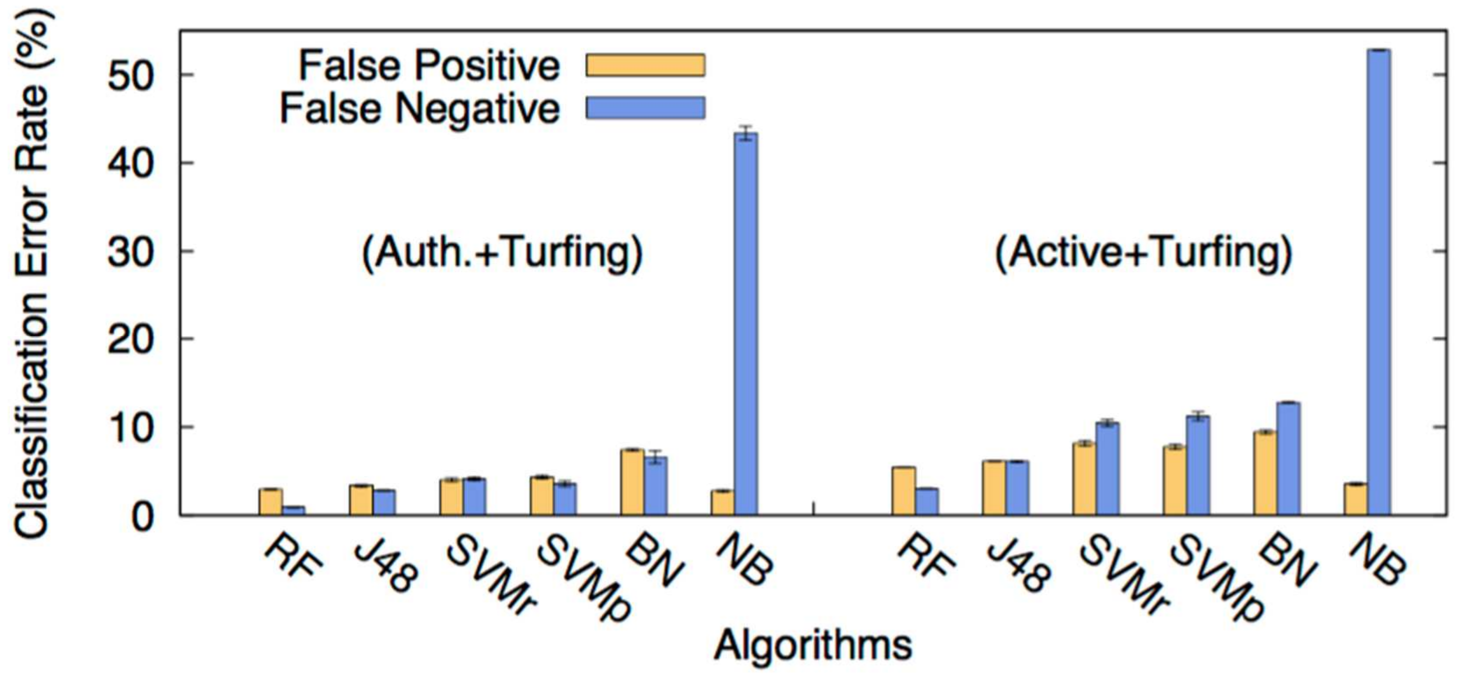
- **Datasets**
  - Extract from Sina Weibo, China’s microblogging network
    - 28,947 crowdturfing workers
    - 71890 authenticated users
    - 371,588 active users with at least 50 followers and 10 tweets
  - Classify these accounts using SVMs, Bayesian, Decision Trees and Random Forests.



Category	# Weibo IDs	# (Re) Tweets	# Comments
Turfing	28,947	18,473,903	15,970,215
Authent.	71,890	7,600,715	13,985,118
Active	371,588	34,164,885	75,335,276

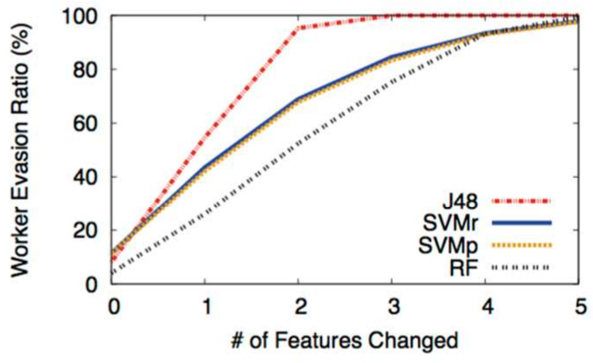
Classifier Performance

- *Authenticated+Turfing* Dataset:
  - 28K turfing accounts, 28K randomly sampled “authenticated” users.
- *Active+Turfing* Dataset:
  - 28K turfing accounts, 28K randomly sampled “active” users.

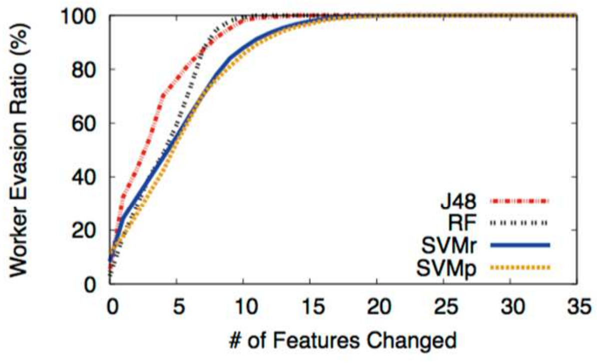


Impact of Evasion Attacks

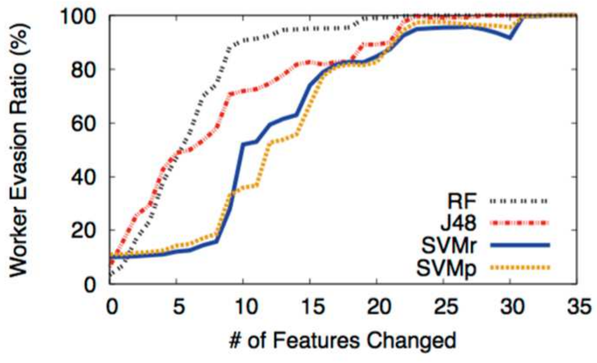
- *Optimal Evasion Attack:*
  - *Per-worker optimal evasion:* exhaustive search for optimal data modification
  - *Global evasion:* exhaustive search for global optimal strategy
  - *Feature-aware evasion:* alter the (known) most important features



(a) Per-worker Optimal Evasion



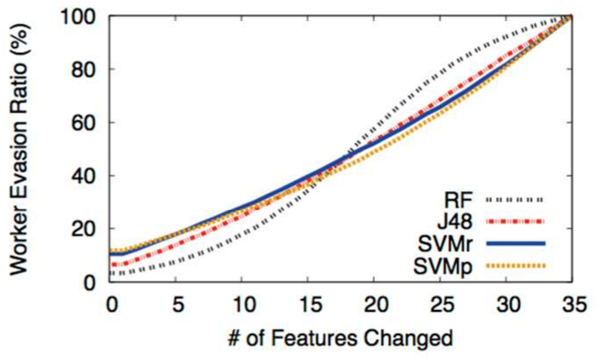
(b) Global Optimal Evasion



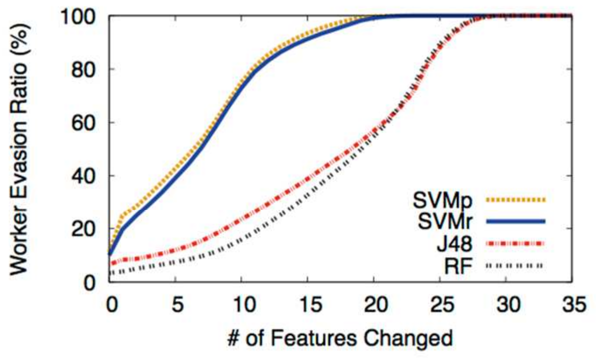
(c) Feature Importance Aware Evasion

Impact of Evasion Attacks (cont.)

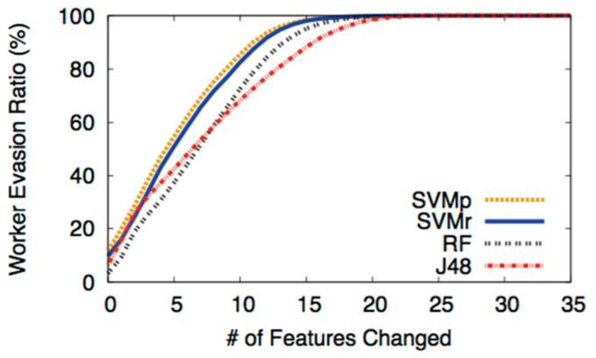
- *Practical Evasion Attack:*
  - *Random evasion*
  - *Value distance-aware evasion*
  - *Distribution distance-aware evasion*



(a) Random Evasion Strategy (Random)



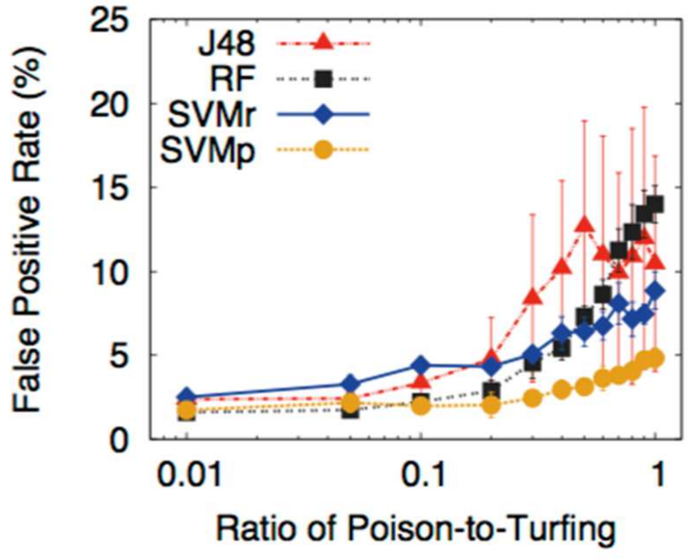
(b) Value Distance Aware Strategy (VD)



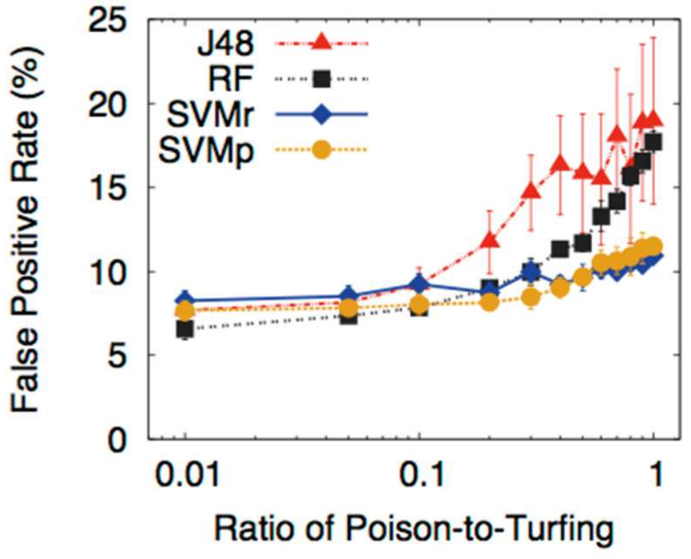
(c) Distribution Distance Aware Strategy (DD)

Impact of Poisoning Attacks

- Training data used to build ML classifiers is contaminated.
  - Poisoning training dataset by *injecting random* normal user samples to the turfing class.



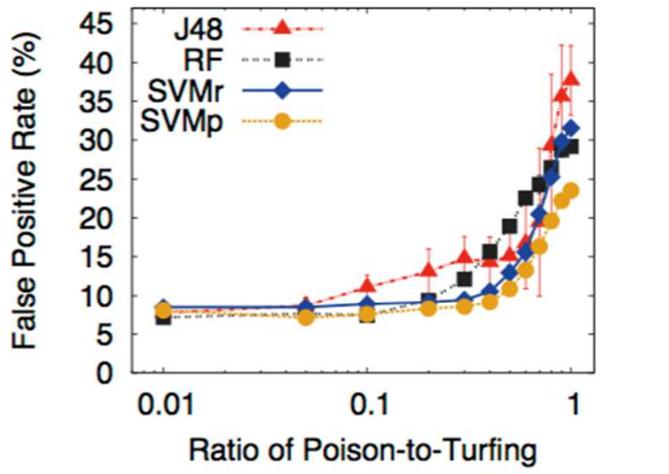
(a) Professional Workers



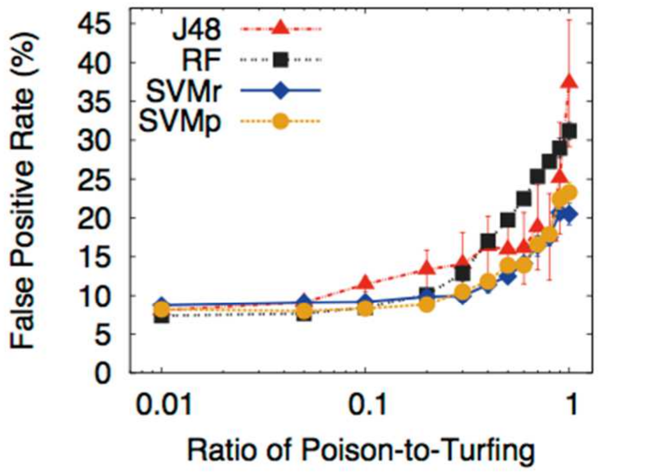
(b) All Workers

Impact of Poisoning Attacks (cont.)

- Training data used to build ML classifiers is contaminated.
  - Adversaries *inject specific type of normal users* to the turfing class (all workers).



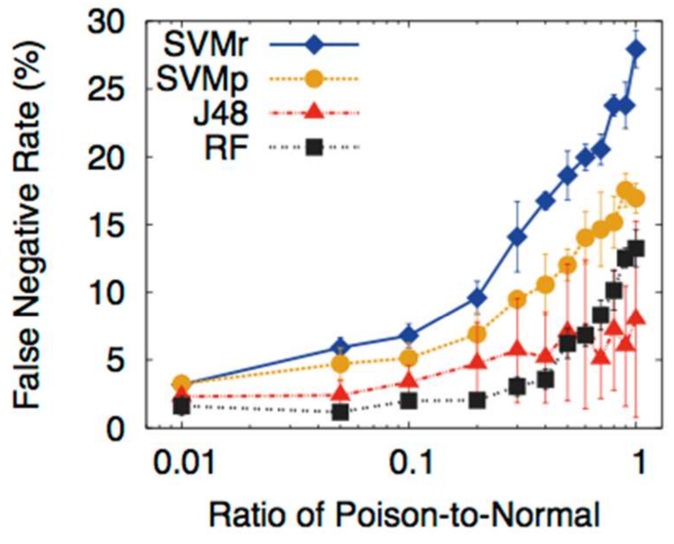
(a) Injecting Accounts with > 50% tweets commented



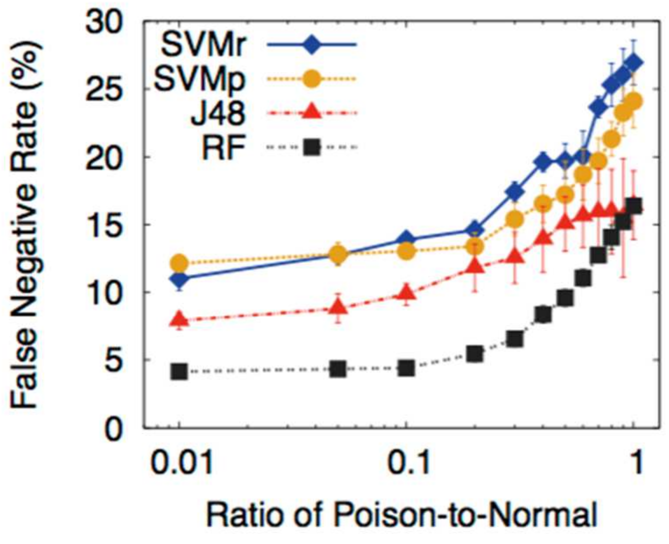
(b) Injecting Accounts with < 150 followers

Impact of Poisoning Attacks (cont.)

- Training data used to build ML classifiers is contaminated.
  - Poisoning training dataset by adding turfing samples to normal class .



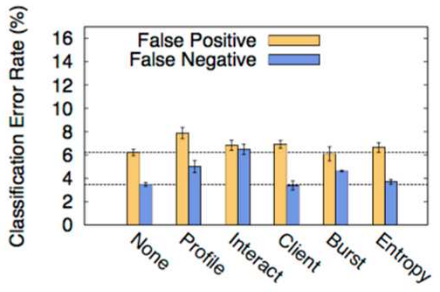
(a) Professional Workers



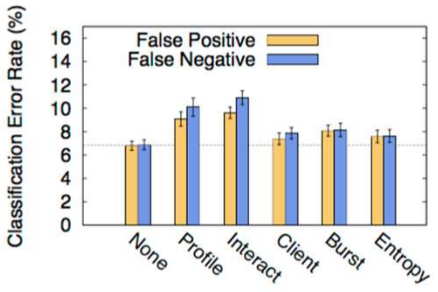
(b) All Workers

Impact of Poisoning Attacks (cont.)

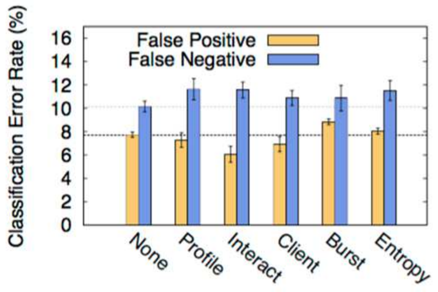
- Instead of adding data to the training set, data in the training set is altered.



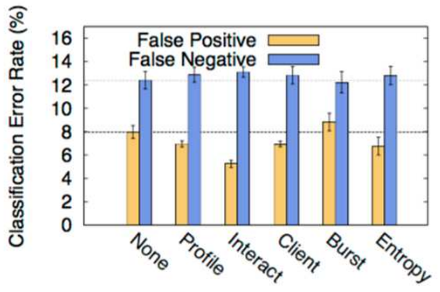
(a) Random Forests



(b) J48

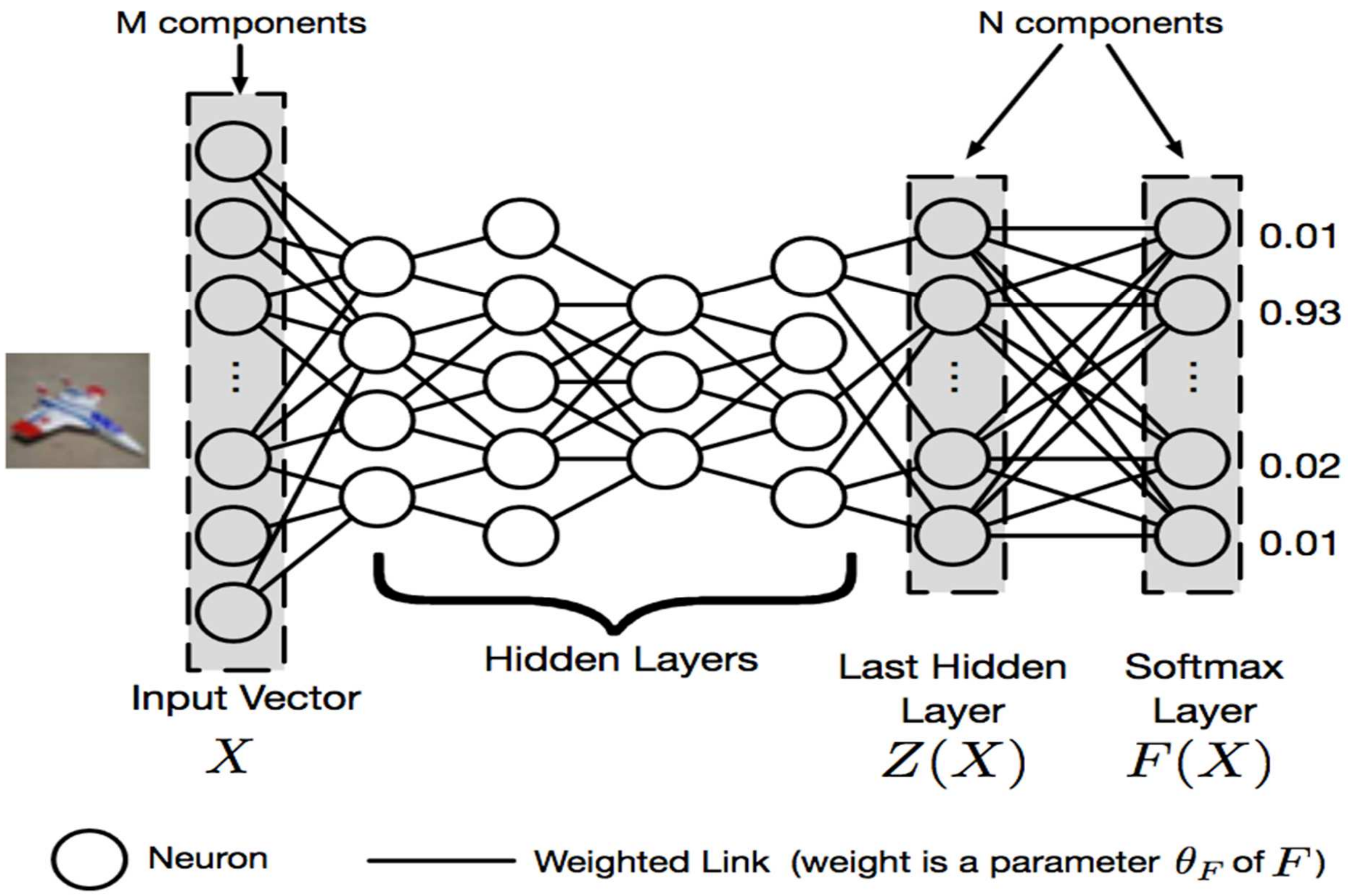


(c) SVMr



(d) SVMp

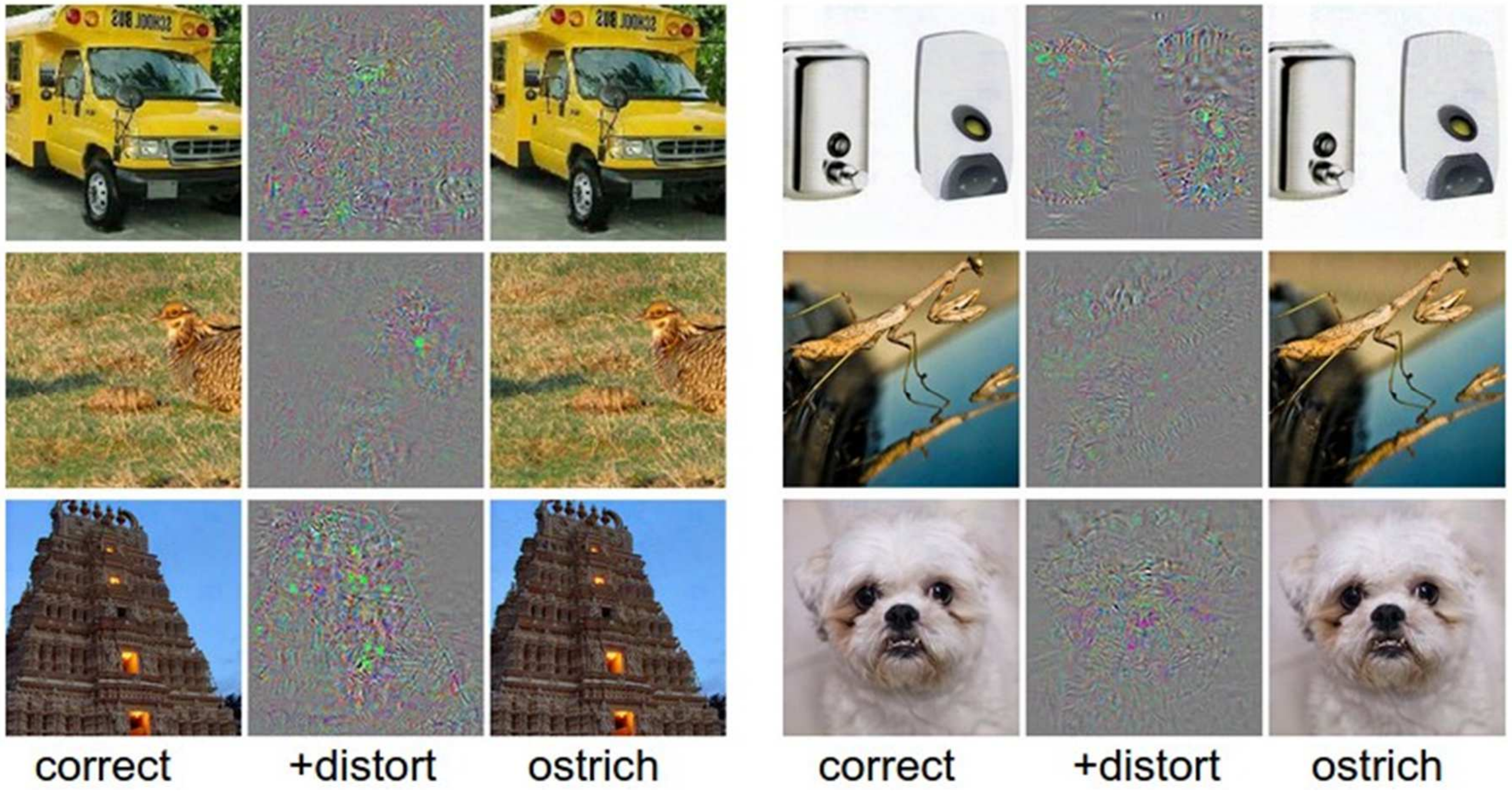
DNN: Attacks and Defenses [16]



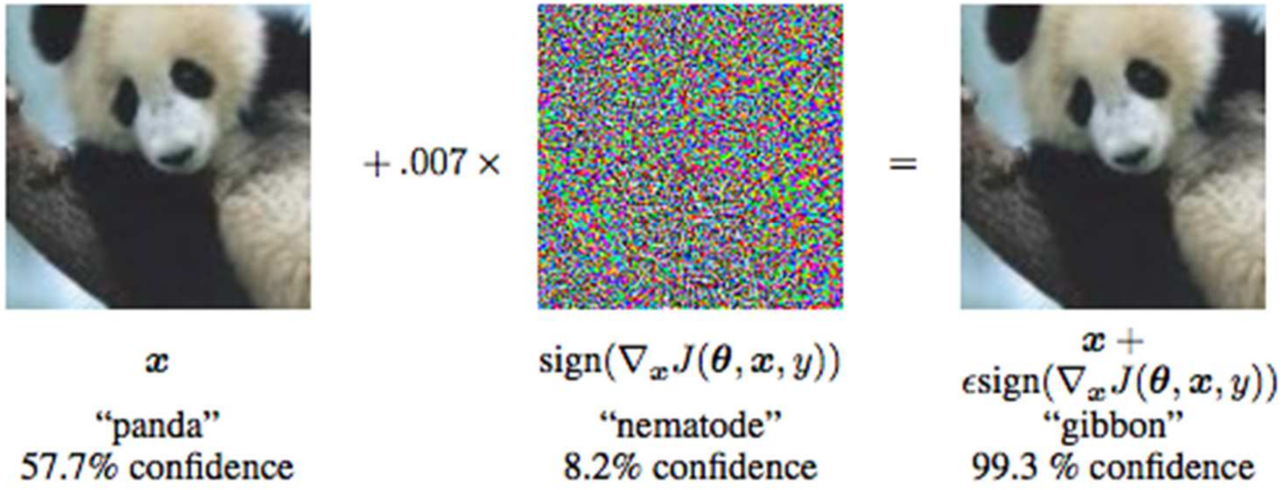
Attacks against Deep Neural Networks

- Recent work in the machine learning and security communities have shown that adversaries can force DNNs to produce adversary-selected outputs using carefully crafted input.

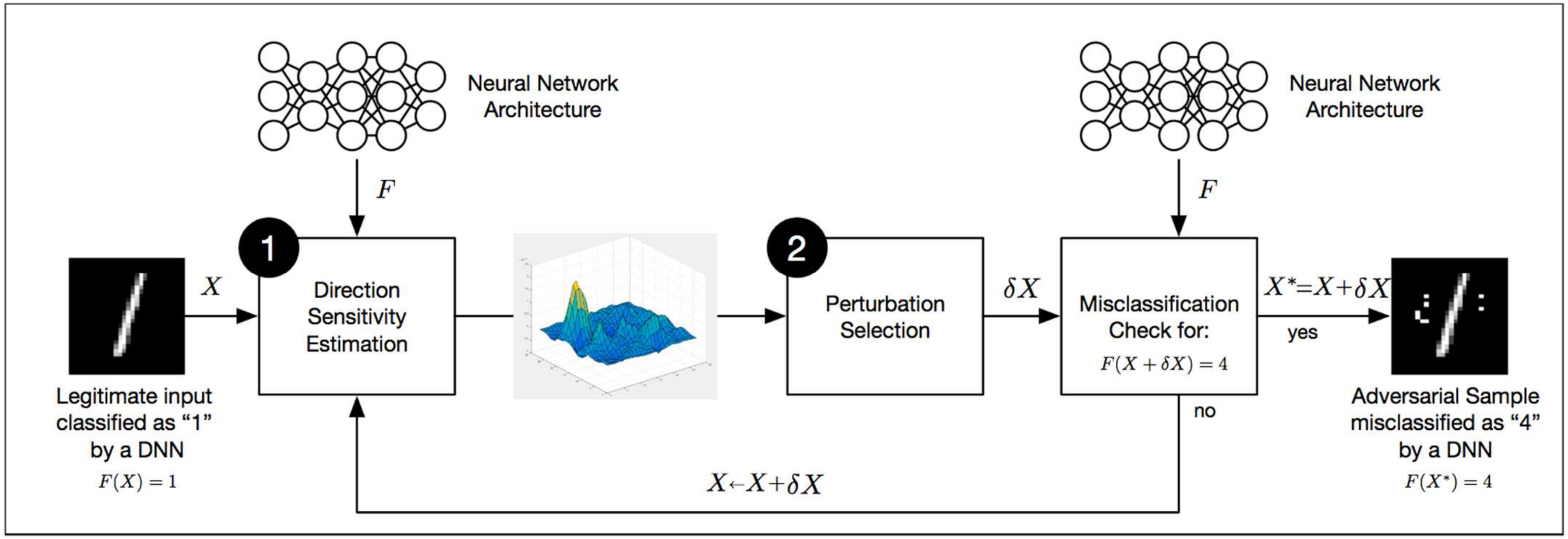
Szagedy et al. [18]



Goodfellow et al. Example [17]


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Adversarial Sample Crafting


Neural Network Architecture
Neural Network Architecture
F
F
1
Direction Sensitivity Estimation
2
Perturbation Selection
Misclassification Check for:
$F(X+\delta X)=4$
X
δX
$X^*=X+\delta X$
yes
no
$X \leftarrow X+\delta X$
Legitimate input classified as "1" by a DNN
$F(X)=1$
Adversarial Sample misclassified as "4" by a DNN
$F(X^*)=4$

Adversarial Crafting

- Crafting consists of two steps: **direction sensitivity estimation** and **perturbation selection**.
- Step 1 evaluates the sensitivity of model *F* at the input point corresponding to sample *X*.
- Step 2 uses this knowledge to select a perturbation affecting sample X's classification.
  - If the resulting sample *X* +*δX* is misclassified by model *F* in the adversarial target class, an adversarial sample *X** has been found.

Direction Sensitivity Estimation

- The goal is to find the dimensions of X that will produce the expected adversarial behavior with the smallest perturbation.
- Goodfellow et al. propose the fast sign gradient method
  - Computes the gradient of the cost function with respect to the input of the neural network
- Papernot et al. propose the forward derivative, which is the Jacobian of F
  - Directly compute the gradients of the output components with respect to each input component.

Perturbation Selection

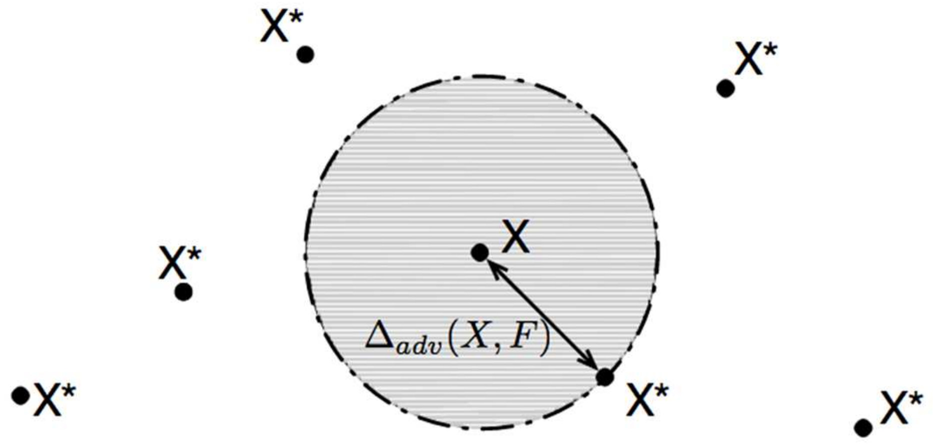
- Goodfellow et al. choose to perturb all input dimensions by a small quantity in the direction of the sign of the gradient they computed.
- Papernot et al. follow a more complex process involving saliency maps to only select a limited number of input dimensions to perturb.
  - Saliency maps assign values to combinations of input dimensions indicating whether they will contribute to the adversarial goal or not if perturbed.

Defending DNNs Using Distillation

- The robustness of a trained DNN model F is:

$$\rho_{adv}(F) = E_{\mu}[\Delta_{adv}(X, F)]$$

$$\Delta_{adv}(X, F) = \arg\min_{\delta X}\{\|\delta X\| : F(X + \delta X) \neq F(X)\}$$



Defense Requirements

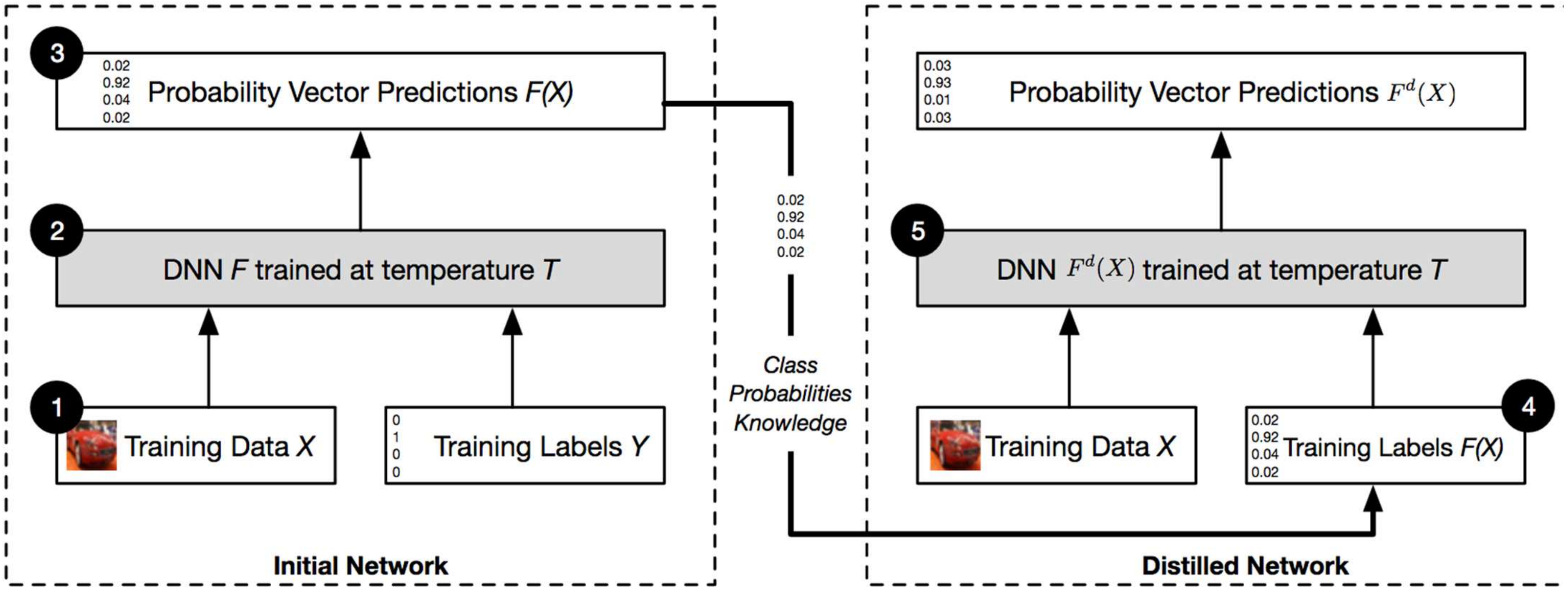
- Low impact on the architecture
- Maintain accuracy
- Maintain speed of network
- Defenses should work for adversarial samples relatively close to points in the training dataset

Distillation

- Distillation is a training procedure initially designed to train a DNN using knowledge transferred from a different DNN.
  - extract class probability vectors produced by a first DNN or an ensemble of DNNs to train a second DNN of reduced dimensionality without loss of accuracy.
  - label inputs in the training dataset of the second DNN using their classification predictions according to the first DNN.

- Use defensive distillation to smooth the model learned by a DNN architecture during training by helping the model generalize better to samples outside of its training dataset.

Distillation as a Defense



$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Evaluation

- **Question**: Does defensive distillation improve resilience against adversarial samples while retaining classification accuracy?
- **Result**:
  - Distillation reduces the success rate of adversarial crafting from 95.89% to 0.45% on their first DNN and dataset, and from 87.89% to 5.11% on the second DNN and dataset.
  - Distillation has negligible or non existent degradation in model classification accuracy in these settings

DNN Architectures

- 9 layer convolutional neural networks, T = 20
- The resulting Distilled DNN achieves a 99.05% accuracy on the MNIST data set, and 81.39% on the CIFAR10 data set.
- Applying attacks proposed by Papernot et al.
  - Construct an adversarial sample X* from a benign sample X by adding a perturbation vector δX solving the following optimization problem

$$\arg\min_{\delta_{\mathbf{X}}} \|\delta_{\mathbf{X}}\| \text{ s.t. } \mathbf{F}(\mathbf{X} + \delta_{\mathbf{X}}) = \mathbf{Y}^*$$

Evaluation (cont.)

- **Question**: Does defensive distillation reduce DNN sensitivity to inputs?
- **Result**:
  - Defensive distillation reduces DNN sensitivity to input perturbations, where experiments show that performing distillation at high temperatures can lead to decreases in the amplitude of adversarial gradients by factors up to 10^{30}.

Evaluation (cont.)

- **Question**: Does defensive distillation lead to more robust DNNs?
- **Result**:
  - Defensive distillation impacts the average minimum percentage of input features to be perturbed to achieve adversarial targets (i.e., robustness).
  - In their DNNs, distillation increases robustness by 790% for the first DNN and 556% for the second DNN.

Datasets

- MNIST
  - 70,000 black and white images of handwritten digits in 10 classes
  - training set of 60,000 samples, test on 10,000 samples
- CIFAR10
  - a collection of 60,000 color images in 10 classes
  - 50,000 for training, and 10,000 for testing

Still this can be attacked

- New work on showing that distillation can be easily attacked (See Oakland ‘17)

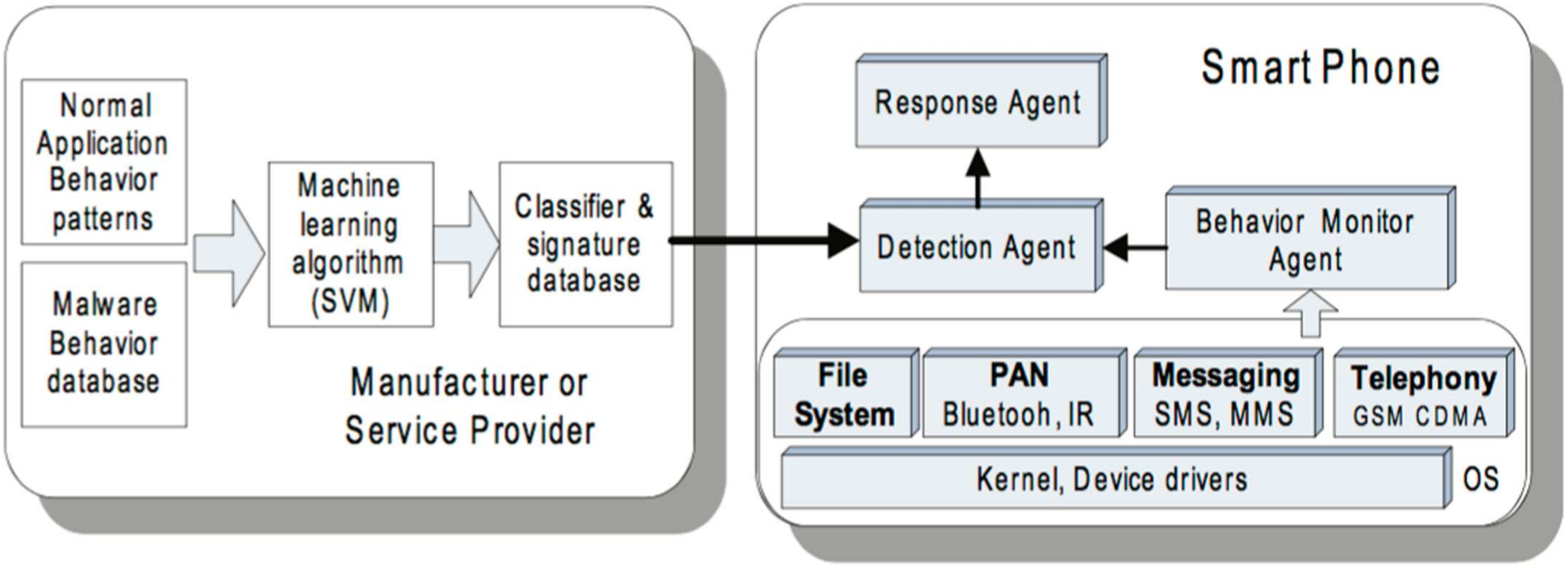
Behavioral Detection of Malware on Mobile Handsets [12]

- Mobile Malware Detection
  - signature-based solutions are not efficient for resource-constrained mobile devices
  - behavioral detection solution to detecting mobile worms, viruses and Trojans
    - Monitor the run-time behavior of an application (e.g., file accesses, API calls)
    - More resilient to polymorphic worms and code obfuscation
    - Database of behavior profiles is much smaller than that needed for storing payload signatures

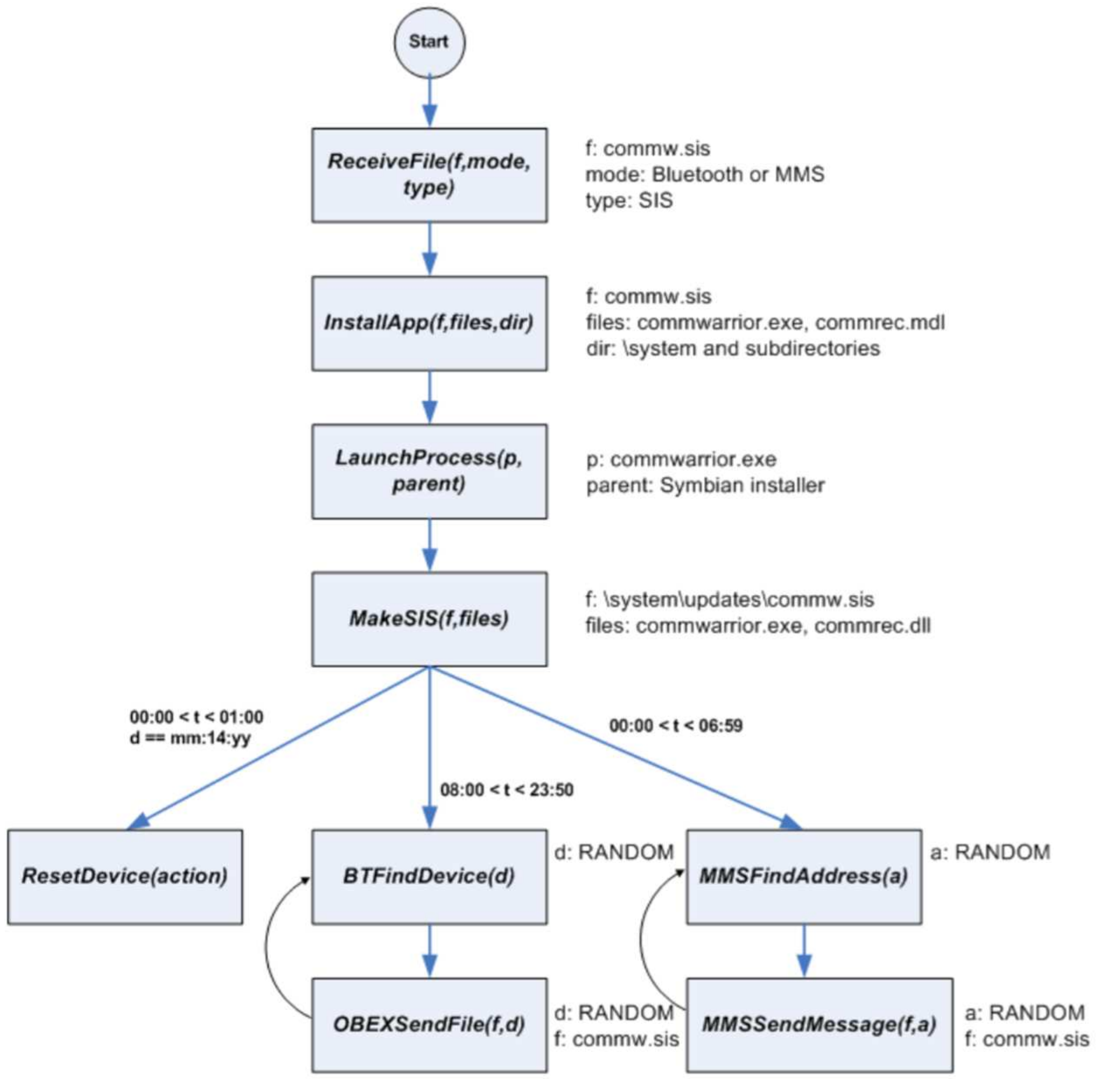
Challenges

- Behavior specification
  - temporal logic of causal knowledge
- Online reconstruction of suspicious behavior
  - Train a SVM to differentiate partial signatures for malicious behavior from those of normal applications.
  - The resulting SVM model and the malicious signature database are preloaded onto the handset.

System Overview

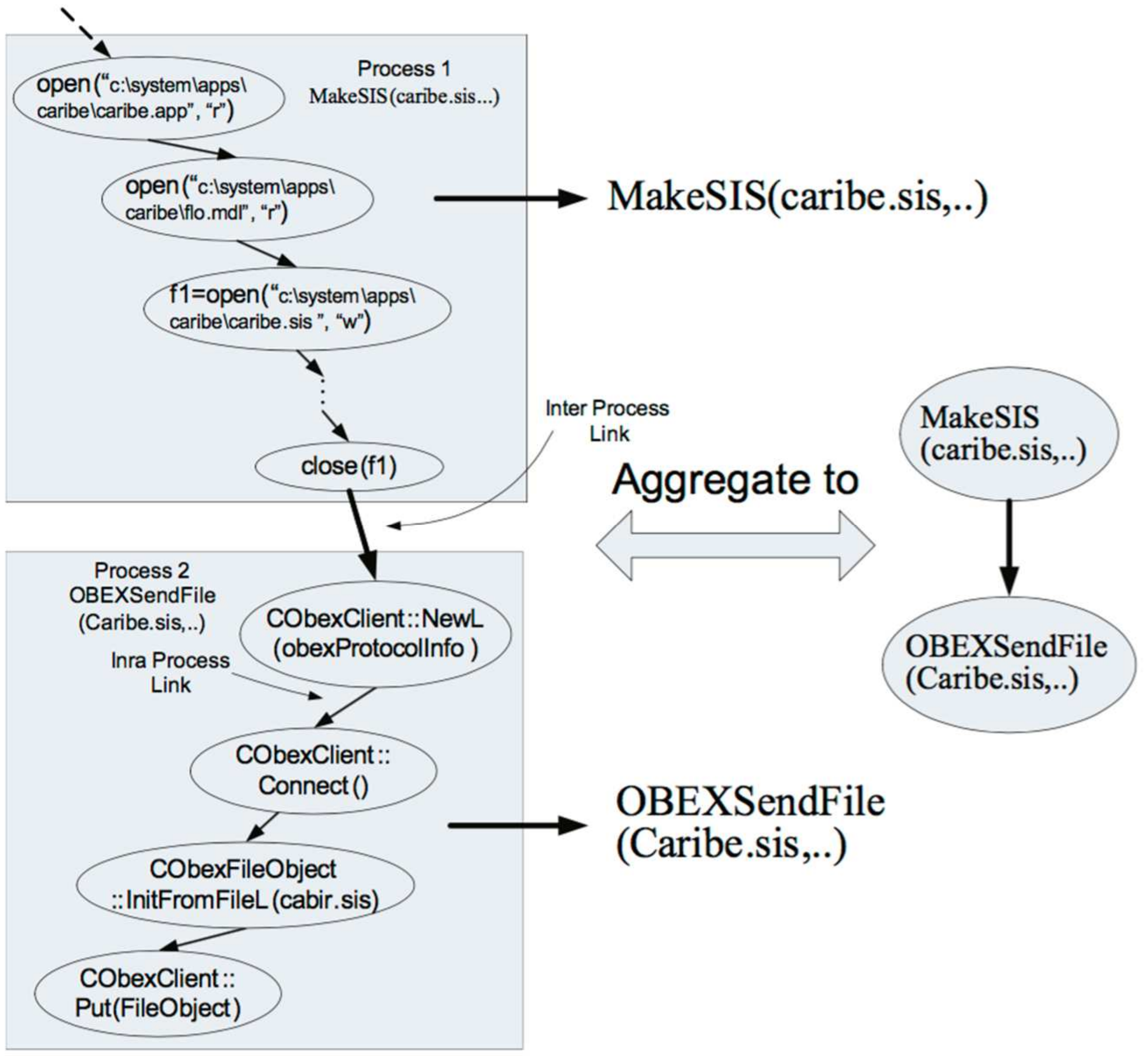

Normal Application Behavior patterns
Malware Behavior database
Machine learning algorithm (SVM)
Classifier & signature database
Manufacturer or Service Provider
Smart Phone
Response Agent
Detection Agent
Behavior Monitor Agent
File System
PAN Bluetooh, IR
Messaging SMS, MMS
Telephony GSM CDMA
Kernel, Device drivers
OS

Behavior Signature of Commwarrior Worm



Two-stage Runtime Behavior Signature Construction

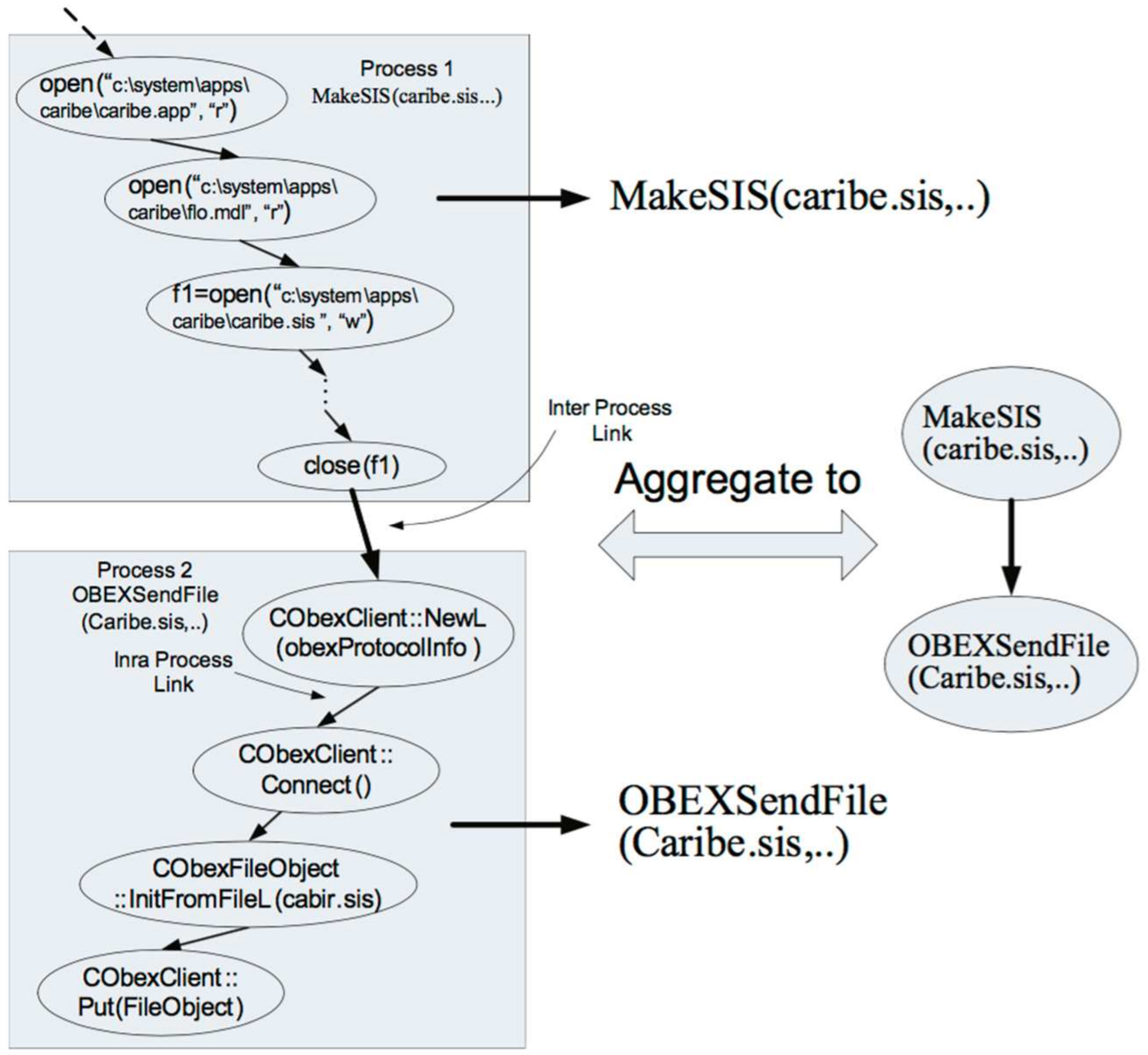
- Stage 1: generation of dependency graph



Process 1
MakeSIS(caribe.sis...)
open("c:\system\apps\caribe\caribe.app", "r")
open("c:\system\apps\caribe\flo.mdl", "r")
f1=open("c:\system\apps\caribe\caribe.sis", "w")
close(f1)
MakeSIS(caribe.sis,..)
Inter Process Link
Aggregate to
MakeSIS (caribe.sis,..)
OBEXSendFile (Caribe.sis,..)
Process 2
OBEXSendFile (Caribe.sis,..)
Inra Process Link
CObexClient::NewL (obexProtocolInfo)
CObexClient::Connect()
CObexFileObject ::InitFromFileL(cabir.sis)
CObexClient::Put(FileObject)
OBEXSendFile (Caribe.sis,..)

Two-stage Runtime Behavior Signature Construction cont.

- Stage 2: graph pruning and aggregation



Process 1
MakeSIS(caribe.sis...)
open("c:\system\apps\caribe\caribe.app", "r")
open("c:\system\apps\caribe\flo.mdl", "r")
f1=open("c:\system\apps\caribe\caribe.sis", "w")
close(f1)
MakeSIS(caribe.sis,..)
Inter Process Link
Aggregate to
MakeSIS (caribe.sis,..)
OBEXSendFile (Caribe.sis,..)
Process 2
OBEXSendFile (Caribe.sis,..)
Inra Process Link
CObexClient::NewL (obexProtocolInfo)
CObexClient:: Connect()
CObexFileObject ::InitFromFileL(cabir.sis)
CObexClient:: Put(FileObject)
OBEXSendFile (Caribe.sis,..)

Malware Detection on Mobile Devices using Distributed Machine Learning [13]

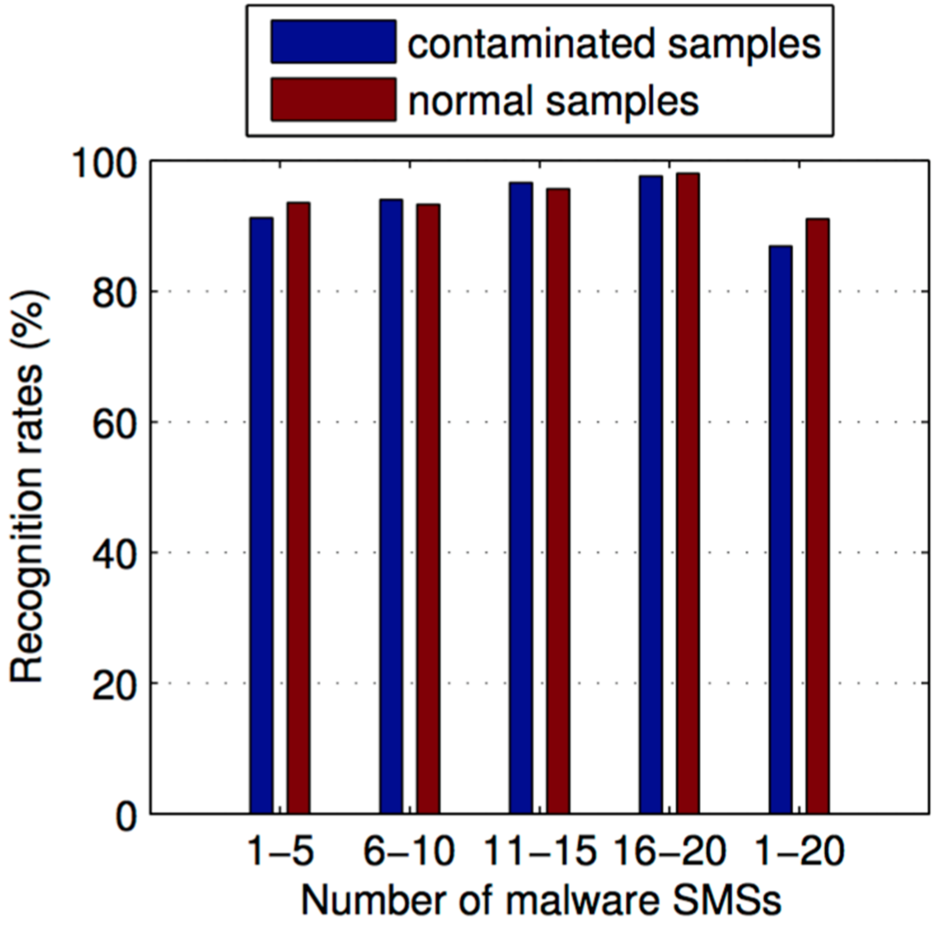
- The distributed SVM for detecting mobile malware:
  - lightweight in terms of bandwidth usage
  - preserve the privacy of the participating users
  - automatically generate a general behavioral signature of malware

Distributed SVM Learning

- Divide the quadratic SVM binary classification problem into multiple sub-problems by relaxing it using a penalty function.
- Next, distributed continuous- and discrete-time gradient algorithms are applied to solve the relaxed problem iteratively.

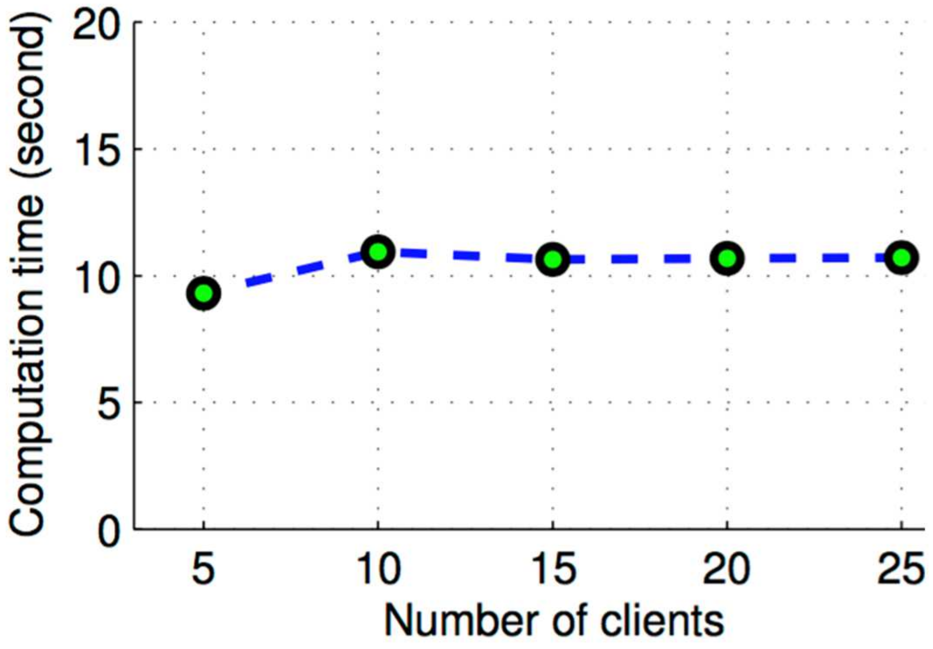
Empirical Results

- MIT Reality Mining user data
  - 897922 communication logs collected from 97 users
  - Infect half of data set with malware symptoms

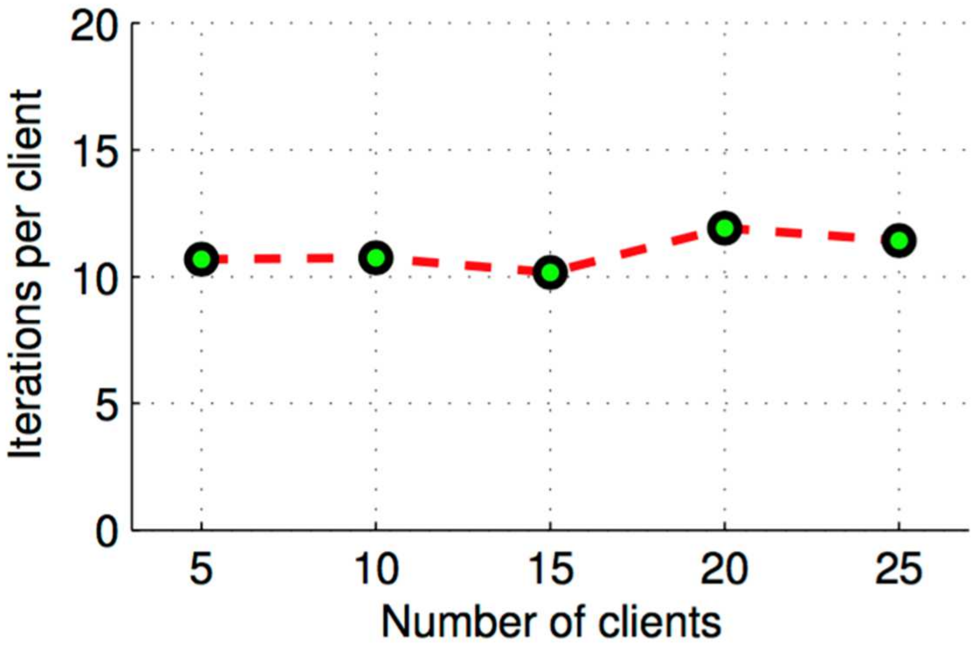


Computational Requirement

avg. computation time/client



avg. number of updates/client



Outside the closed world: On using machine learning for network intrusion detection [9]

- Network Intrusion Detection:
  - Misuse detection – precise descriptions of known malicious behavior.
  - Anomaly detection – flag deviations from normal activities.

Machine Learning in Intrusion Detection

- Well-known problems:
  - High false positive rate
  - Lack of attack-free data for training
    - Theoretical results indicate this is should not be an issue for big data
  - Attackers can foil the system to evade detection
    - More resistant techniques now available as we discussed before
  - Difficulties with Evaluation

Machine Learning in Intrusion Detection

- Challenges of ML for NID
  - Outlier Detection
  - High Cost of Errors
  - Semantic Gap (interpretation of results)
  - Diversity of Network Traffic

Recommendations for Using Machine Learning

Understand what the system is doing!

- Recommendations:
  - – Understanding the Threat Model
    - What kind of environment does the system target?
    - What do missed attacks cost?
    - What skills and resources will attackers have?
    - What concern does evasion pose?
  - – Keep the Scope Narrow
  - – Reducing the Costs
  - – Evaluation
    - Working with data
    - Understanding results

Summary of [9]

- Domain-specific challenge:
  - An extensive amount of research on machine learning-based anomaly detection, versus the lack of operational deployments of such systems.
    - Now start-ups are trying to change that.

- Follow a set of guidelines for applying ML to network intrusion detection
  - obtain *insight* into the operation of an anomaly detection system *from an operational point of view*.
  - *Semantic* understanding of the gain on ROC curves is crucial.

CONCLUSIONS

Lessons Learned

- Data Mining for Cyber Security requires better understanding of attacker.
  - Game theory provides natural tools for such modeling
- Dynamic adaptation, cost of adaptation, utility of the attacker and defender needs to be considered.
- Other issues not discussed but important:
  - Provenance of data
  - Class Imbalance
  - Adversarial Active Learning

Main Take Away

- “If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle.” — Sun Tzu, The Art of War
- Choose the features carefully.
  - Understand attacker capabilities and potential adaptation
- Use robust machine learning techniques
- Scale to large data

Choosing right features for classification

- As game theoretical models indicate, good features are:
  - Hard for attacker to manipulate; and
  - Indicative of the attack
- Example: Malware detection
  - Focus on more behavioral features than syntactic features extracted from binary ?

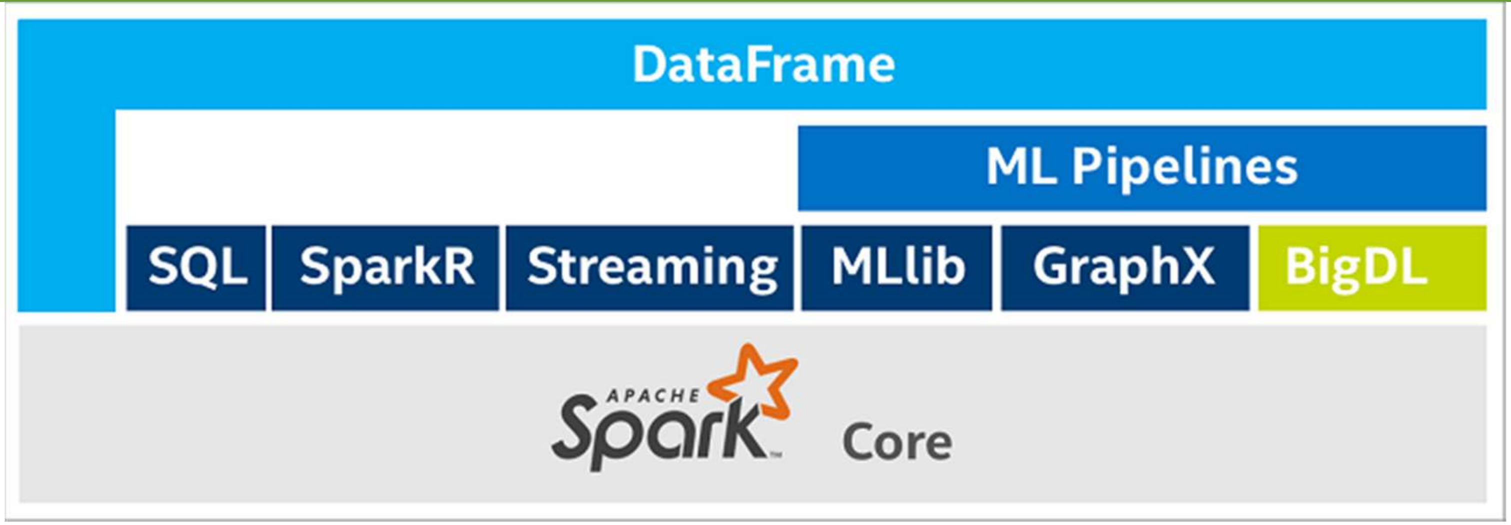
Choosing the right machine learning tool

- Trying large set of tools are critical
  - Random forest
  - SVM
  - Neural networks
  - Deep belief networks etc.
  - Ad-Svm
  - Others ??

Scaling to large data

- Efficient distributed processing systems
  - Hadoop/MapReduce
  - Spark
  - Storm
  - Others

Apache Spark



- high-level APIs in Java, Scala and Python, and an optimized engine that supports general execution graphs
- supports a rich set of higher-level tools provides
  - Spark SQL for SQL and structured data processing
  - MLlib for machine learning,
    - Many algorithms..
  - GraphX for graph processing, and
  - Spark Streaming

Apache Spark (Recent addition DL)

BigDL is a distributed deep learning library for Spark

- Rich deep learning support. Modeled after Torch BigDL provides comprehensive support for deep learning, including numeric computing
- Efficient scale out. BigDL can efficiently scale out to perform data analytics at “big data scale” by using Spark.
- Extremely high performance. To achieve high performance, BigDL uses Intel® Math Kernel Library (Intel® MKL) and multithreaded programming in each Spark task.
- Our experience. For low dimensional data, standard techniques such as random forests, SVM works well enough.

Acknowledgement

Our work on this topic has been supported by Army Research Office Grant 58345-CS.

More papers on the topic could be found on the data mining for cyber security course web page:

http://www.utdallas.edu/~muratk/courses/dbmsec-15s.html

References

1. M. Kearns and M. Li. Learning in the presence of malicious errors. SIAM Journal on Computing, 22:807-837, 1993.
2. N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, Adversarial classification, KDD '04.
3. M. Kantarcioglu, B. Xi, and C. Clifton, Classifier evaluation and attribute selection against active adversaries, Data Min. Knowl. Discov., vol. 22, pp. 291-335, January 2011.
4. M. Bruckner and T. Scheffer. Stackelberg games for adversarial prediction problems, SIGKDD, 2011.
5. Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and B. Xi, Adversarial support vector machine learning, SIGKDD '12.
6. Y. Zhou and M. Kantarcioglu, Modeling Adversarial Learning as Nested Stackelberg Games, PAKDD ’16.
7. Dekel, O., O. Shamir, Learning to classify with missing and corrupted features, ICML 2008.

References

8. D. Lowd and C. Meek., Adversarial learning, page 641-647, KDD 2005.
9. Sommer et al., Outside the closed world: On using machine learning for network intrusion detection, IEEE S&P 2010.
10. C. Smutz and A. Stavrou, Malicious PDF detection using metadata and structural features, in Annual Computer Security Applications Conference (ACSAC), 2012, pp. 239-248.
11. Nedim Srndic and Pavel Laskov, Practical Evasion of a Learning-Based Classifier: A Case Study, Proceedings of the 2014 IEEE Symposium on Security and Privacy, Pages 197-211.
12. Abhijit Bose, Xin Hu, Kang G. Shin, and Taejoon Park, Behavioral detection of malware on mobile handsets, MobiSys '08. pp. 225-238.
13. A.S. Shamili, C. Bauckhage, and T. Alpcan, Malware Detection on Mobile Devices Using Distributed Machine Learning, ICPR ’10.

References

14. Wang et al. "Man vs. Machine: Practical Adversarial Detection of Malicious Crowdsourcing Workers", Usenix Security 2014
15. Weilin Xu, Yanjun Qi, and David Evans, Network and Distributed System Security Symposium (NDSS) 2016
16. Distillation as a defense to adversarial perturbations against deep neural networks, Papernot et al., 2016
17. Goodfellow et al. “Explaining and harnessing adversarial examples”, ICLR 2015
18. Szegedy et al. “Intriguing properties of neural networks”, ICLR 2013