TRUSTWORTHY MACHINE LEARNING

POISONING

KORBINIAN KOCH, RAHAVI SELVARAJAN, ANGELA WANG

SEPTEMBER 21, 2021

UNIVERSITY OF TORONTO

ATTACKING SPAM FILTERS

In practice, we regularly see some of the most advanced spammer groups trying to throw the Gmail filter off-track by reporting massive amounts of spam emails as not spam. As shown in the figure, between the end of Nov 2017 and early 2018, there were at least four malicious large-scale attempts to skew our classifier.

Elie Bursztein, Google

https://elie.net/blog/ai/attacks-against-machine-learning-an-overview/

EXAMPLE POISONED SPAM DATAPOINT

Hello Friend!

Your professor doesn't want you to know this!

*** FIND GLOBAL OPTIMA IN SECONDS ***

>>> READ THE FREE PAPER HERE <<<

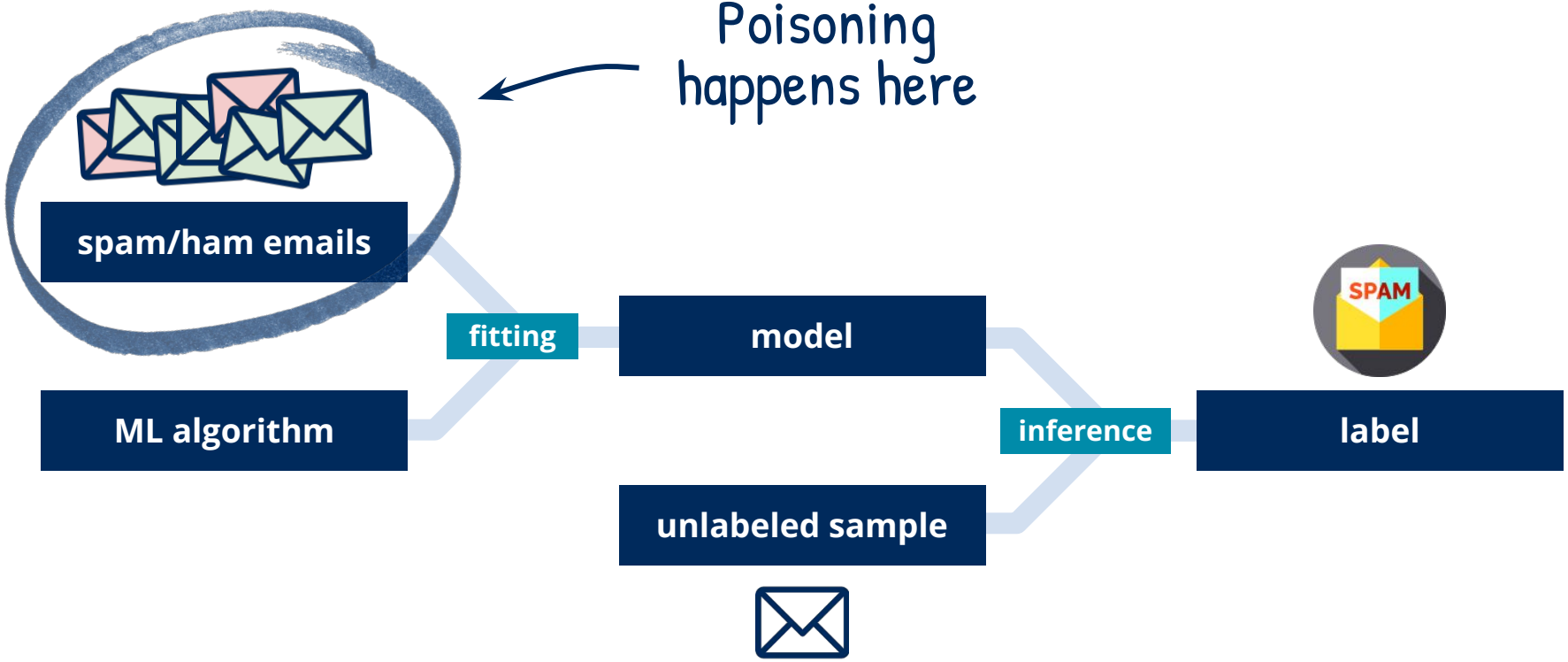
Supply is limited!!1!

Use referral code: TrustworthyML

www.global-optima-fast.xx

Manual label: **not spam**

THE SPAM PIPELINE



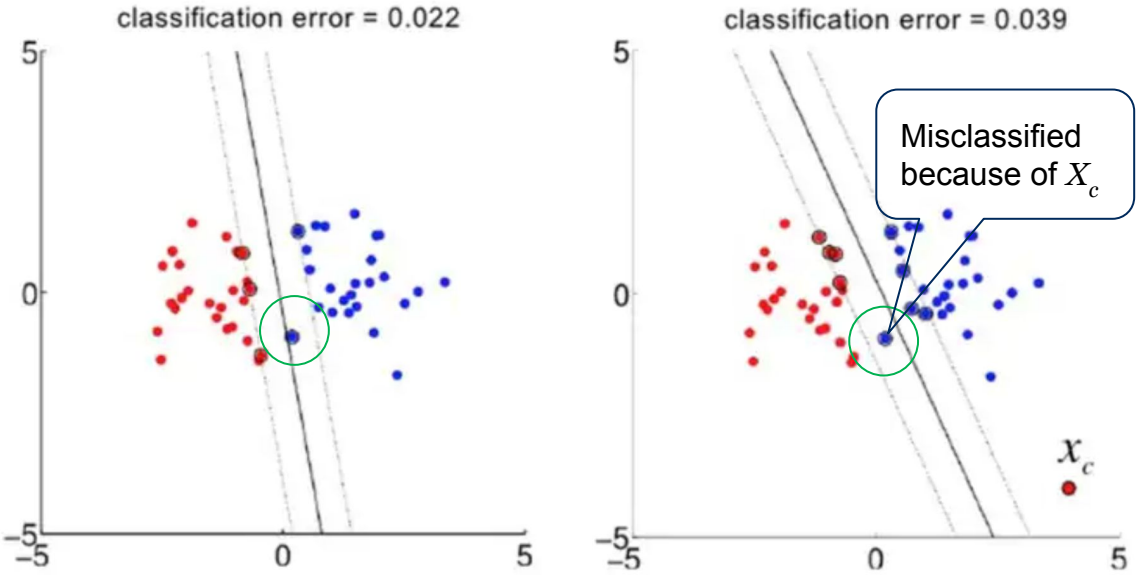
THREAT MODEL

- Model owner is the victim
- Attacker tries to modify the dataset before model owner trains the model in order to influence prediction output of the model
- Most frequent goals

  → affect the model's accuracy

  → create model backdoor

EXAMPLE: SHIFT OF DECISION BOUNDARY



https://www.slideshare.net/DavidDao1/causative-adverserial-learning

TYPES OF POISONING

1. **Targeted vs. untargeted:**
   - want a specific email to pass the spam detection: targeted
   - generally reduce model's accuracy: untargeted

2. **Availability vs. integrity**
   - data injection by adding confusing emails: availability
   - backdoor for specific keyword in email: integrity

and more ...

LABEL MODIFICATION

Label: ~~2~~ **9**

Label: 5

Label: ~~9~~ **8**

Label: 4

DATA INJECTION

Label: 2

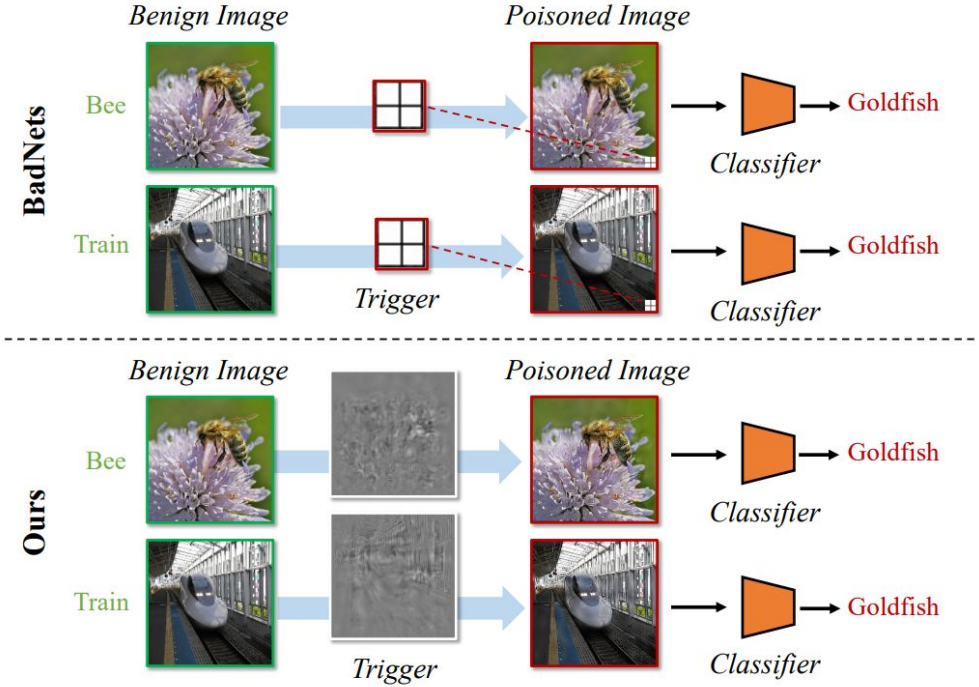
Label: 5

Label: 9

Label: 4

Label: 5

BACKDOORING



Li, Yuezun & Li, Yiming & Wu, Baoyuan & Li, Longkang & He, Ran & Lyu, Siwei. (2020). Backdoor Attack with Sample-Specific Triggers.

CHOOSE YOUR POISON

Backdooring

embeds secret trigger for target prediction

✘ no desired effect on normal performance

Skewing

shifts decision boundary in desired direction

✔ desired effect on normal performance

CONCLUSION

Attackers actively attempt to shift the learned boundary between abusive and legitimate use in their favor.

Poisoning training data by manipulating the real world.

OVERVIEW


Paper #1
Poisoning
PCA
Paper #3
Poisoning
SVD
Paper #2
Poisoning
Regression
Models
2009
2019
2021

OUTLINE

Intro

~10 min

Paper #1

Presentation and Discussion

~24 min

Paper #2

Presentation and Discussion

~24 min

Break

Paper #3

Presentation and Discussion

~24 min

Summary Discussion

~7 min

1

In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, November 2009, pp. 1-14

ANTIDOTE: Understanding and Defending against Poisoning of Anomaly Detectors

Benjamin I. P. Rubinstein[1] Blaine Nelson[1] Ling Huang[2] Anthony D. Joseph[1,2]
Shing-hon Lau[1] Satish Rao[1] Nina Taft[2] J. D. Tygar[1]

[1]Computer Science Division, University of California, Berkeley [2]Intel Labs Berkeley

ABSTRACT

Statistical machine learning techniques have recently garnered increased popularity as a means to improve network design and security. For intrusion detection, such methods build a model for normal behavior from training data and detect attacks as deviations from that model. This process invites adversaries to manipulate the training data so that the learned model fails to detect subsequent attacks.

We evaluate poisoning techniques and develop a defense, in the context of a particular anomaly detector—namely the PCA-subspace method for detecting anomalies in backbone networks. For three poisoning schemes, we show how attackers can substantially increase their chance of successfully evading detection by only adding moderate amounts of poisoned data. Moreover such poisoning throws off the balance between false positives and false negatives thereby dramatically reducing the efficacy of the detector.

To combat these poisoning activities, we propose an antidote based on techniques from robust statistics and present a new robust PCA-based detector. Poisoning has little effect on the robust model, whereas it significantly distorts the

Keywords

Network Traffic Analysis, Principal Components Analysis, Adversarial Learning, Robust Statistics

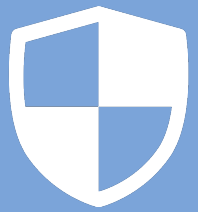
1. INTRODUCTION

Statistical machine learning (SML) techniques are increasingly being used as tools for analyzing and improving network design and performance. They have been applied to a variety of problems such as enterprise network fault diagnosis [1, 5, 14], email spam filtering [24, 27], worm detection [25], and intrusion detection [16, 30, 33], as well as many others. These solutions draw upon a variety of techniques from the SML domain including Singular Value Decomposition, clustering, Bayesian inference, spectral analysis, maximum-margin classification, etc. In many scenarios, these approaches have been demonstrated to perform well. Many of these SML techniques include a learning phase during which a model is trained using collected data. Such techniques have a serious vulnerability, namely they are susceptible to adversaries who purposefully inject mali-

OUTLINE

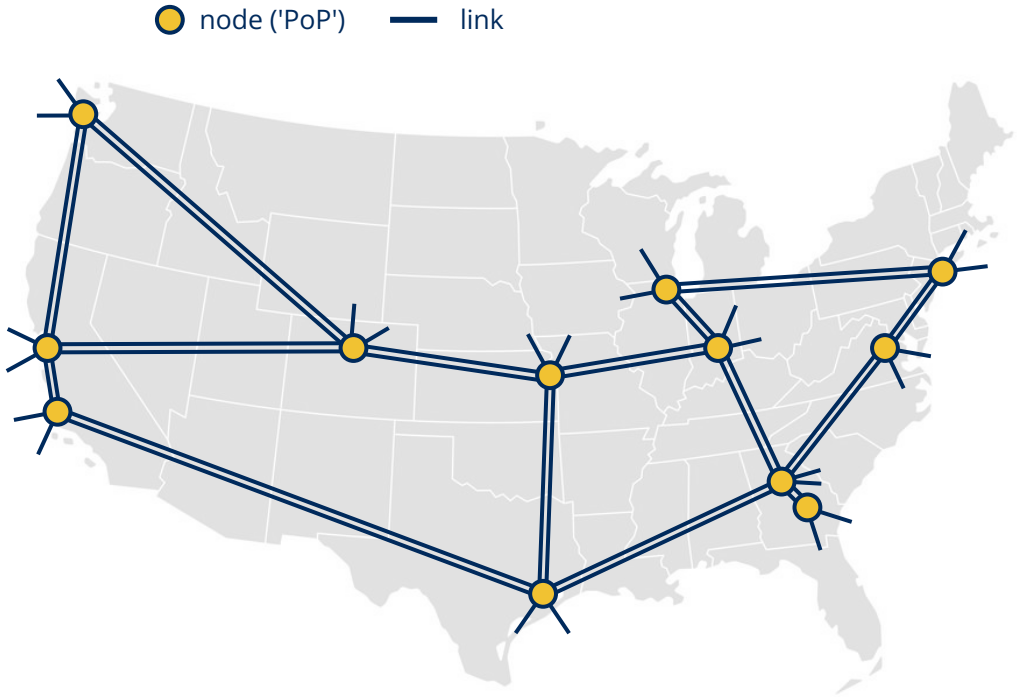
Understanding Anomaly Detectors

Understanding Poisoning of Anomaly Detectors

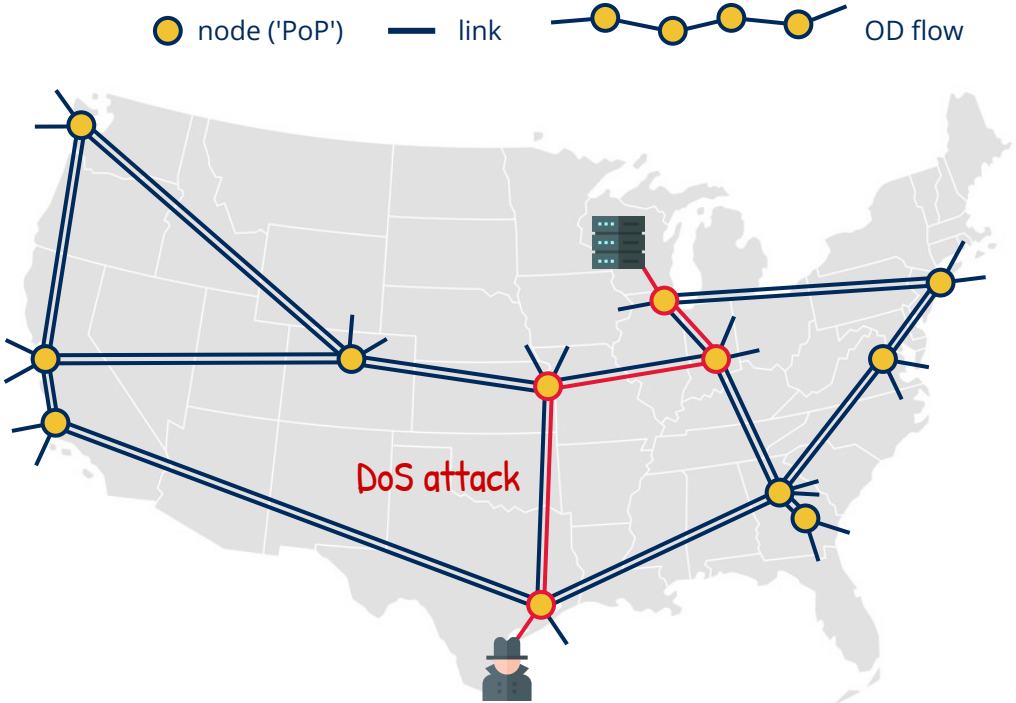


Defending against Poisoning of Anomaly Detectors

ABILENE BACKBONE NETWORK



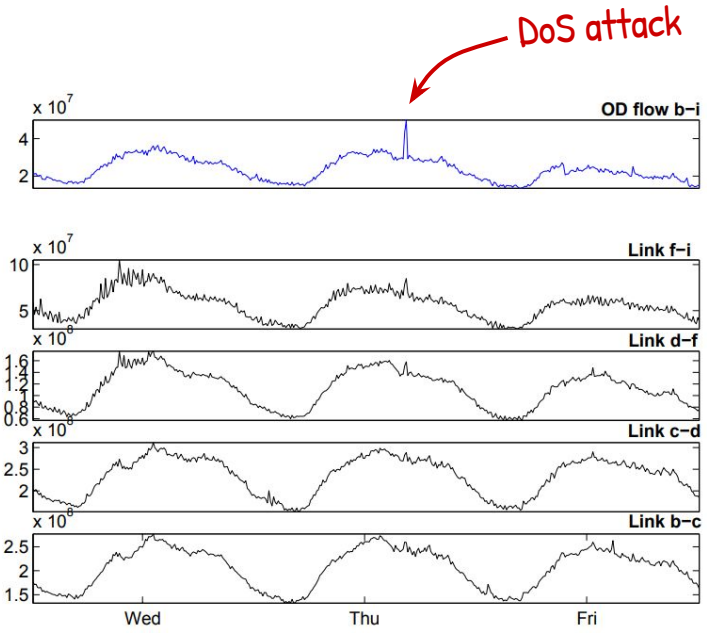
ABILENE BACKBONE NETWORK



BACKBONE NETWORK TRAFFIC

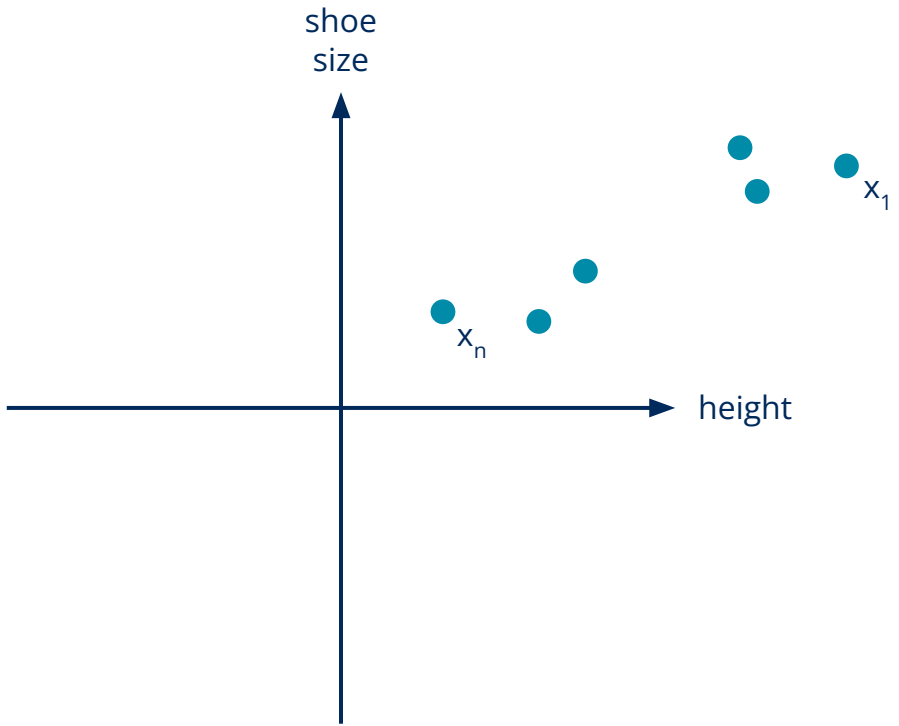
144 OD flows

54 links



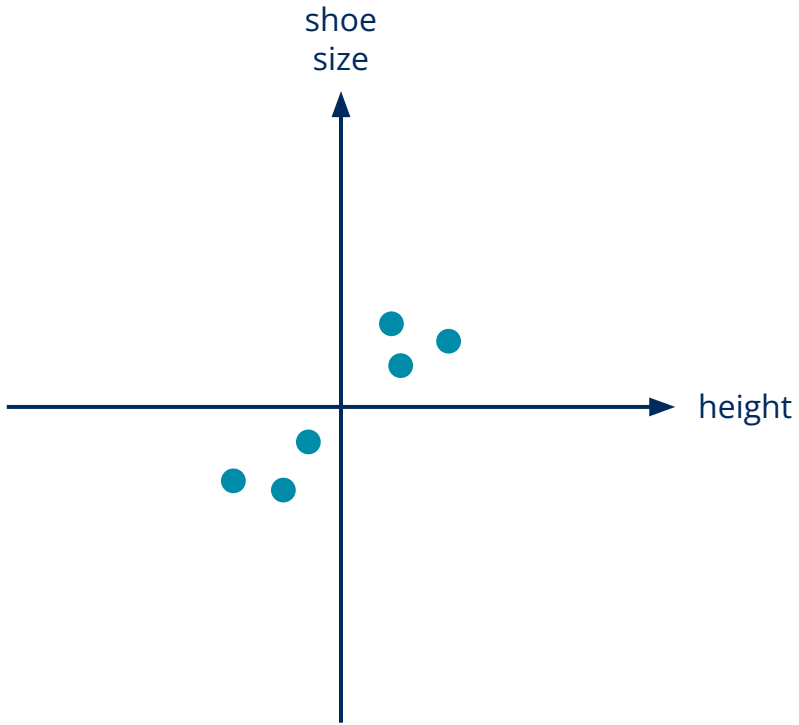
Lakhina, Anukool & Crovella, Mark & Diot, Christophe. (2004). Diagnosing Network-Wide Traffic Anomalies. Computer Communication Review. 34.

RECAP: VANILLA PCA



RECAP: VANILLA PCA

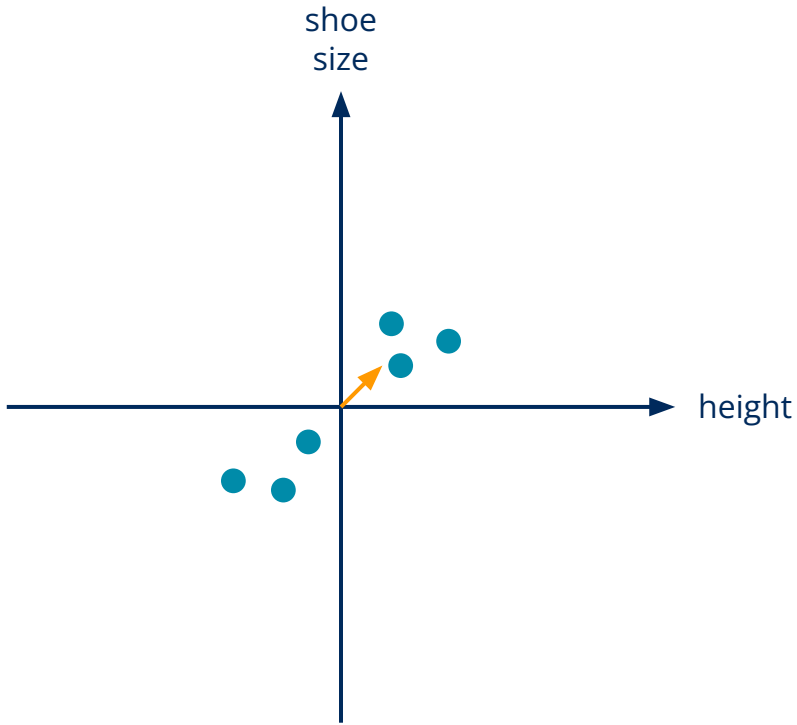
1. Standardize data such that **mean** and **sd** of each dimension are 0 and 1



RECAP: VANILLA PCA

1. Standardize data such that **mean** and **sd** of each dimension are 0 and 1

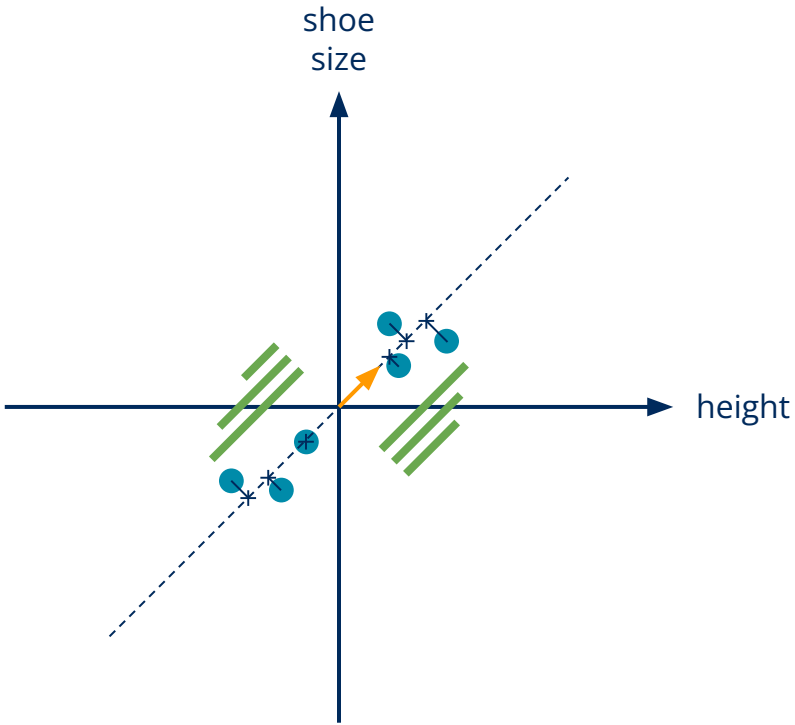
2. Find **unit vector**



RECAP: VANILLA PCA

1. Standardize data such that **mean** and **sd** of each dimension are 0 and 1

2. Find **unit vector** such that it maximizes the **mean squared distance** of the projected points to the origin



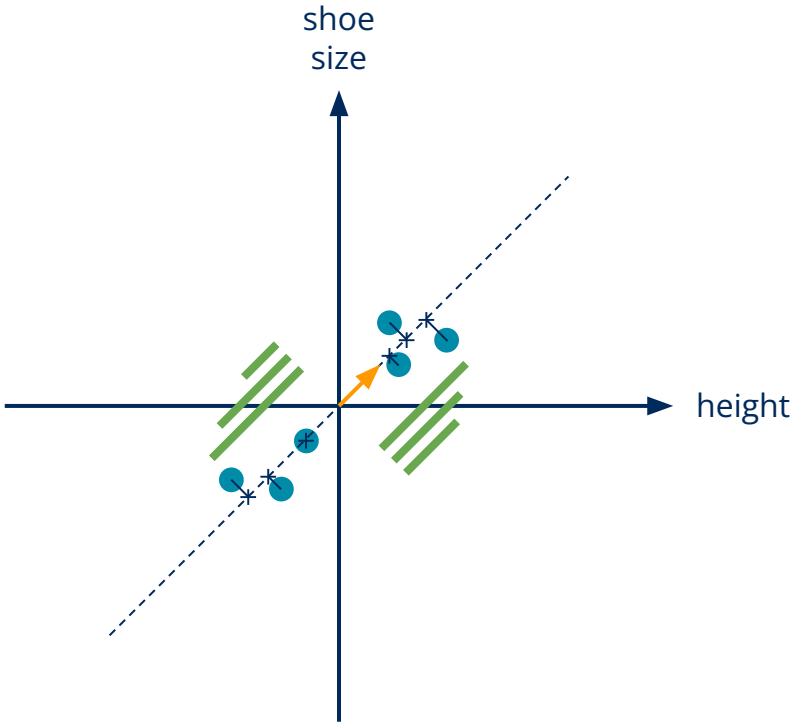
RECAP: VANILLA PCA

1. Standardize data such that **mean** and **sd** of each dimension are 0 and 1

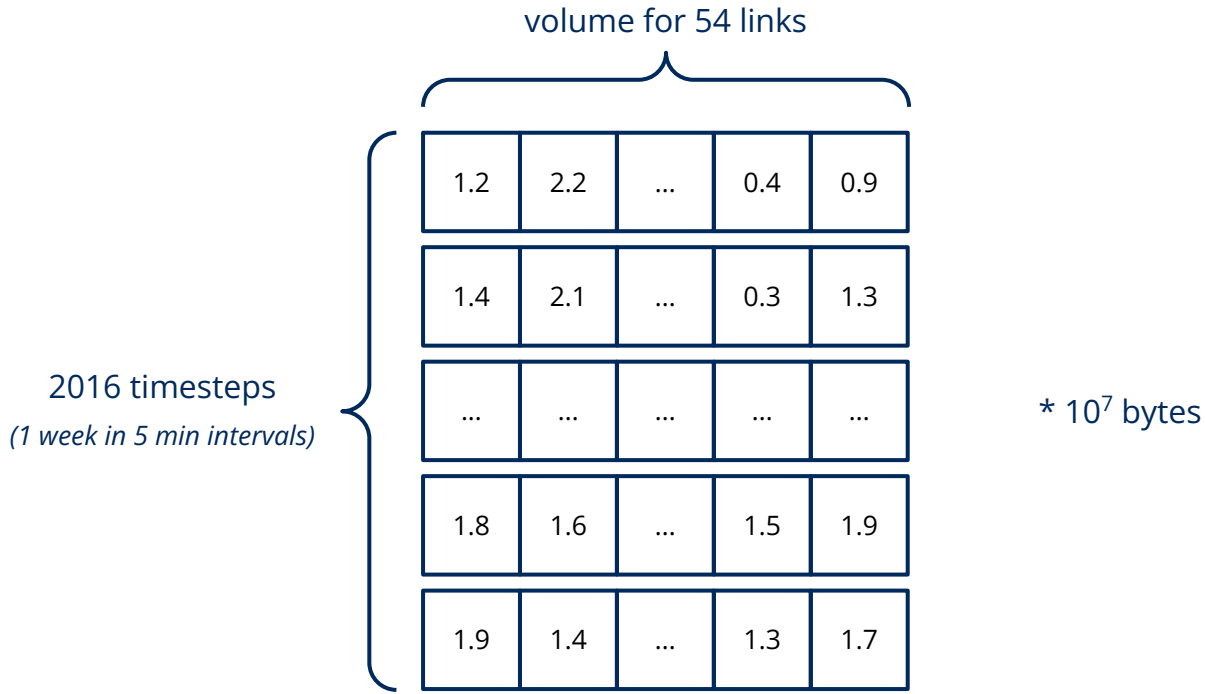
2. Find **unit vector** such that it maximizes the **mean squared distance** of the projected points to the origin

PC1

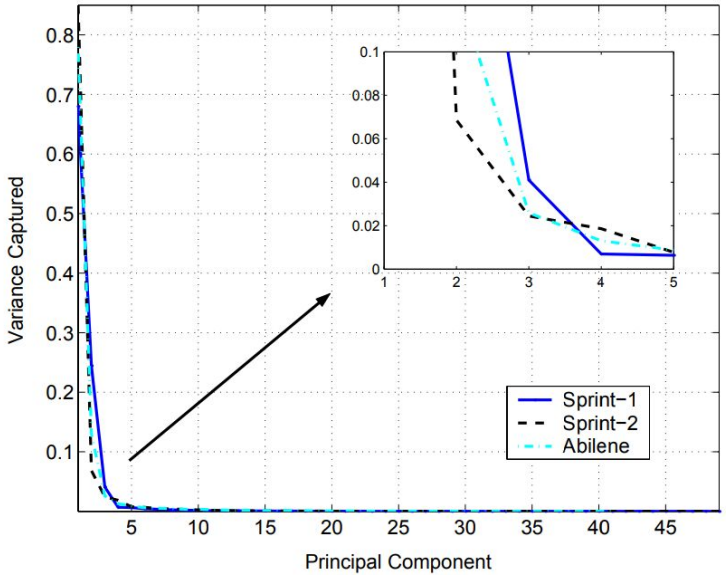
explained variance



NETWORK TRAFFIC TRAINING DATA

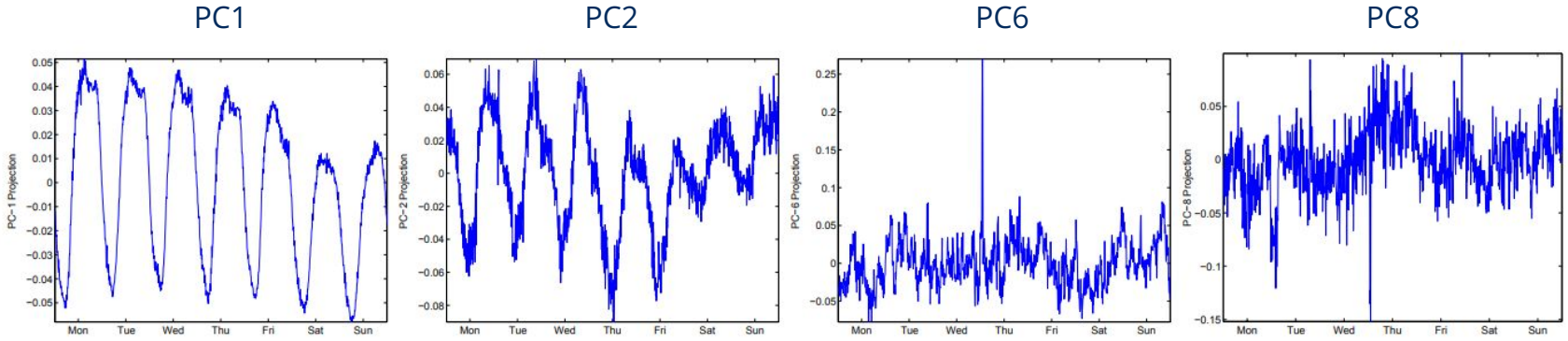


PCA SCREE PLOT



Lakhina, Anukool & Crovella, Mark & Diot, Christophe. (2004). Diagnosing Network-Wide Traffic Anomalies. Computer Communication Review. 34.

PCA PROJECTION OF DATA

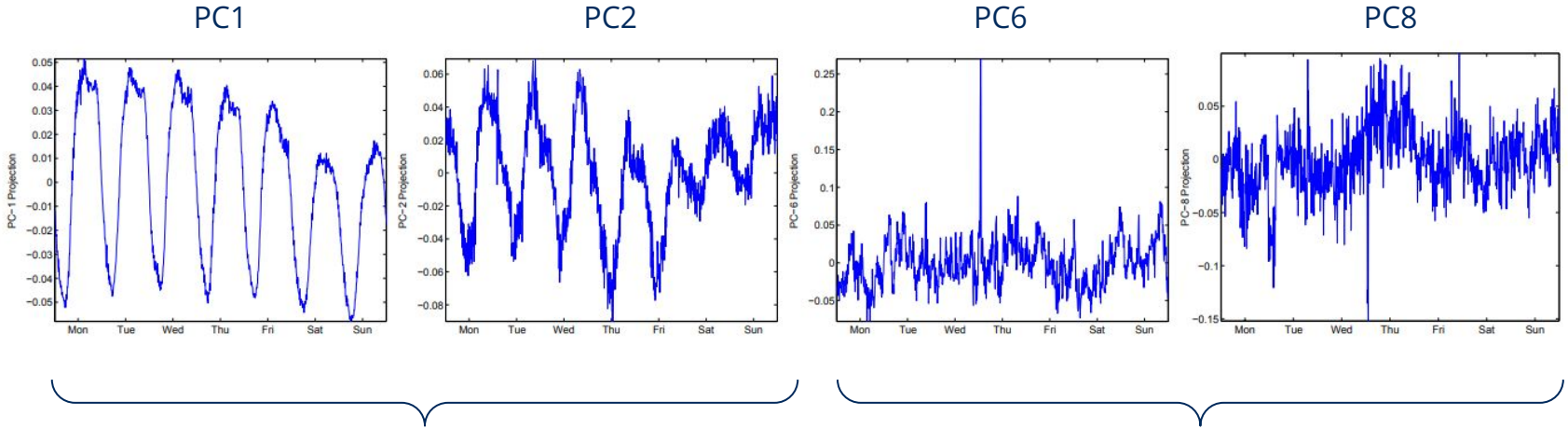


capture frequent (= normal) variance in data

capture infrequent (= anomalous) variance in data

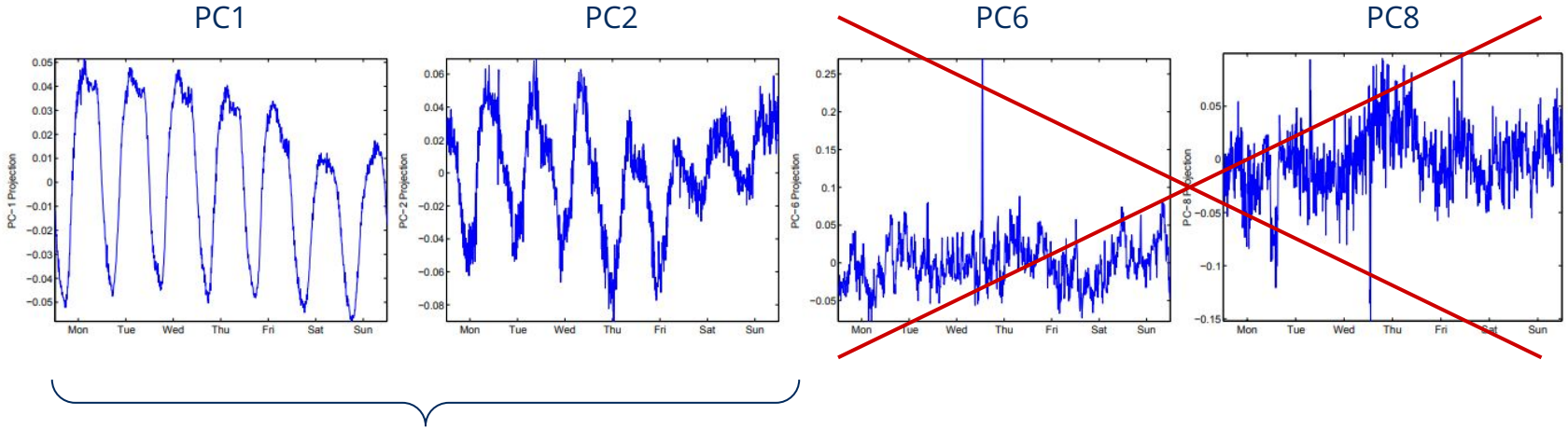
Lakhina, Anukool & Crovella, Mark & Diot, Christophe. (2004). Diagnosing Network-Wide Traffic Anomalies. Computer Communication Review. 34.

PCA PROJECTION OF DATA



Lakhina, Anukool & Crovella, Mark & Diot, Christophe. (2004). Diagnosing Network-Wide Traffic Anomalies. Computer Communication Review. 34.

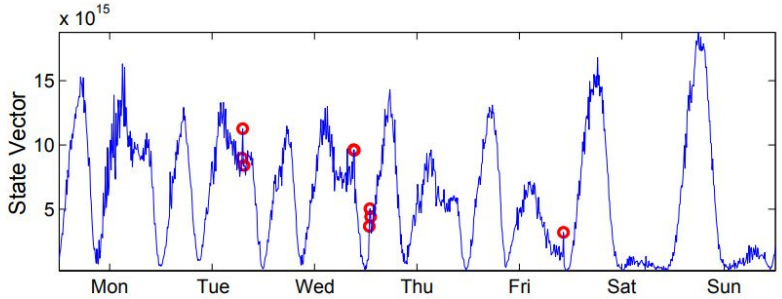
PCA PROJECTION OF DATA



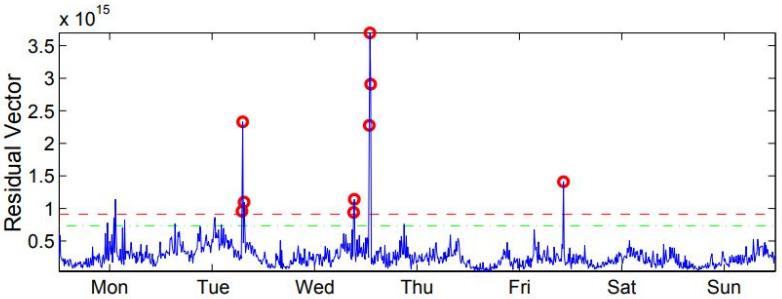
Lakhina, Anukool & Crovella, Mark & Diot, Christophe. (2004). Diagnosing Network-Wide Traffic Anomalies. Computer Communication Review. 34.

STATE VS. RESIDUAL MAGNITUDE

variance captured by normal PCA subspace



variance **not** captured by PCA subspace



99.9% Q-statistic

99.5% Q-statistic

Lakhina, Anukool & Crovella, Mark & Diot, Christophe. (2004). Diagnosing Network-Wide Traffic Anomalies. Computer Communication Review. 34.

“PCA subspace method”

for anomaly detection

OUTLINE

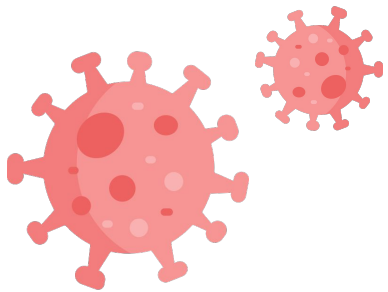
Understanding Anomaly Detectors

Understanding Poisoning of Anomaly Detectors

Defending against Poisoning of Anomaly Detectors

PCA SUBSPACE RECALCULATION

- 'Normal' network usage changes over time (e.g. during lockdowns)
- PCs and threshold get **recalculated every week**
- Values used in week w were learned in week w-1



ASSUMPTIONS

- We are the adversary
- We know that the network owner is using the PCA subspace method and when the subspaces get re-calculated (**grey box attack**)
- Our goal: launch DoS attack in a given week w
- Our objective: change PCA subspaces learned in previous week(s) such that we will not be detected in week w (**skewing attack**)
- Our means: add superfluous traffic ('chaff') into the network

POISONING THE PCA SUBSPACE METHOD: KNOWLEDGE LEVELS

Uninformed

We know nothing about the network traffic

Locally informed

We know how much traffic currently occurs at our ingress link

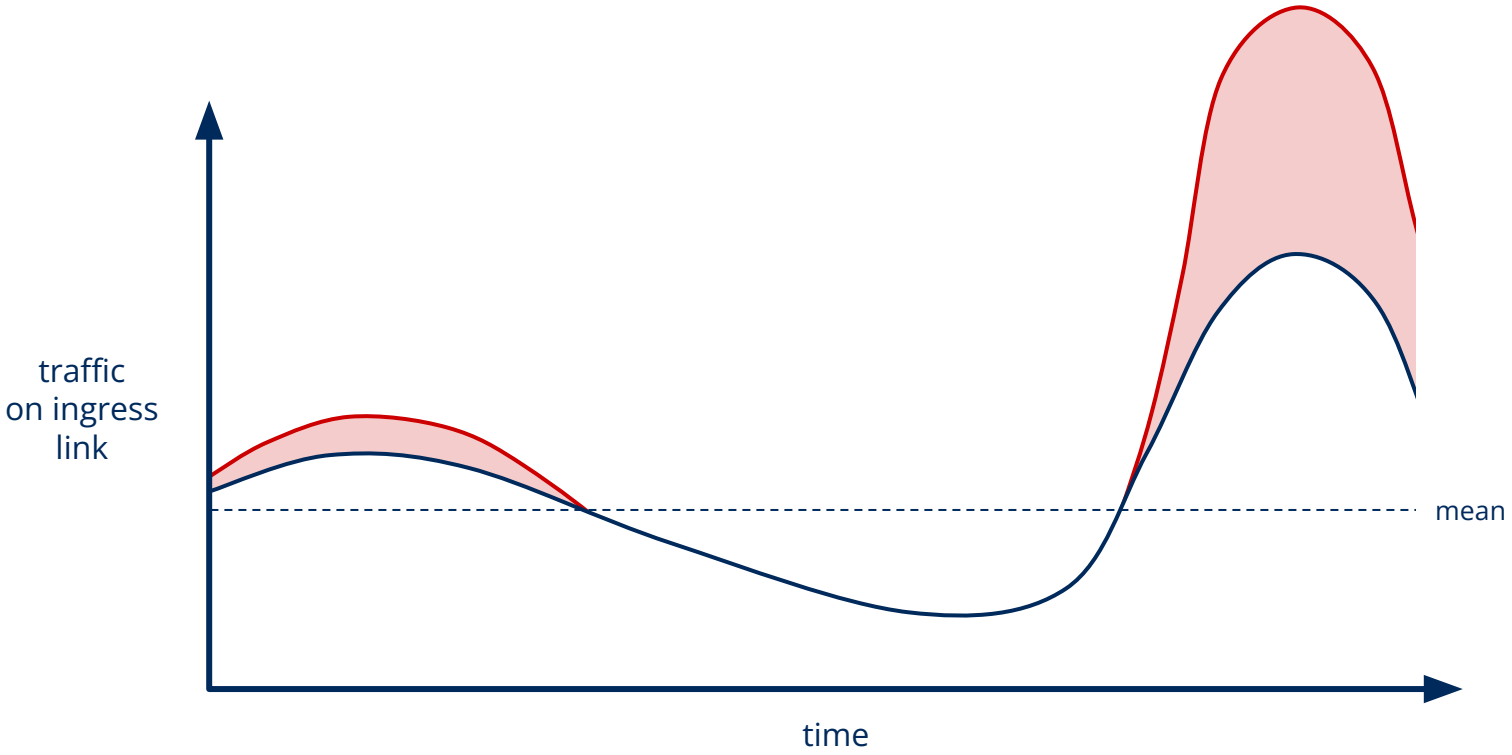
Globally informed

We know all current and future traffic volumes at all links and can insert traffic anywhere

KNOWLEDGE LEVELS: UNINFORMED

- Either add or don't add traffic in each timestep (according to random variable)
- Amount of traffic is fixed

KNOWLEDGE LEVELS: LOCALLY INFORMED



KNOWLEDGE LEVELS: GLOBALLY INFORMED

- Finding the right flows and chaff volumes is an optimization problem
- Unfortunately, this optimization is difficult to solve
- The authors make a few assumptions™ and do some math™
- the [...] assumption does not hold in practice
- The full proof is ommitted due to space constraints.
- Conclusion: even in this scenario we only want to add chaff along the links on the target OD flow

KNOWLEDGE LEVELS: METHODS

- **Increase** the overall **variance of traffic volumes**

  → Uninformed: randomly add constant amount of traffic

  → Locally informed: add more traffic if volumes are already high

  → Globally informed: like locally informed, but with more knowledge and more influenced links

- Chaff **must not be detected as anomaly** in order to be used for the calculation of PCs

POISONING THE PCA SUBSPACE METHOD: ATTACK DURATION

Single period attack

Chaff is only added in week w-1

Boiling frogs attack

Chaff is gradually increased from week w-n to w-1

USED DATA PER CONDITION

- **Single period**

  Used data: week 20 and 21 from Abilene dataset + synthetic anomalies in week 21

  We **pretend no chaff anomalies are detected** in week 20

- **Boiling frogs**

  Used data: synthesized traffic data + anomalies

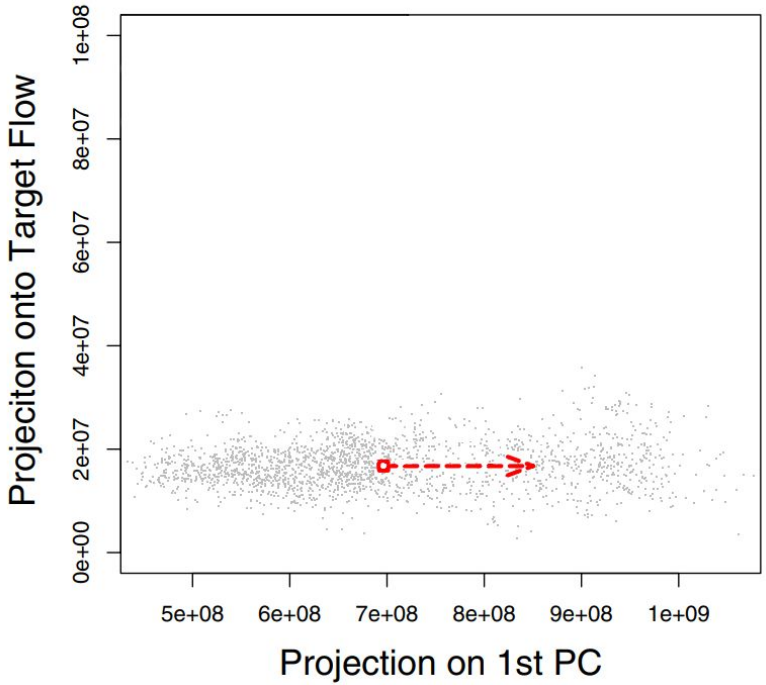
  'Normal' traffic does not change from week to week

  Chaff can be rejected as anomalous in every week

  Locally informed chaff insertion strategy

ATTACK EFFECT ON NORMAL SUBSPACE

initial PC1

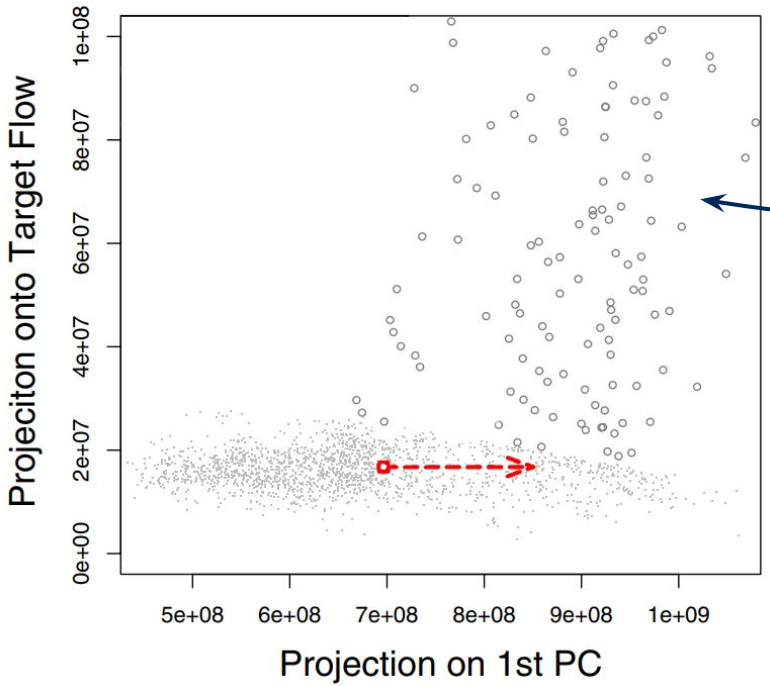


Rubinstein et al. (2009)

globally informed single period attack

ATTACK EFFECT ON NORMAL SUBSPACE

initial PC1



35% chaff

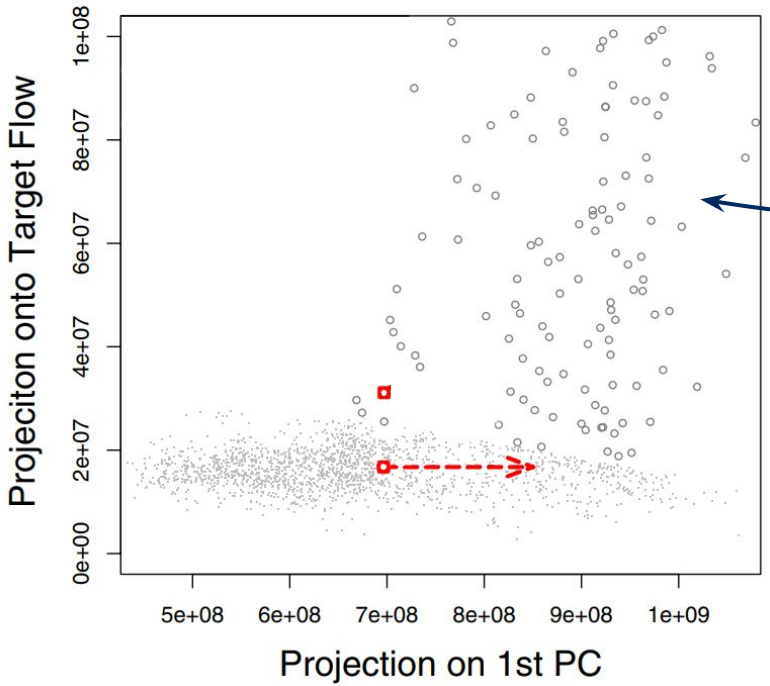
globally informed single period attack

Rubinstein et al. (2009)

ATTACK EFFECT ON NORMAL SUBSPACE

initial PC1

poisoned PC1



35% chaff

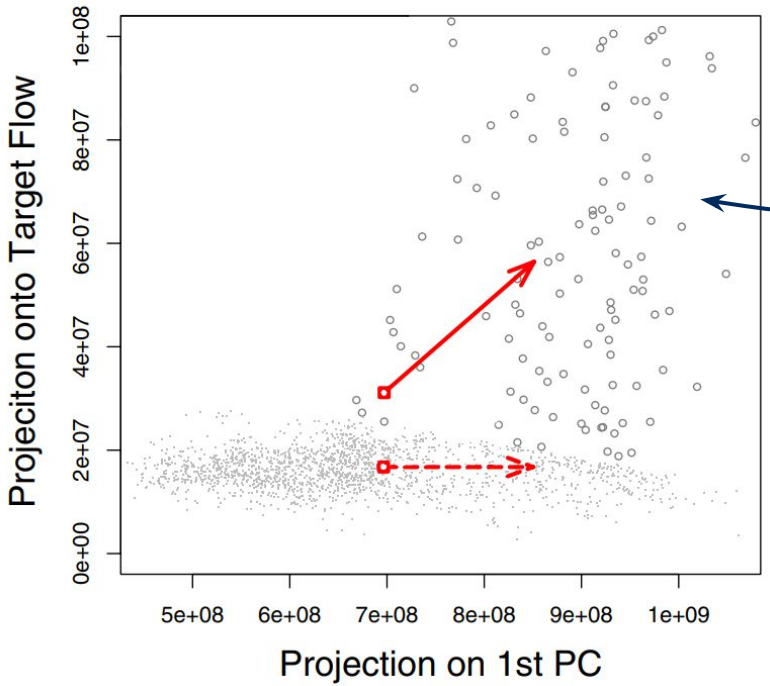
globally informed single period attack

Rubinstein et al. (2009)

ATTACK EFFECT ON NORMAL SUBSPACE

initial PC1

poisoned PC1

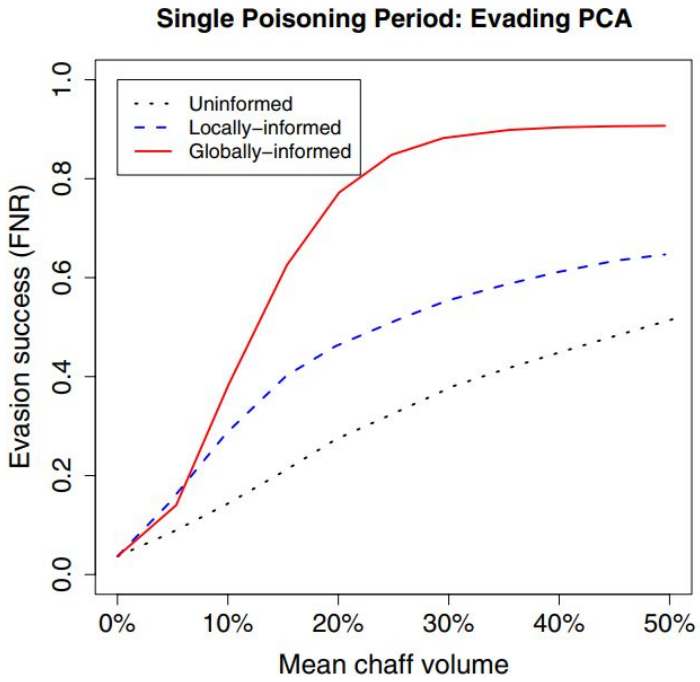


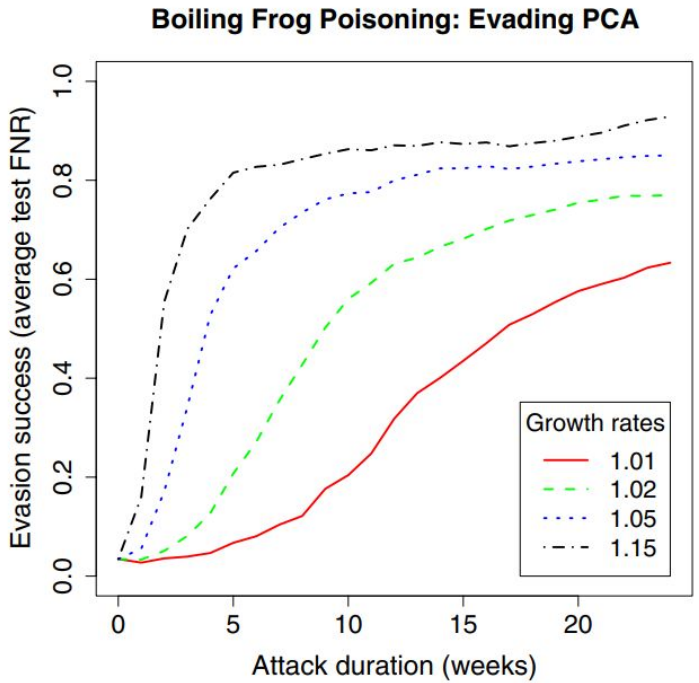
35% chaff

globally informed single period attack

Rubinstein et al. (2009)

POISONING EVASION SUCCESS (FALSE NEGATIVE) RATES





Rubinstein et al. (2009)

OUTLINE

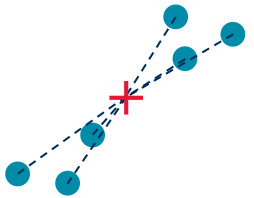
Understanding Anomaly Detectors

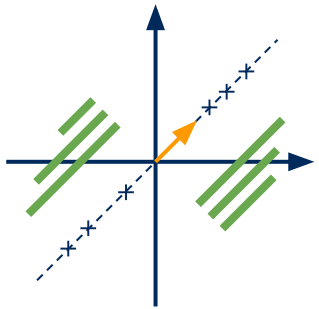
Understanding Poisoning of Anomaly Detectors

Defending against Poisoning of Anomaly Detectors

MEDIAN-BASED PRINCIPAL COMPONENT ANALYSIS

- PCA tries to capture the dispersion of the data
- Some statistics are more or less sensitive to outliers
- Instead of centering data around **mean**, center it around **spatial median** (= location estimate)
- Replace **mean squared distance** (= variance) by **median absolute deviation** (MAD) (= dispersion measure)
- PCs are found using **PCA-GRID** (uses grid search)

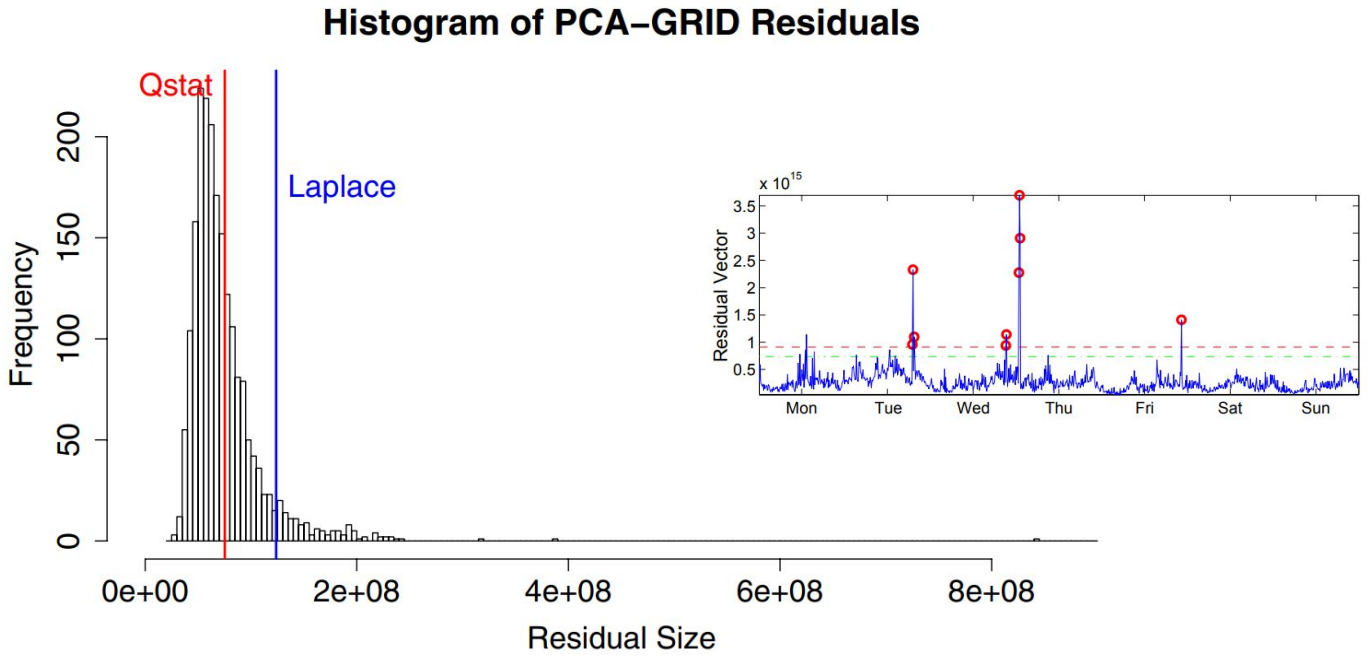




PCA-GRID (Croux et al., 2007)

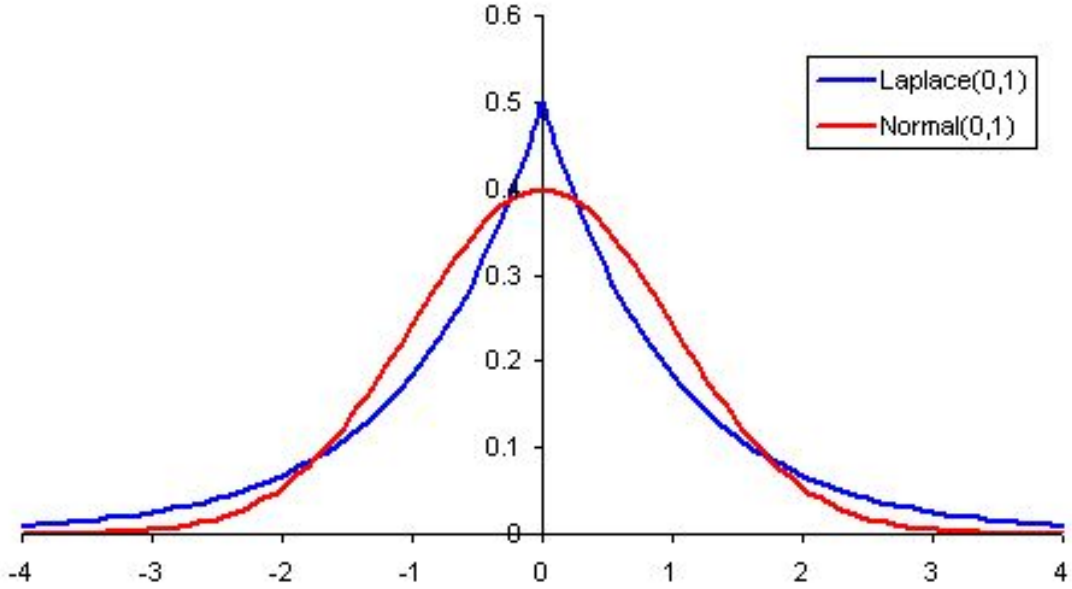
- Given location estimate and dispersion measure PCA-GRID finds the principal components via **grid search**
- Iteratively divide the search space up into subspaces using candidate vectors, refining the angle of the best performing vector
- This yields an approximate solution that maximizes the dispersion measure
- Project and repeat for multiple PCs

LAPLACE VS. Q-STATISTIC THRESHOLD



Rubinstein et al. (2009)

LAPLACE-DISTRIBUTION VS. Q-DISTRIBUTION



https://www.vosesoftware.com/riskwiki/images/image15_632.gif

ANTIDOTE

median-based PCA-GRID + Laplace threshold = ANTIDOTE

ATTACK EFFECT ON NORMAL SUBSPACE

PCA

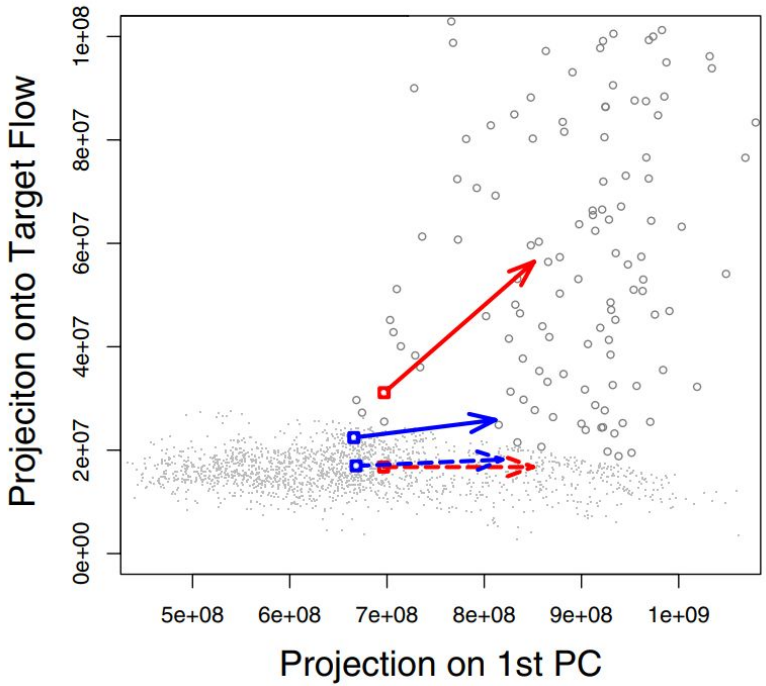
initial PC1

poisoned PC1

ANTI-DOTE

initial PC1

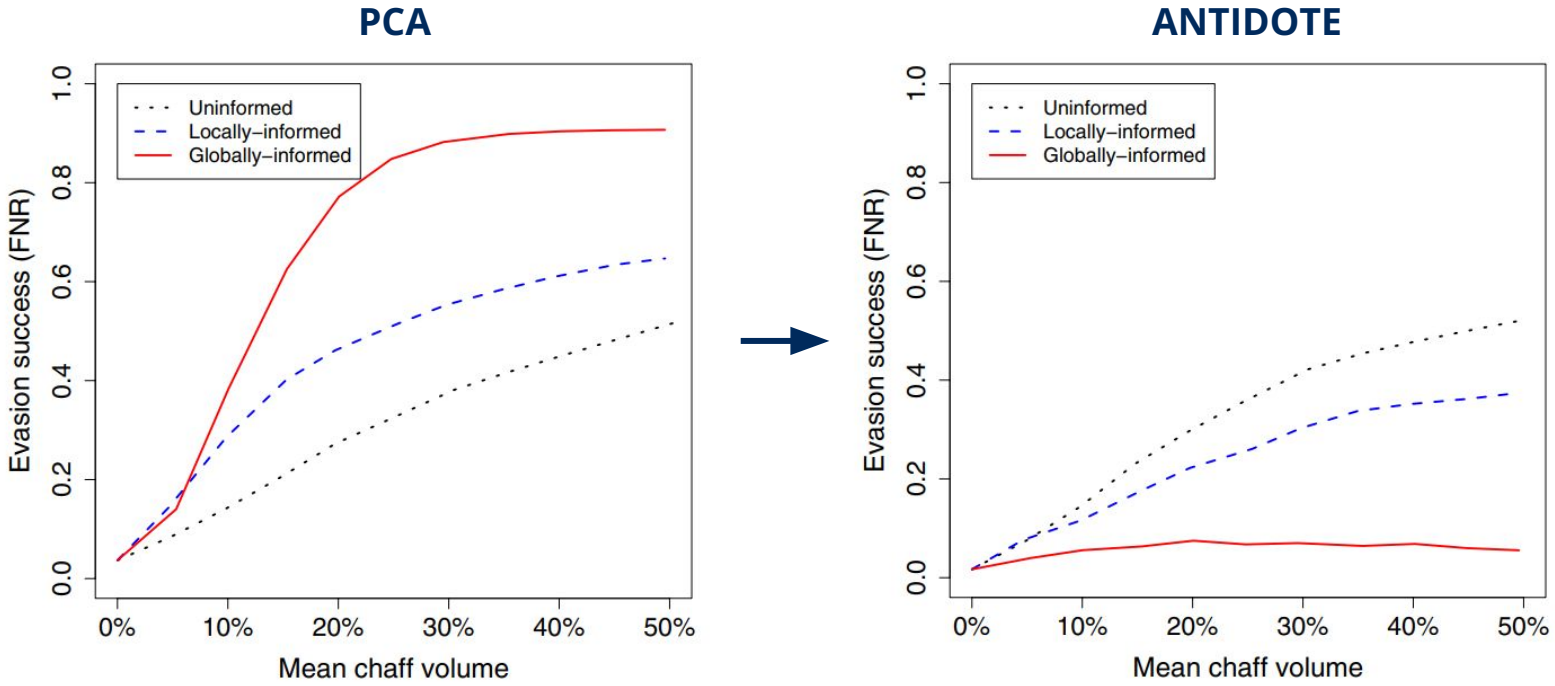
poisoned PC1



Rubinstein et al. (2009)

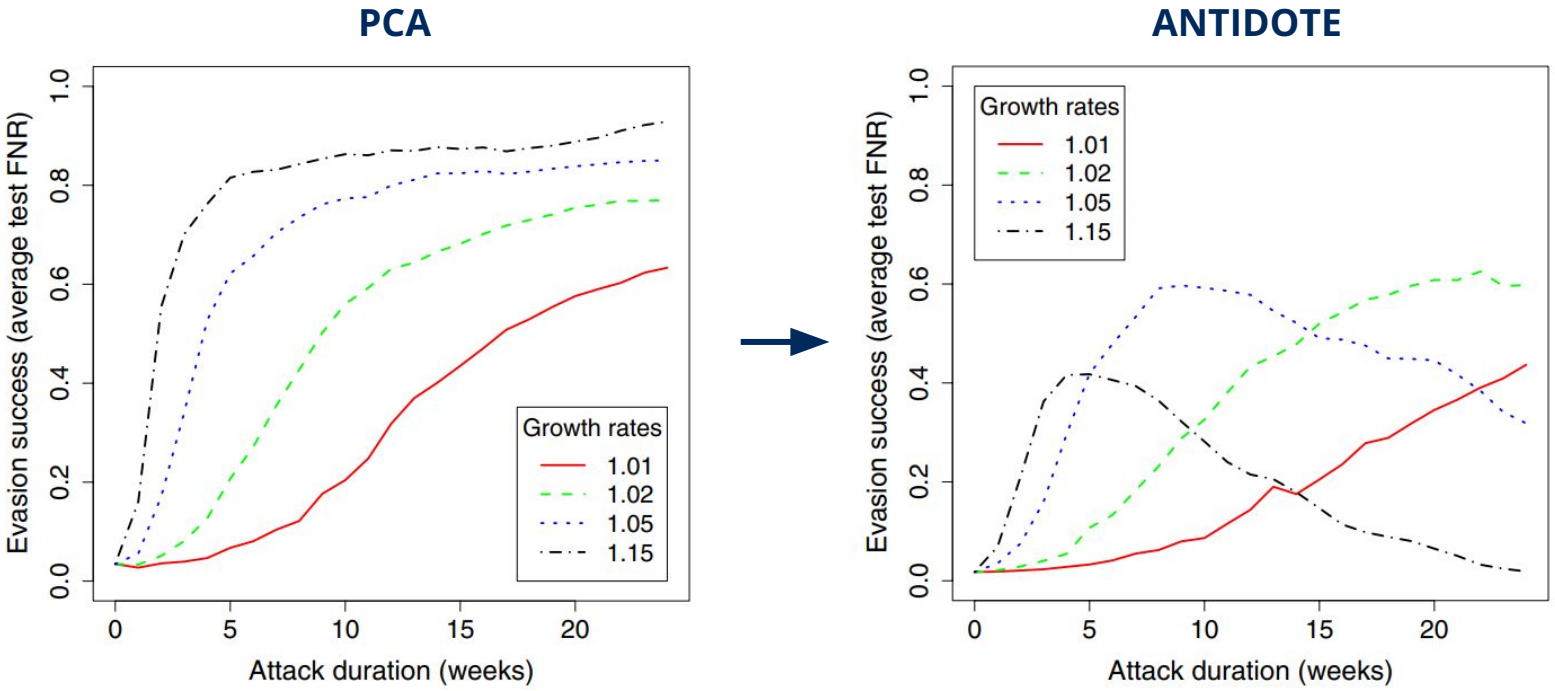
35% chaff globally informed single period attack

POISONING EVASION SUCCESS RATES: SINGLE PERIOD



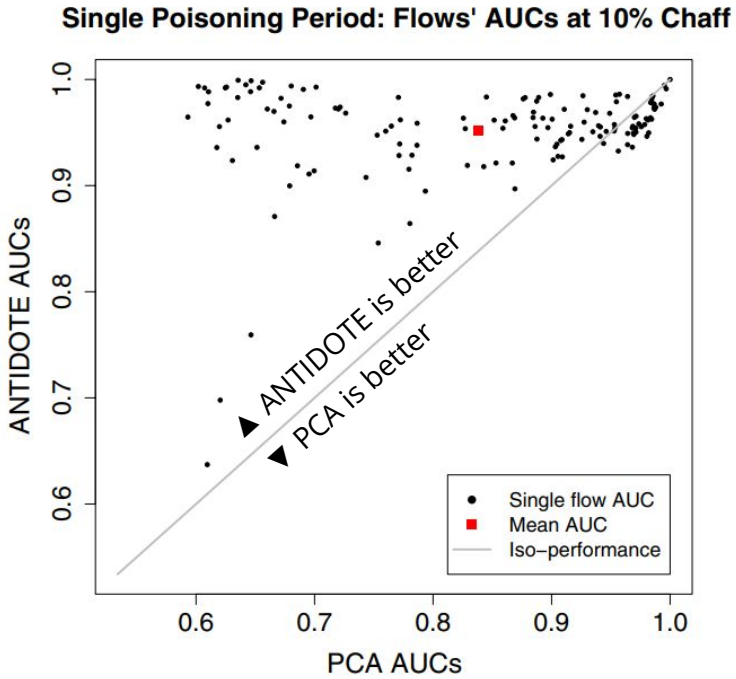
Rubinstein et al. (2009)

POISONING EVASION SUCCESS RATES: BOILING FROGS



Rubinstein et al. (2009)

AREA UNDER CURVE (AUC): PCA VS. ANTIDOTE



- = single OD flow

Rubinstein et al. (2009)

SHORTCOMINGS: UNREALISTIC DATA

- Abilene was a university research network, not public internet traffic
- Week 20 and 21 were cherry-picked
- The anomalies were just simulated
- The 'normal' traffic was simulated and completely stationary (boiling frogs)
- Authors acknowledge the approach didn't work using real non-stationary data (boiling frogs)

SHORTCOMINGS: UNREALISTIC KNOWLEDGE LEVEL

- Authors claim the unrealistic 'globally informed' strategy was included to test the limitations of ANTIDOTE
- However, they report their best results unter this strategy
- If you remove the results on the 'globally informed' strategy, you are left with

  → no effect on the uninformed strategy

  → 50% reduction for the locally informed strategy

  and those **only if 100% of chaff remains undetected** in week w-1

SHORTCOMINGS: DOS VARIANCE IS FREQUENT

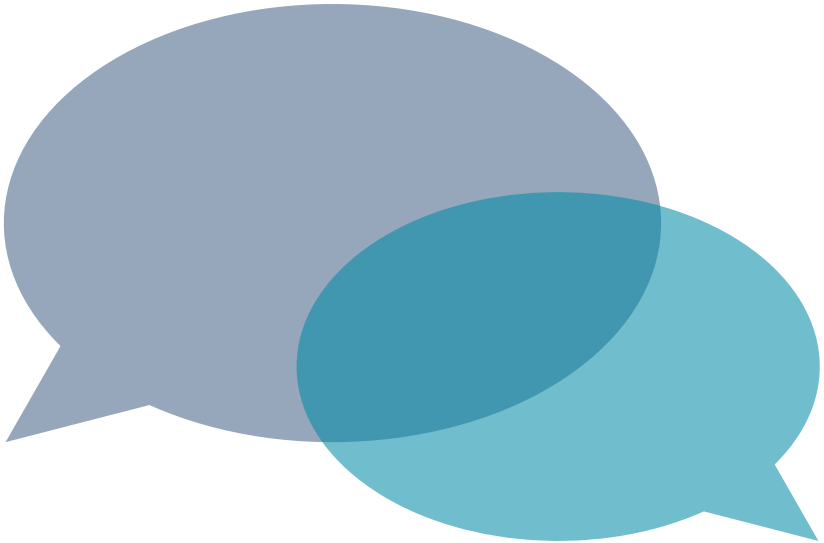
- The authors assume DoS attacks are outliers
- There are around 28,700 distinct DoS attacks per day [1]
- If DoS attacks are ubiquitous, they will be encoded in the normal subspace no matter which method we use
- Distributed DoS attacks using botnets are even less detectable

[1] Jonker, Mattijs & King, Alistair & Krupp, Johannes & Rossow, Christian & Sperotto, Anna & Dainotti, Alberto. (2017). Millions of targets under attack: a macroscopic characterization of the DoS ecosystem. 100-113.

IMPROVEMENTS

- Try to gather genuine and more recent network data
- Don't pretend all chaff remains undetected (single period)
- Cross-validate on multiple weeks
- Report results on locally informed scheme
- Release code

DISCUSSION



CONCLUSION

- Good explanation on how PCA subspace poisoning works by shifting normal PCs
- Systematic comparison of different attack strategies, some of which are quite successful (e.g. random chaff, boiling frogs)
- Provides evidence for robust median-based PCA
- But: unrealistic performance for both attack and defense due to utilized data and assumptions made

SHORTCOMINGS: OTHERS

- Projection on 1st PC in graph is not centered around 0, implying that the data was not standardized

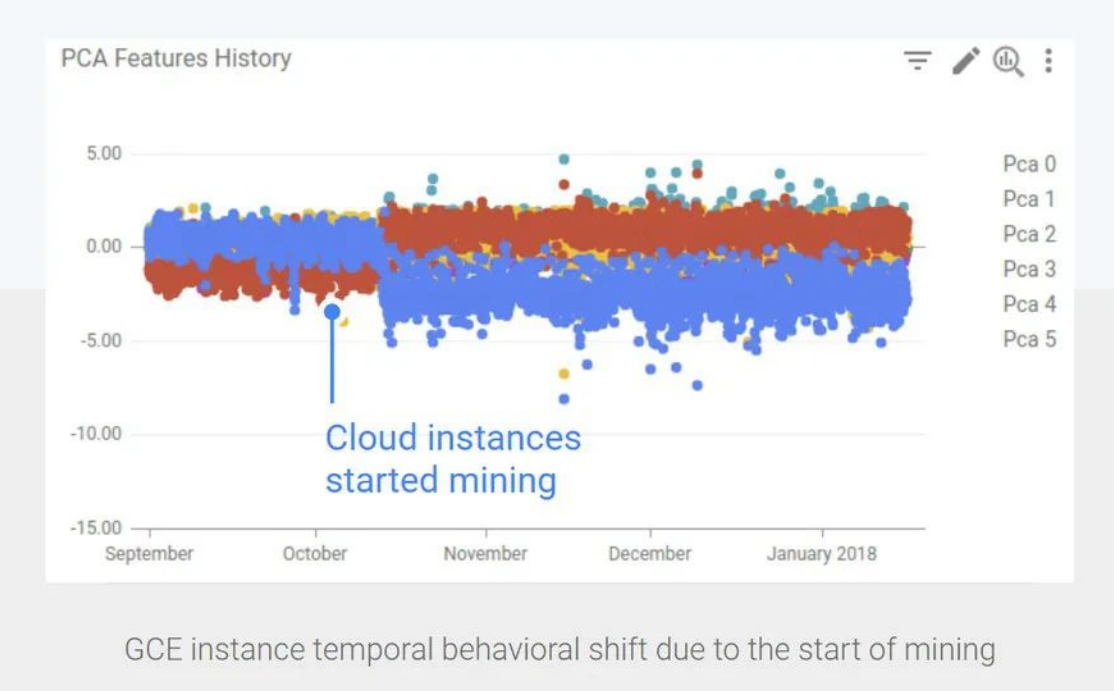
  → mismatch between what authors claim to do and show in graph

- PCA subspace method does not consider order of data points

- Evasion success rate unrealistic: if you fail once, you get blocked and can't retry the attack 1000 times

- No code was released

PCA IN REAL LIFE: DETECTING CLOUD MINING

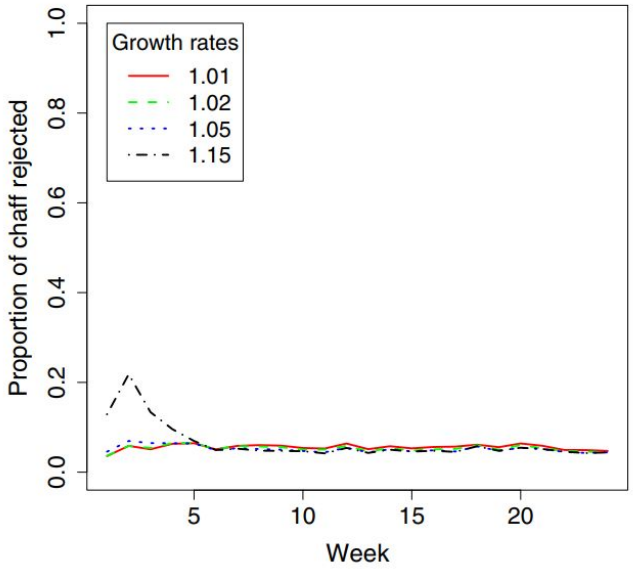


GCE instance temporal behavioral shift due to the start of mining

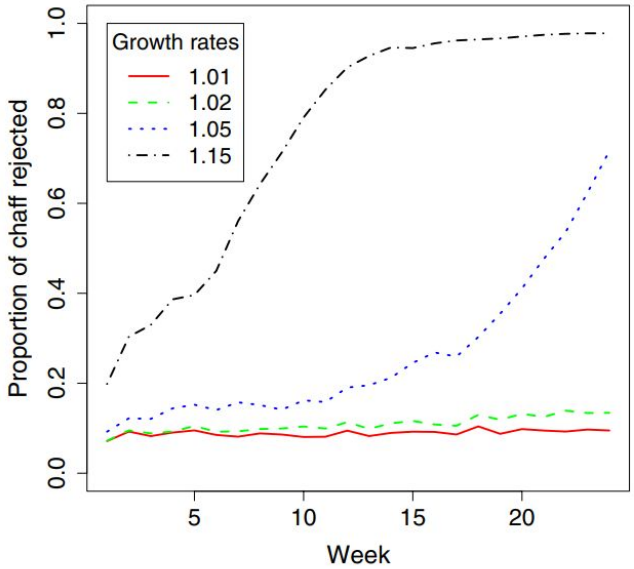
https://elie.net/blog/ai/attacks-against-machine-learning-an-overview/

REJECTION RATES (BOILING FROGS)

PCA

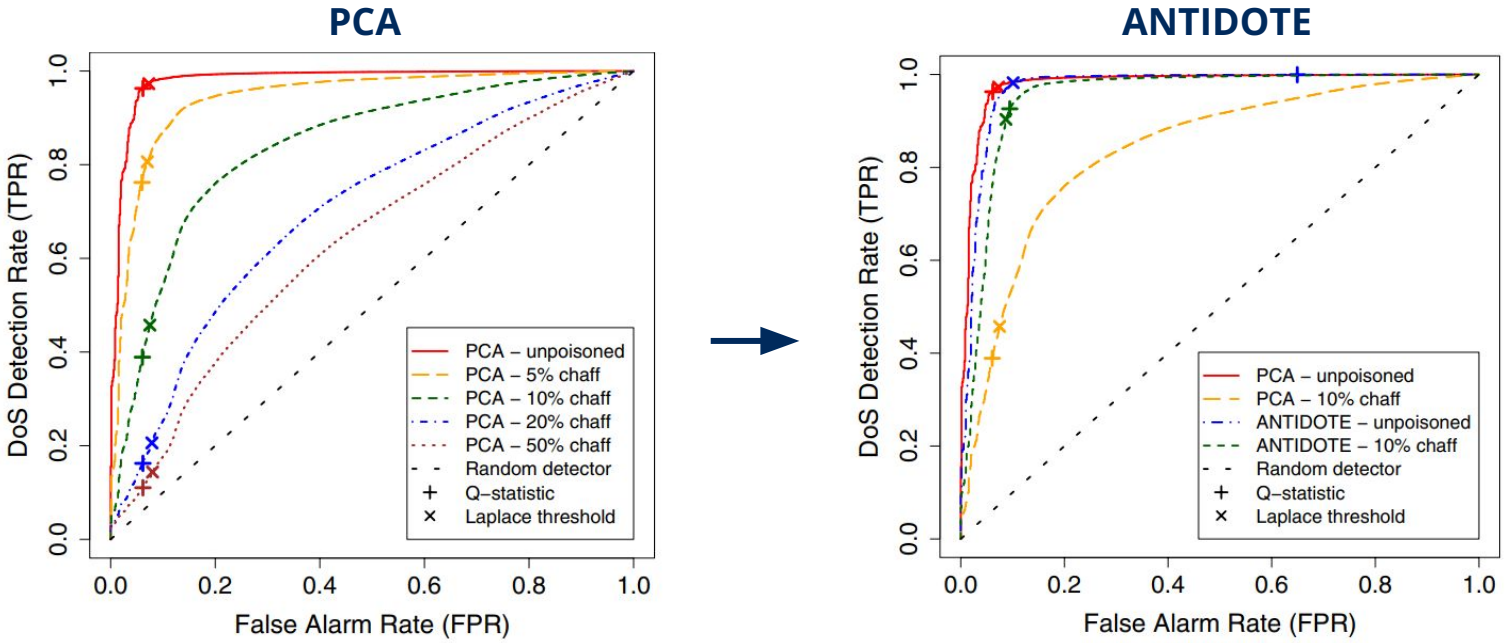


ANTIDOTE



Rubinstein et al. (2009)

ROC CURVES (SINGLE PERIOD)



Rubinstein et al. (2009)

2

Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning

Matthew Jagielski*, Alina Oprea*, Battista Biggio †‡, Chang Liu§, Cristina Nita-Rotaru*, and Bo Li§

*Northeastern University, Boston, MA †University of Cagliari, Italy ‡Pluribus One, Italy §UC Berkeley, Berkeley, CA

***Abstract*—As machine learning becomes widely used for automated decisions, attackers have strong incentives to manipulate the results and models generated by machine learning algorithms. In this paper, we perform the first systematic study of poisoning attacks and their countermeasures for linear regression models. In poisoning attacks, attackers deliberately influence the training data to manipulate the results of a predictive model. We propose a theoretically-grounded optimization framework specifically designed for linear regression and demonstrate its effectiveness on a range of datasets and models. We also introduce a fast statistical attack that requires limited knowledge of the training process. Finally, we design a new principled defense method that is highly resilient against all poisoning attacks. We provide formal guarantees about its convergence and an upper bound on the effect of poisoning attacks when the defense is deployed. We evaluate extensively our attacks and defenses on three realistic datasets from health care, loan assessment, and real estate domains.[1] [2]**

I. Introduction

As more applications with large societal impact rely on machine learning for automated decisions, several concerns have emerged about potential vulnerabilities introduced by ma-

training process. Such poisoning attacks have been practically demonstrated in worm signature generation [42], [45], spam filters [40], DoS attack detection [47], PDF malware classification [55], handwritten digit recognition [5], and sentiment analysis [41]. We argue that these attacks become easier to mount today as many machine learning models need to be updated regularly to account for continuously-generated data. Such scenarios require *online training*, in which machine learning models are updated based on new incoming training data. For instance, in cyber-security analytics, new Indicators of Compromise (IoC) rise due to the natural evolution of malicious threats, resulting in updates to machine learning models for threat detection [23]. These IoCs are collected from online platforms like VirusTotal, in which attackers can also submit IoCs of their choice. In personalized medicine, it is envisioned that patient treatment is adjusted in real-time by analyzing information crowdsourced from multiple participants [16]. By controlling a few devices, attackers can submit fake information (e.g., sensor measurements), which is then used for training models applied to a large set of pa-

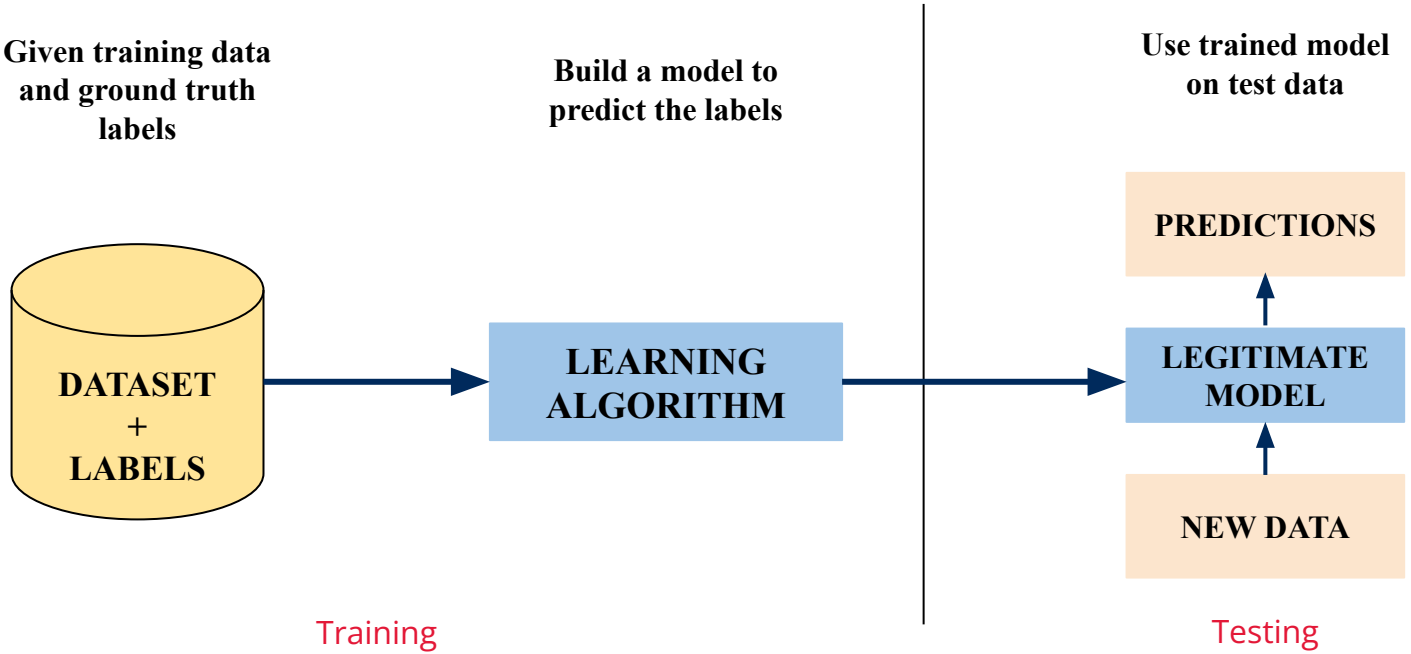
MOTIVATION

- Existence of strong incentives to manipulate the automated decisions by machine learning model.
- No robust algorithm for detecting poisoning points in regression models

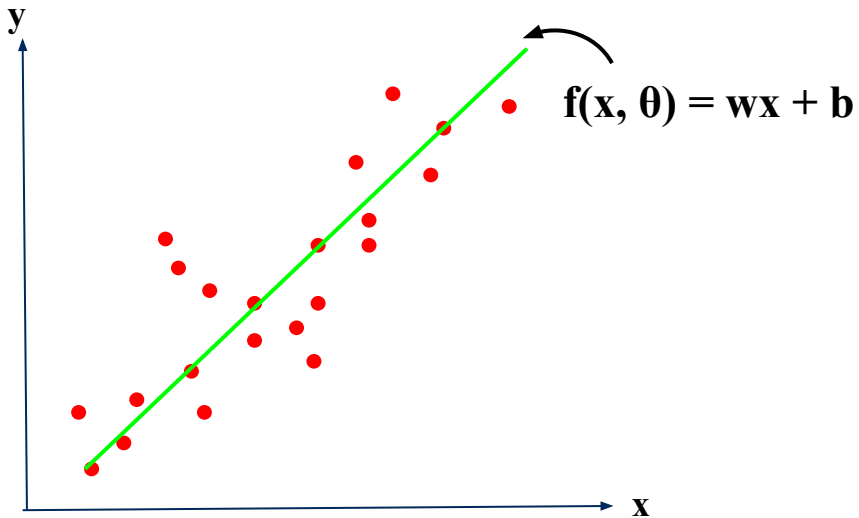
CONTRIBUTIONS

- Theoretically grounded optimization framework tuned for regression models.
- Designed a statistical attack method
- Robust defense algorithm to tackle inliers.
- First to consider poisoning attacks against regression models

SUPERVISED MACHINE LEARNING



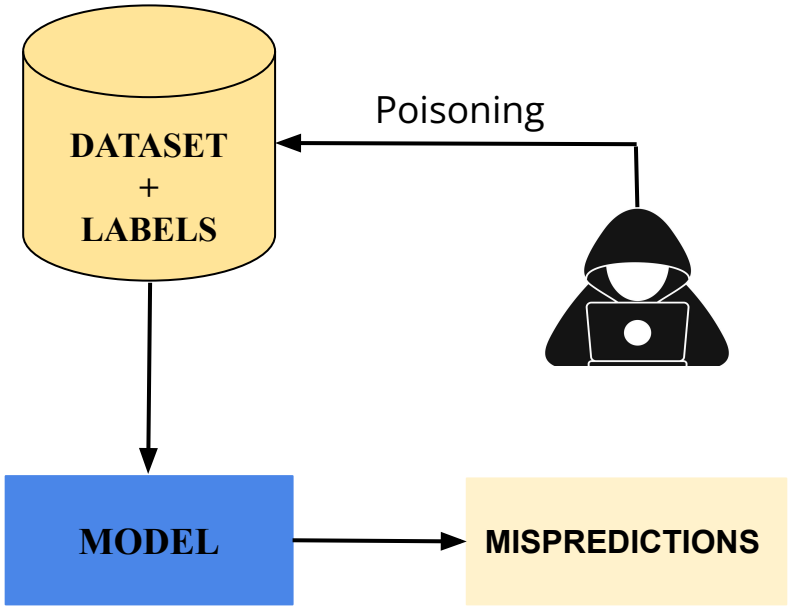
LINEAR REGRESSION



$$\mathcal{L}(\mathcal{D}_{\text{tr}}, \theta) = \underbrace{\frac{1}{n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i, \boldsymbol{\theta}) - y_i)^2}_{\text{MSE}(\mathcal{D}_{\text{tr}}, \boldsymbol{\theta})} + \lambda\Omega(\boldsymbol{w})$$

y → predictions
x → data points
w → weights
b → biases

ATTACK METHODOLOGY



White box attacks:

$$\arg\max_{\mathcal{D}_p} \quad \mathcal{W}(\mathcal{D}', \boldsymbol{\theta}_p^\star)$$
$$\text{s.t.} \quad \boldsymbol{\theta}_p^\star \in \arg\min_\theta \mathcal{L}(\mathcal{D}_{\text{tr}} \cup \mathcal{D}_p, \boldsymbol{\theta})$$

D' - untainted dataset
θ_p^* - poisoned regression parameters
D_{tr} - training dataset
D_{tr}'- substitute dataset

Black box attacks:

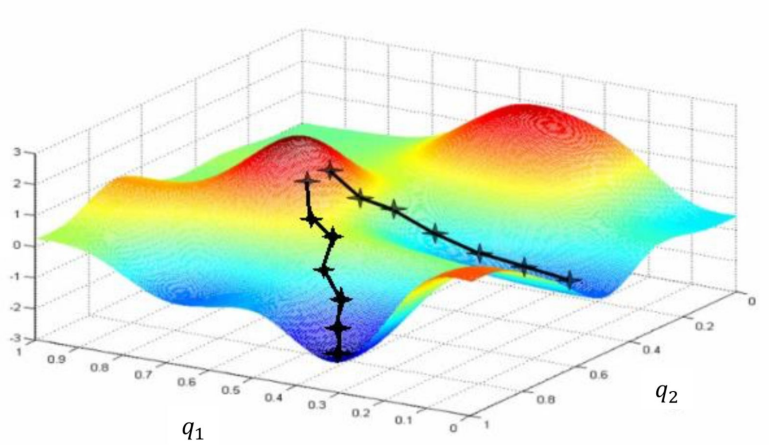
- The attacker do not know the knowledge of the dataset.
- Replace the original dataset with substitute dataset

BASELINE GRADIENT DESCENT (BGD) ALGORITHM

- Attacker's goal: maximize the regularized loss function
- Uses gradient ascent to optimize the poisoning points
- Challenges:
  - Inner learning problem - neural network is not convex and hence requires efficient numerical approximations

OPTIMIZATION BASED POISONING ATTACK:

- Start with a set of initial points.
- Gradient ascent - the direction which improves attacker's objective.
- Iteratively update each point with gradient ascent.
- Stop at convergence and output poisoning points.



ATTACK METHODOLOGY

Algorithm 1 Poisoning Attack Algorithm

Input: $\mathcal{D} = \mathcal{D}_{\text{tr}}$ (white-box) or $\mathcal{D}'_{\text{tr}}$ (black-box), $\mathcal{D}'$, $\mathcal{L}$, $\mathcal{W}$, the initial poisoning attack samples $\mathcal{D}_p^{(0)} = (x_c, y_c)_{c=1}^{p}$, a small positive constant ε.

1: $i \leftarrow 0$ (iteration counter)
2: $\theta^{(i)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D} \cup \mathcal{D}_p^{(i)}, \theta)$
3:
4: $w^{(i)} \leftarrow \mathcal{W}(\mathcal{D}', \theta^{(i)})$
5: $\theta^{(i+1)} \leftarrow \theta^{(i)}$
6: **for** $c = 1, \ldots, p$ **do**
7: line_search $\left(x_c^{(i)}, \nabla_{\boldsymbol{x}_c} \mathcal{W}(\mathcal{D}', \theta^{(i-1)})\right)$
8: $\theta^{(i+1)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D} \cup \mathcal{D}_p^{(i+1)}, \theta)$
9: $w^{(i+1)} \leftarrow \mathcal{W}(\mathcal{D}', \theta^{(i+1)})$
10: $i \leftarrow i + 1$
11: **until** $|w^{(i)} - w^{(i-1)}| < \varepsilon$

Output: the final poisoning attack samples $\mathcal{D}_p \leftarrow \mathcal{D}_p^{(i)}$

Considers KKT equilibrium condition - assumes derivative w.r.t x_c is in equilibrium

Complex gradient computation due to lack of closed form expression

STATISTICAL BASED POISONING ATTACK

- Points drawn from similar distribution as the training data
- Need to know the mean and covariance of the training data
- Black box access to the model

$$(X^T X)^{-1}(X^T Y) \rightarrow \boxed{(X^T X)^{-1}}((1-\alpha)X^T Y + \boxed{\alpha X_p^T Y_p})$$

Keeps covariance similar

Modifies the correlations

EXISTING DEFENCES:

EXISTING DEFENSE PROPOSALS	PROS	CONS
HUBER	• Noise resilient regression • Identifies and removes outliers	• Reduces the growth of loss function during large errors. • Incorporates poisoning points
RANSAC		• Samples and checks whether enough points fit the model well, random samples might contain poisoning points
Chen et al.,	• Robust algorithm for poisoning	• Made unrealistic assumptions like sub-Gaussian data and noise.
RONI	• Suitable for spam scenario	• Identifies only outliers with high impact

TRIM ALGORITHM

- Given a set of data points (n) and $\propto$n poisoning points, TRIM tries to find the best n data points out of (1 + $\propto$)n points
- The regression parameters $\theta = (w, b)$ are unknown and no assumptions are made on the distribution of the data points.
- TRIM estimates a model and identifies the points with the lowest residual from the training set.

TRIM ALGORITHM

Algorithm 2 [TRIM algorithm]

1: **Input**: Training data $\mathcal{D} = \mathcal{D}_{\text{tr}} \cup \mathcal{D}_p$ with $|\mathcal{D}| = N$; number of attack points $p = \alpha \cdot n$.
2: **Output**: θ.
3: $\mathcal{I}^{(0)} \leftarrow \{1, ..., N\}$ /* First train with all samples */
$\arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}^{\mathcal{I}^{(0)}}, \theta)$ /* Initial estimation of θ*
/* Iteration count */
$+ 1$;
8: $\mathcal{I}^{(i)} \leftarrow$ subset of size n that min. $\mathcal{L}(\mathcal{D}^{\mathcal{I}^{(i)}}, \theta^{(i-1)})$
9: $\theta^{(i)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}^{\mathcal{I}^{(i)}}, \theta)$ /* Current estimator */
$R^{(i)} = \mathcal{L}(\mathcal{D}^{\mathcal{I}^{(i)}}, \theta^{(i)})$ /* Current loss */
$R^{(i)} = R^{(i-1)}$ /* Convergence condition*/
return $\theta^{(i)}$ /* Final estimator */.

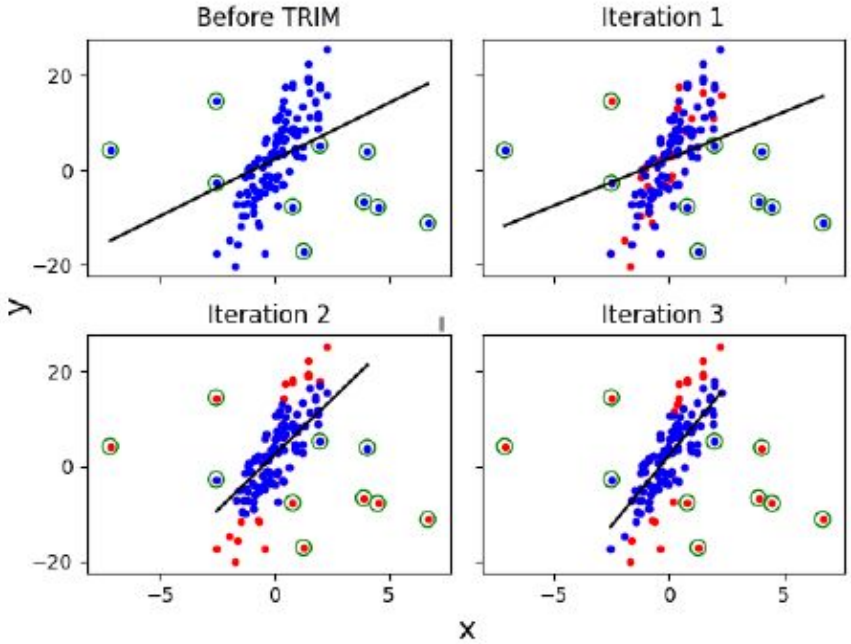
Updates the model with the subset of data points

Selects a subset of points from the dataset which has the lowest residuals

Stops training when there's no change in the gradients

Jagielski et al., (2021)

TRIM ALGORITHM



Jagielski et al., (2021)

EXPERIMENTAL EVALUATION

The experiments are conducted on three datasets for

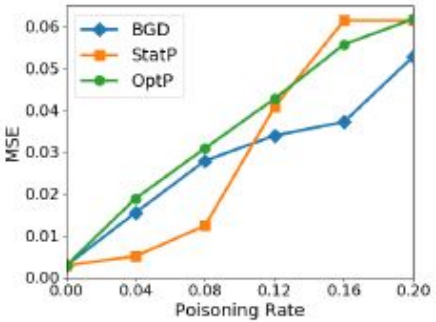
1. Ridge regression
2. LASSO

The datasets are:

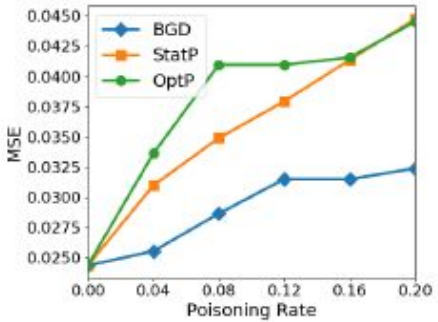
- Health care dataset
- Loan dataset
- House Price dataset

COMPARISON OF ATTACK METHODS

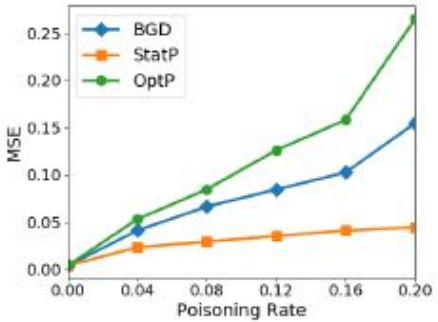
RIDGE REGRESSION:



(a) Health Care Dataset

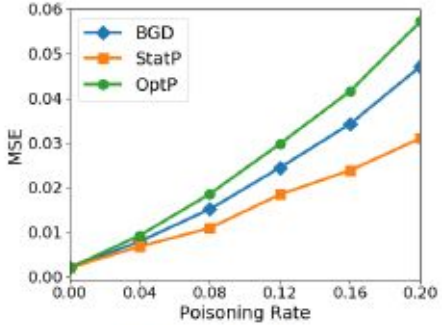


(b) Loan Dataset

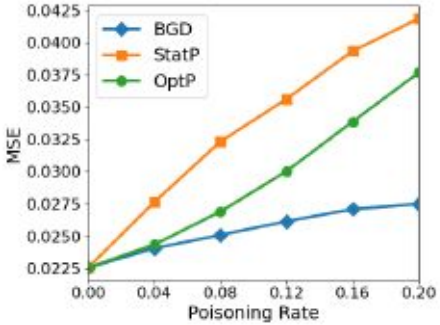


(c) House Price Dataset

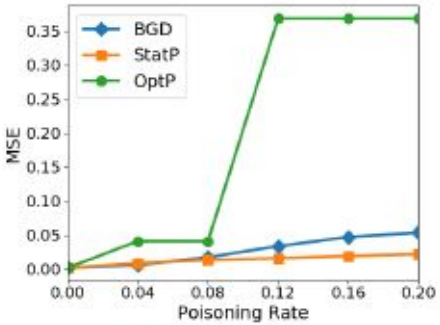
LASSO:



(a) Health Care Dataset



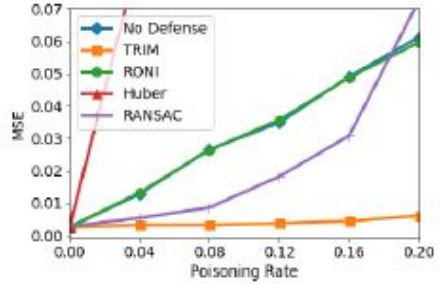
(b) Loan Dataset



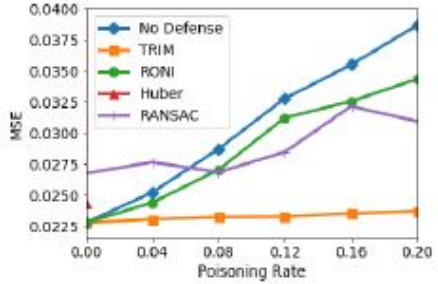
(c) House Price Dataset

COMPARISON BETWEEN TRIM AND EXISTING DEFENCES

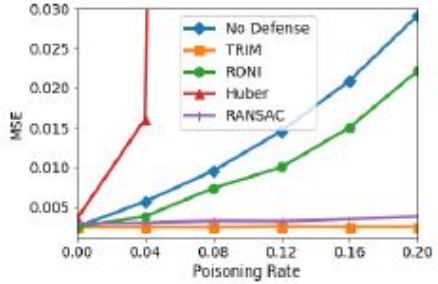
RIDGE REGRESSION:



(a) Health Care Dataset

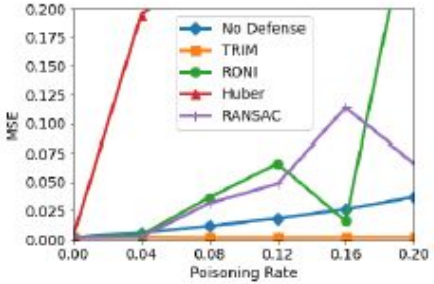


(b) Loan Dataset

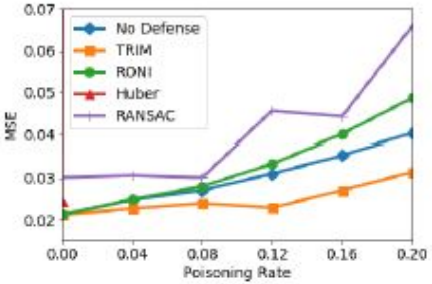


(c) House Price Dataset

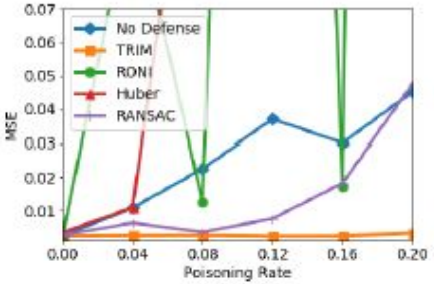
LASSO:



(a) Health Care Dataset



(b) Loan Dataset



(c) House Price Dataset

LIMITATIONS

- Conducted experiments only on two linear regression models:
  - LASSO
  - Ridge regression

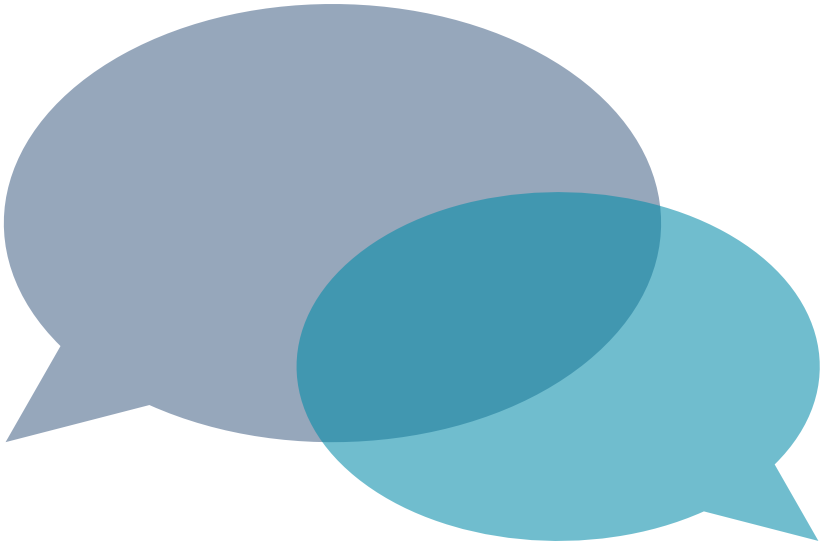
while considered four models in the paper (OLS, LASSO, Ridge regression, Elastic-net)

- The proposed attack strategy (StatP) didn't outperform the BGD (baseline attack strategy) on few datasets.
- Randomly samples a subset of data points from a large dataset without analyzing the dataset
- Considered a small dataset for experiments and concluded that TRIM converges in a finite number of iterations
- Released code but - unstructured and have unsolved issues

IMPROVEMENTS

- Use of larger datasets for the experiments might help
- Other linear regression models can also be incorporated:
  - OLS
  - Elastic-net
- Could possibly extend the experiments to polynomial regression as well.

DISCUSSION



3

SEVER: A Robust Meta-Algorithm for Stochastic Optimization

Ilias Diakonikolas *
CS, USC
diakonik@usc.edu

Gautam Kamath †
CS, University of Waterloo
g@csail.mit.edu

Daniel M. Kane ‡
CSE & Math, UCSD
dakane@cs.ucsd.edu

Jerry Li §
Microsoft Research AI
jerryzli@mit.edu

Jacob Steinhardt ¶
Statistics, UC Berkeley
jsteinha@stanford.edu

Alistair Stewart ‖
Web3 Foundation
stewart.al@gmail.com

May 31, 2019

Abstract

In high dimensions, most machine learning methods are brittle to even a small fraction of structured outliers. To address this, we introduce a new meta-algorithm that can take in a *base learner* such as least squares or stochastic gradient descent, and harden the learner to be resistant to outliers. Our method, SEVER, possesses strong theoretical guarantees yet is also highly scalable—beyond running the base learner itself, it only requires computing the top singular vector of a certain $n \times d$ matrix. We apply SEVER on a drug design dataset and a spam classification dataset, and find that in both cases it has substantially greater robustness than several baselines. On the spam dataset, with 1% corruptions, we achieved 7.4% test error, compared to $13.4\% - 20.5\%$ for the baselines, and 3% error on the uncorrupted dataset. Similarly, on the drug design dataset, with 10% corruptions, we achieved 1.42 mean-squared error test error, compared to $1.51 - 2.33$ for the baselines, and 1.23 error on the uncorrupted dataset.

1 Introduction

Learning in the presence of outliers is a ubiquitous challenge in machine learning; nevertheless, most machine

Paper presentation: Sever: A Robust Meta-Algorithm for Stochastic Optimization

Jiaqi Wang

MOTIVATION

1. Mislabelling and Measurement Errors can cause Systematic outliers
2. Outliers can be introduced by poisoning attack

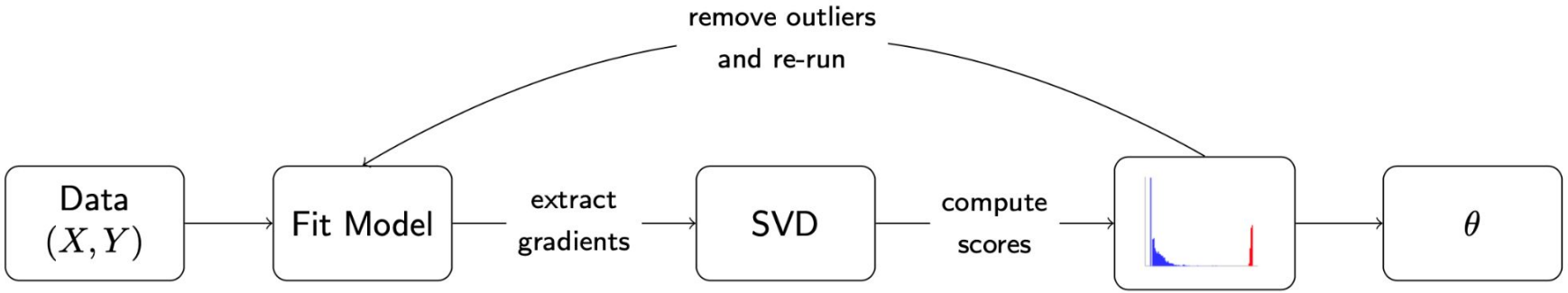
PROBLEMS WITH STATE-OF-THE-ART

1. Fails when data are high-dimensional
2. Only works for obvious outliers, doesn't work on the correlated outliers
3. Specific to some algorithms, lack generalization
4. Significant loss in performance

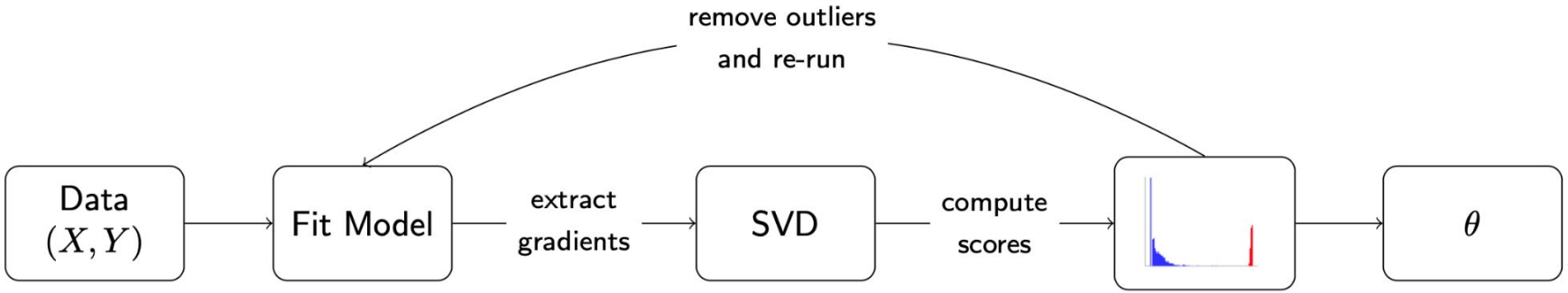
NOVELTY

1. Works in high-dimensional data space
2. Robust to arbitrary outliers with small decrease in performance
3. Applicable to most ML models, including regression and classification tasks, and non-convex models like neural networks

PIPELINE

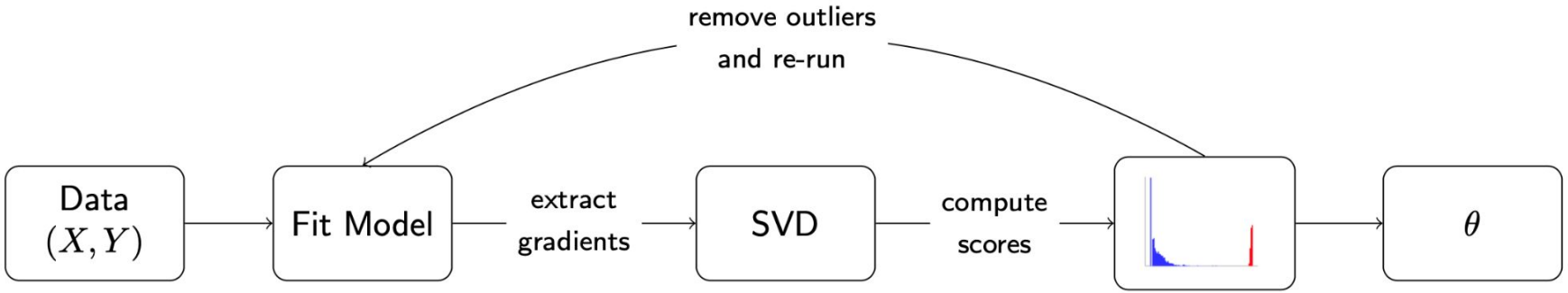

remove outliers
and re-run
Data
(X, Y)
Fit Model
extract
gradients
SVD
compute
scores
θ

PIPELINE



Why this meta-algorithm works on trained model?

PIPELINE



Why this meta-algorithm works on trained model?

- Because an iteration of training (commonly gradient descent) is much cheaper than a run of SEVER

INTUITIONS

An outlier's gradient should be:

1. Large in magnitude
2. Systematically pointing in a specific direction

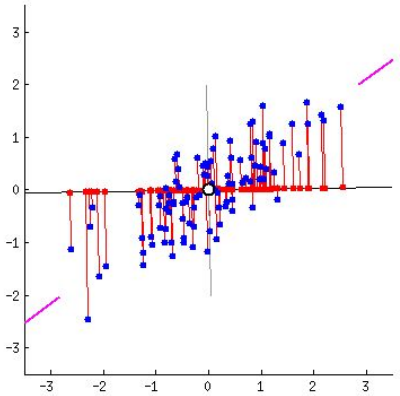
RECAP: SINGULAR VALUE DECOMPOSITION

$$A = U\Sigma V^\top$$

The columns of V are called the right singular vectors.

They are also the principle component of A.

The top right singular vector is the direction of A that preserves the most variance.



Algorithm 1 SEVER$(f_{1:n}, \mathcal{L}, \sigma)$

Input: Sample functions $f_1, \ldots, f_n : \mathcal{H} \to \mathbb{R}$, bounded below on a closed domain $\mathcal{H}$, γ-approximate learner $\mathcal{L}$, and parameter $\sigma \in \mathbb{R}_+$
Initialize $S \leftarrow \{1, \ldots, n\}$.
repeat
  $w \leftarrow \mathcal{L}(\{f_i\}_{i \in S})$. ▷ Run approximate learner on points in S.
  Let $\widehat{\nabla} = \frac{1}{|S|} \sum_{i \in S} \nabla f_i(w)$.
  Let $G = [\nabla f_i(w) - \widehat{\nabla}]_{i \in S}$ be the $|S| \times d$ matrix of centered gradients.
  Let v be the top right singular vector of G.
  Compute the vector τ of *outlier scores* defined via $\tau_i = \left((\nabla f_i(w) - \widehat{\nabla}) \cdot v\right)^2$.
  $S' \leftarrow S$
  $S \leftarrow$ FILTER(S', τ, σ) ▷ Remove some i's with the largest scores τ_i from S; see Algorithm 2.
until $S = S'$.
Return w.

average gradients

Matrix of centered (normalized) gradients

Project data into the direction and square the magnitude

The principle axis of the outlier has larger variance (pointing to a specific direction) and magnitude than the in-distribution data

HOW TO FILTER THE POINTS

- The learned function cannot be changed much by changing on an input while staying in domain.
- An in-distribution input could only move the function at any direction with an upper bound.
- The problem decreases to only keep the points with small singular value.
- In practise, just remove a few points with the highest scores for some iterations.

How to Determine the Fraction Size and Number of Iterations?

How to Determine the Fraction Size and Number of Iterations?

- They are hyper-parameters which are determined by tuning.
- The threshold is provably existent. (See Appendix for proof)

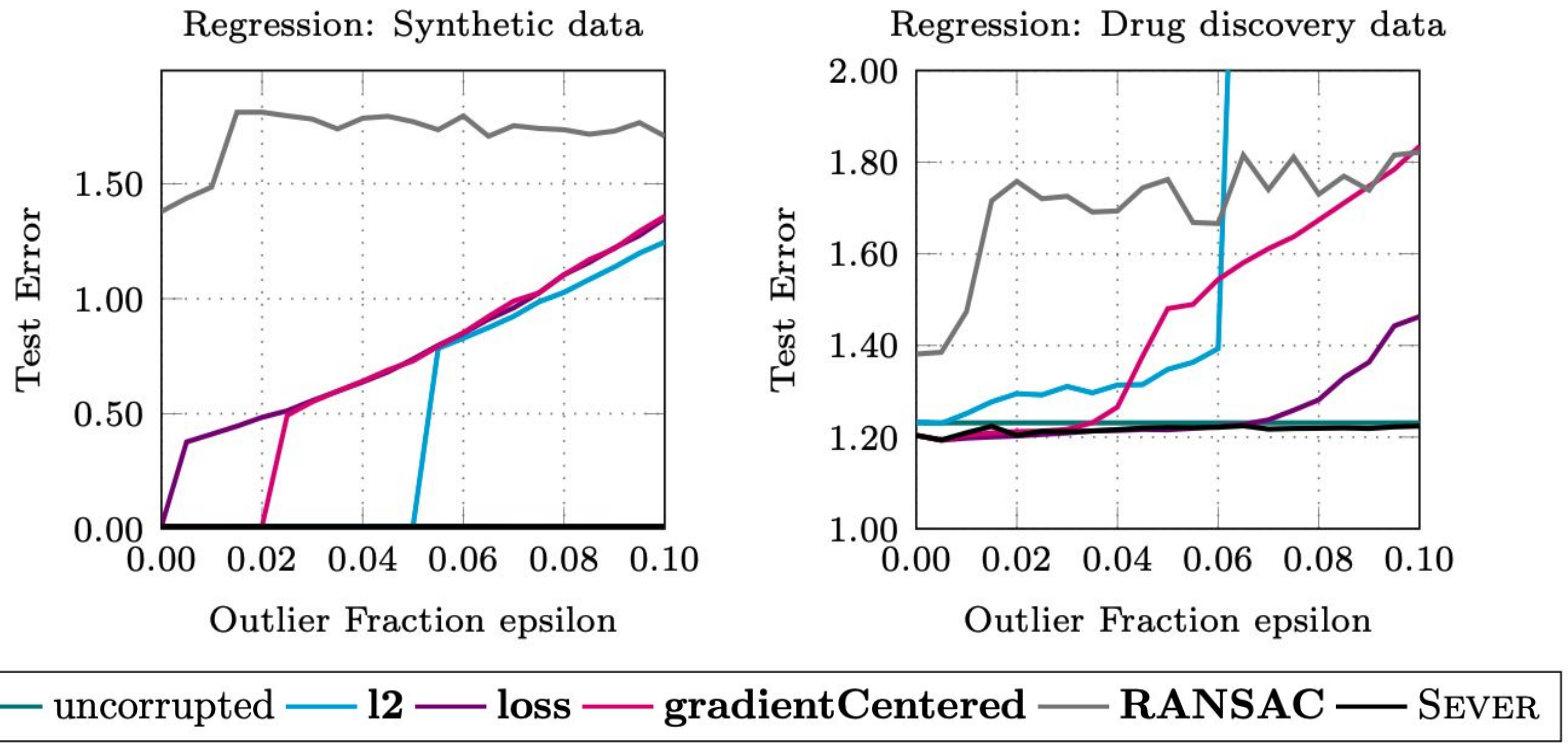
EXPERIMENT RESULTS

Experiments are conducted on two tasks:

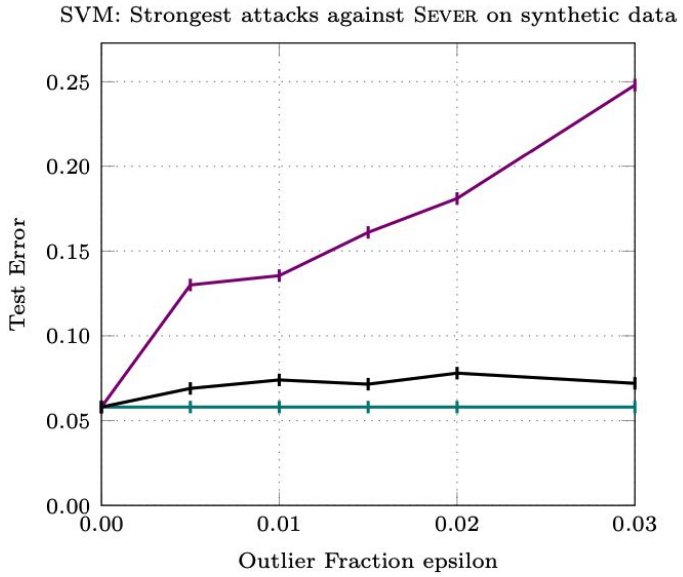
- Ridge Regression
  - Synthetic Gaussian Dataset: 500 dimensions
  - Drug Discovery Dataset: 410 dimensions

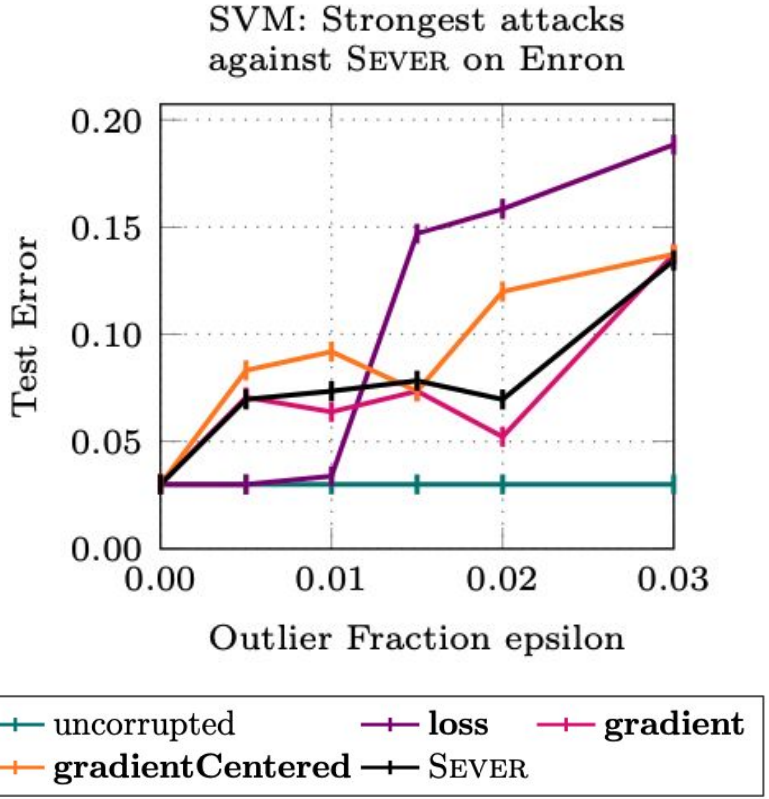
- Support Vector Machine
  - Synthetic Gaussian Dataset: 500 dimensions
  - Spam classification task: 5116 dimensions

RIDGE REGRESSION



SVM


SVM: Strongest attacks against SEVER on synthetic data
Test Error
0.25
0.20
0.15
0.10
0.05
0.00
0.00
0.01
0.02
0.03
Outlier Fraction epsilon
uncorrupted
loss
SEVER


SVM: Strongest attacks against SEVER on Enron
Test Error
0.20
0.15
0.10
0.05
0.00
0.00
0.01
0.02
0.03
Outlier Fraction epsilon
uncorrupted
loss
gradient
gradientCentered
SEVER

WHY BASELINES FAIL

- Baselines remove in-distribution data
- SEVER's score for outlier is clearly within the tail

LIMITATION

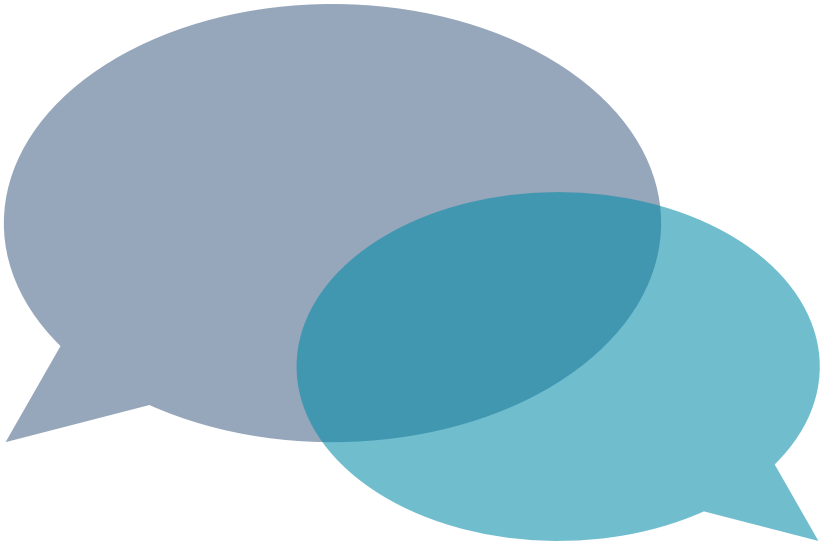
1. The paper only conducted experiments on convex optimization problems.
   - SEVER should show more results on modern ML models such as deep neural network (DNN).
2. The open-source code is implemented in matlab, which is not the most popular language in ML.
   - Should incorporate with more frequently used language like Python and R
3. Outlier design is limited.
   - The experiments use $X_{\text{bad}} = \frac{1}{\alpha \cdot n_{\text{bad}}} y^\top X \quad \text{and} \quad y_{\text{bad}} = -\beta$ to generate outlier, but the state-of-art is more advanced than that.
   - Gradient-based poisoning attack might break the defense.
4. Retraining is expensive, especially for the modern ML frameworks like DNN.
   - Should be integrate in the training process instead of the final model.

(POSSIBLE) FUTURE WORK

1. The attack against this meta algorithm.
   - Design an adversarial outlier that is not large in certain direction.
   - Evade the detection!

2. The attack based on this meta algorithm
   - Design detectable outliers.
   - Take advantage of the nature of retraining, let the loop never stop.
   - Sponge the computational power!

DISCUSSION



PAPER 1-3
SUMMARY DISCUSSION