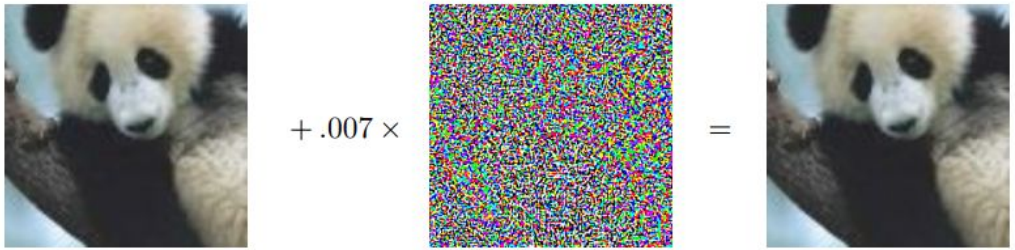
Safety: Poisoning

Jingyi Cui, Jade Gregoire, Srikar Mutnuri
Cheryl Bai & Ganesh Nanduru

University of Virginia
ENGINEERING
Department of Computer Science

Introduction

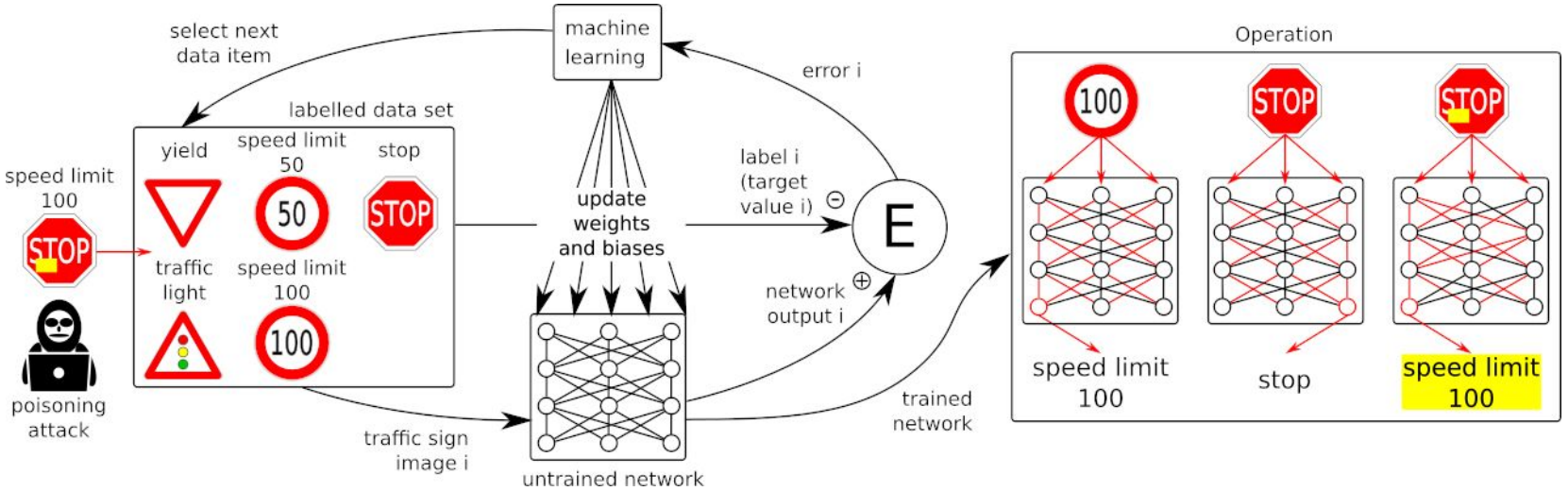
- Machine Learning (ML) powers critical applications, including security
  - Spam/Malware/fraud detection, healthcare
- ML learns from data, often from external sources
  - Data is assumed to be benign
- But the data can be vulnerable to manipulations from attackers
  - Can influence ML model behavior!



Irritating, but probably harmless

Panda + noise = Gibbon!

https://medium.com/onfido-tech/adversarial-attacks-and-defences-for-convolutional-neural-networks-66915ece52e7

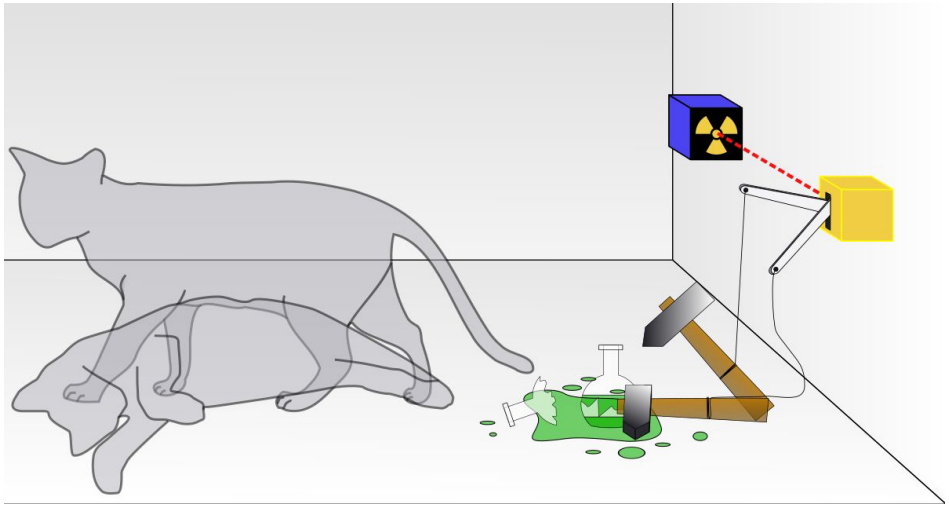


STOP sign + perturbations = Speed Limit 100

Definitely NOT harmless!

https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Automotive/Automatisiertes_Fahren/Automatisiertes_Fahren_node.html

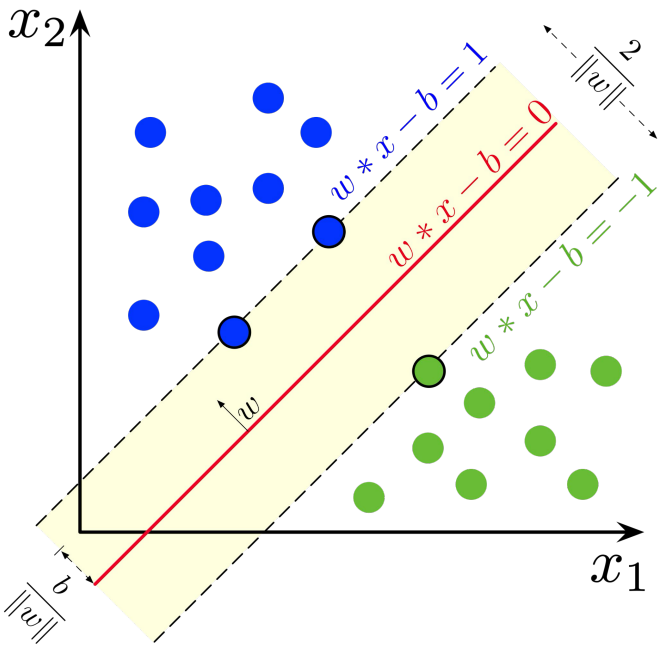
Poisoning 101



Data poisoning involves intentionally manipulating training data to impact the accuracy of a ML model

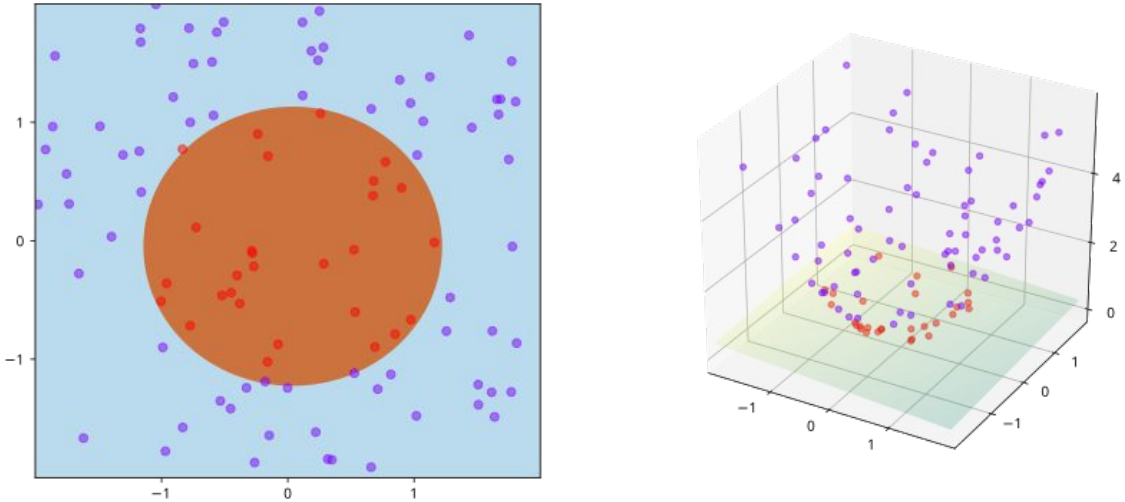
Support Vector Machines

SVMs attempt to find a hyperplane that, to the best degree possible, separates data points of one class from those of another class.



w: normal vector to the hyperplane.

SVMs - Kernel Function



The training points are mapped to a 3-dimensional space where a separating hyperplane can be easily found (kernel trick).

SVMs - Poisoning

Goal: Find an attack point (x_c, y_c) whose addition to the training data maximally decreases the SVM's classification accuracy

$$\max_{x_c} L(x_c) = \sum_{k=1}^{m} (1 - y_k f_{x_c}(x_k))_+ = \sum_{k=1}^{m} (-g_k)_+$$

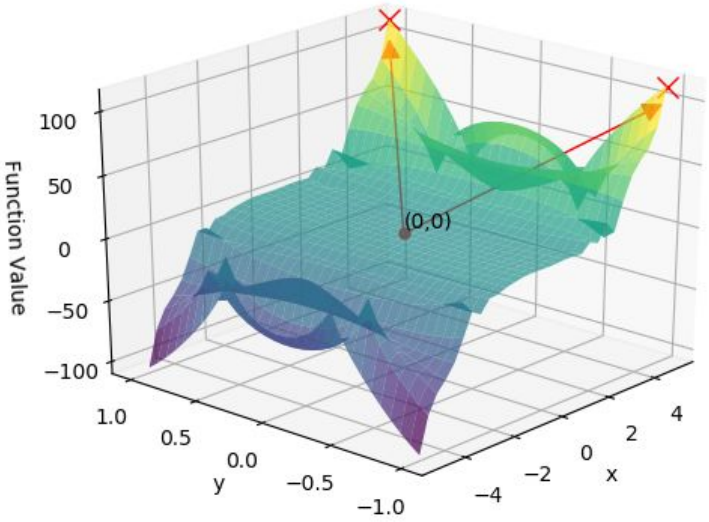
Maximize the Hinge Loss

Biggio et al. 2013

SVMs - Poisoning

We can use gradient ascent to iteratively optimize the objective function

Relies on the fact that infinitesimal change in x_c causes a smooth change in the optimal SVM solution (through an adiabatic update). This can be kernelized!

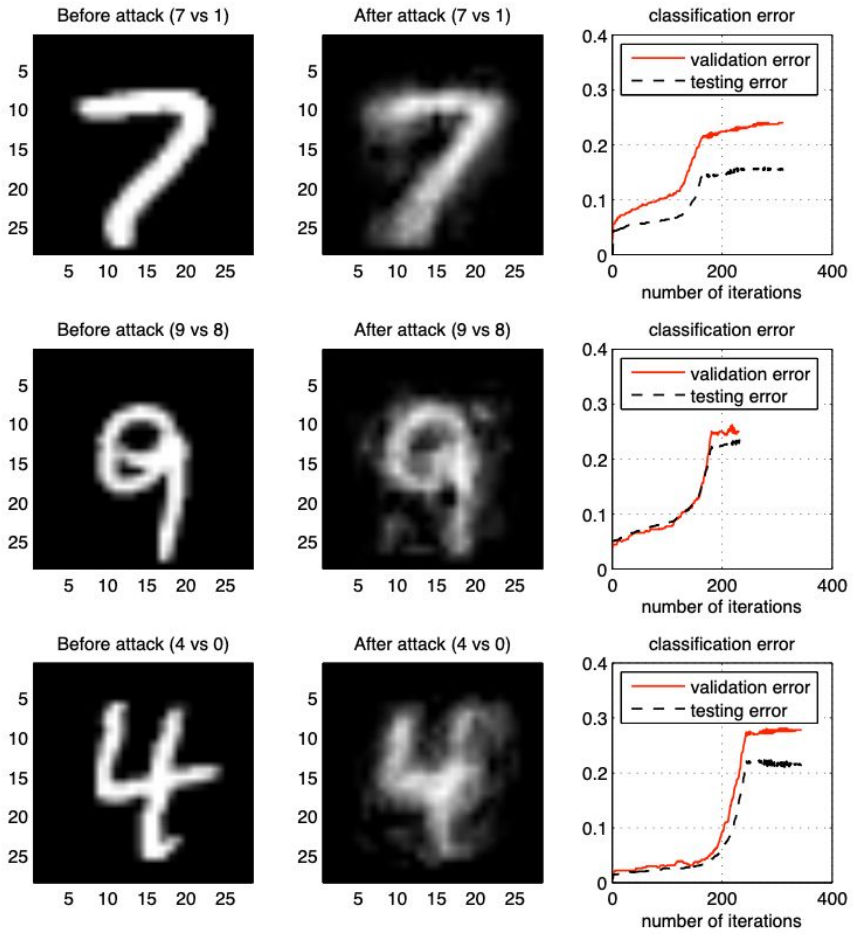


Gradient Ascent

https://vasugupta2000.medium.com/implementation-of-gradient-ascent-using-logistic-regression-7f5343877c21

SVMs - Poisoning MNIST Data

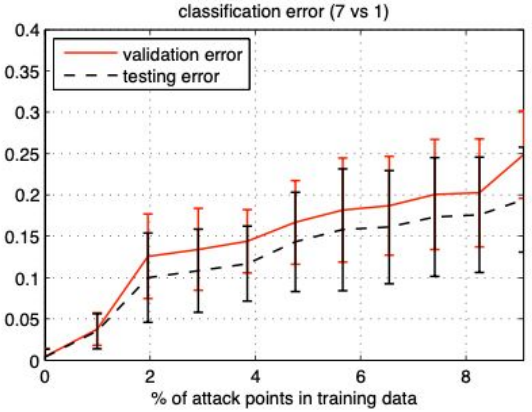
Modifications to the initial (mislabeled) attack point performed by the proposed attack strategy, for the three considered two-class problems from the MNIST data set. The increase in validation and testing errors across different iterations is also reported.

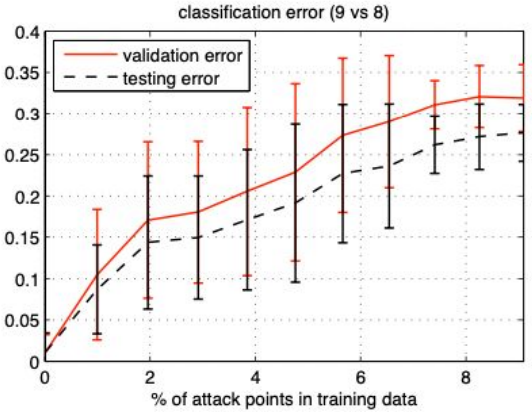


 Biggio et al. 2013

SVMs - Multi-point Poisoning

Results of the multi-point, multi-run experiments on the MNIST data set. In each plot, we show the classification errors due to poisoning as a function of the percentage of training contamination for both the validation (red solid line) and testing sets (black dashed line).





Biggio et al. 2013

Regression - Poisoning

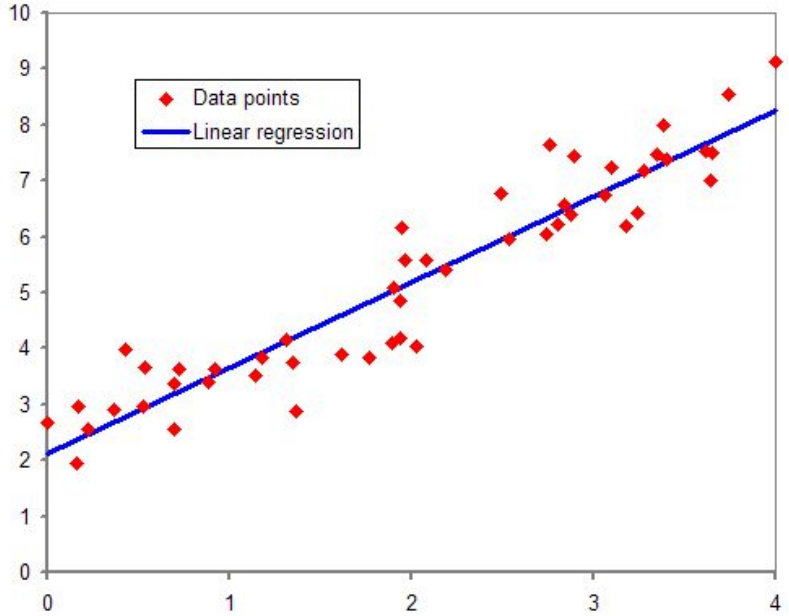
Estimate the relationship between a dependent variable and one or more independent variables

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{w}^\top \boldsymbol{x} + b$$

Linear Regression

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

(Ω(w): OLS, Ridge, LASSO, Elastic-net)



https://en.wikipedia.org/wiki/Regression_analysis

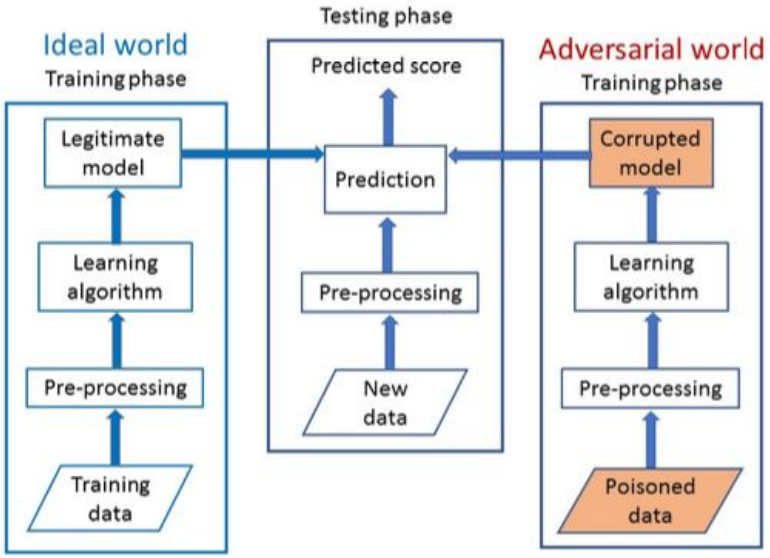
Jagielski et al. 2021

Regression - Methods

Aim to craft training data that maximally increases the test error of a linear regression model.

Paper introduces two attacks:

- OptP (Optimization based)
- StatP (Statistical based)



Jagielski et al. 2021

Regression - Methods: OptP

Optimal choice of several strategic components of the optimization framework. Components include:

- Initialization Strategy: how initial poisoning points are created after picking a set of random points from the training set
  - Inverse Flipping (InvFlip - y = 1- y)
  - Boundary Flipping (BFlip - y = round(1-y))
- Optimization Variable: which params of poisoning point are optimized during attack
- Optimization Objective: function that attacker tries to maximize by gradient ascent
  - Loss on training data
  - Loss on separate validation dataset

Jagielski et al. 2021

Regression - Methods: OptP

Attack proceeds iteratively, updating points with gradient ascent

Can be framed as a bilevel optimization problem:

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

Selecting poisoning points to maximize a loss function on an unpolluted dataset

Retraining the regression algo on a poisoned training set

Jagielski et al. 2021

Regression - Methods: StatP

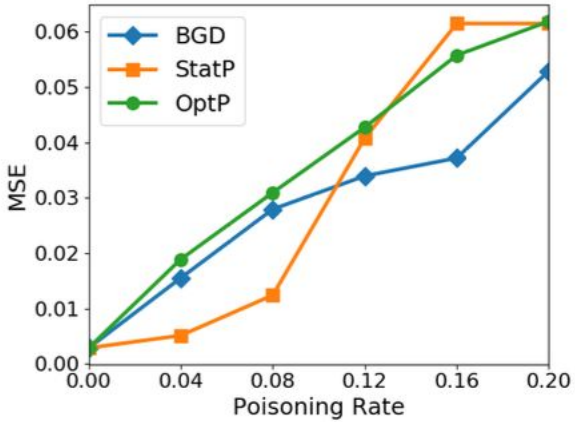
Fast, statistical attack that requires minimal knowledge about the targeted regression model. Leverages statistical properties of the training data to generate effective poisoning points:

- Sampling from training data distribution
- Rounding the feature values of sampled points
- Response variable selection set at the boundary to maximize the loss

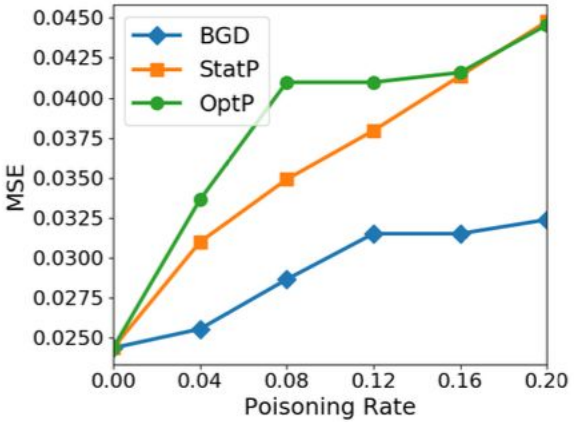
Unlike OptP, this does not involve an iterative optimization process, but instead relies on statistical data characteristics and heuristic choices for feature and response variable values.

Jagielski et al. 2021

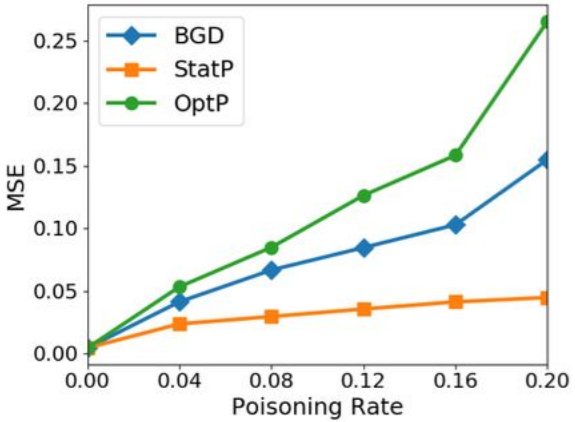
Regression - Poisoning Results



(a) Health Care Dataset



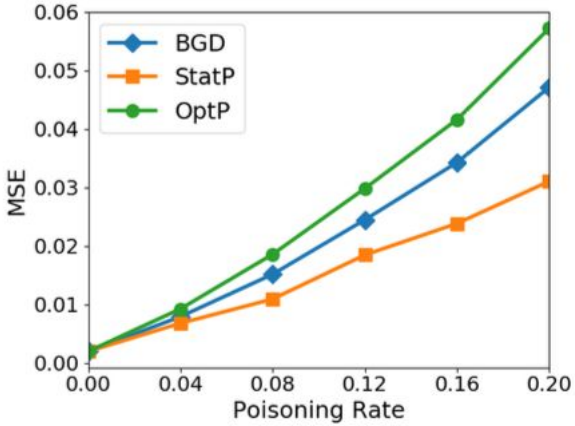
(b) Loan Dataset



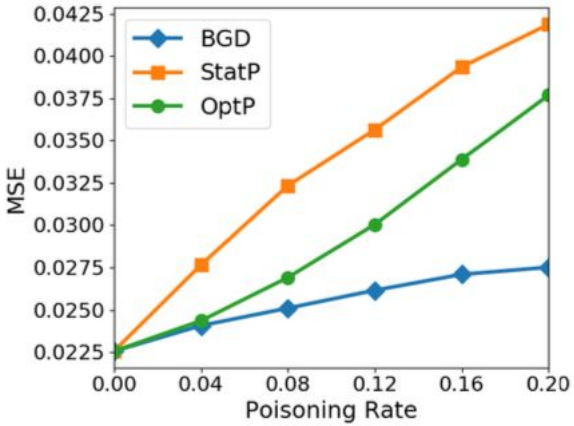
(c) House Price Dataset

MSE of attacks on ridge regression on the three datasets. Our new optimization (OptP) and statistical (StatP) attacks are more effective than the baseline gradient descent (BGD).

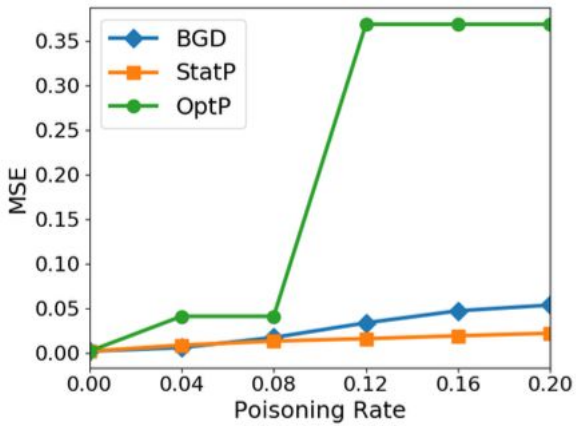
Regression - Poisoning Results



(a) Health Care Dataset



(b) Loan Dataset



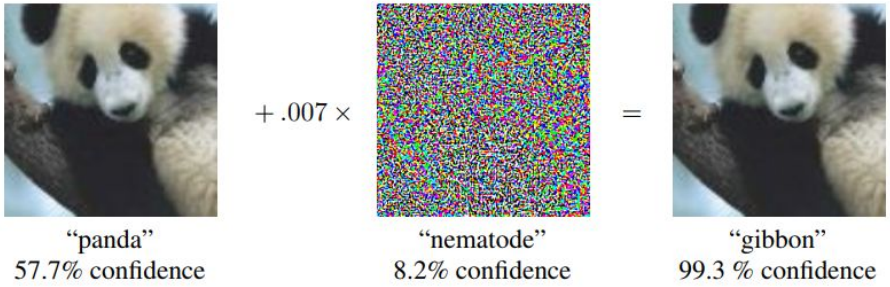
(c) House Price Dataset

MSE of attacks on LASSO on the three datasets. As for ridge, we find that StatP and OptP are able to poison the dataset very effectively, outperforming the baseline (BGD).

Jagielski et al. 2021

Neural Networks - Attacks

- Evasion attacks
  - Happen at test time
  - A target instance is modified to avoid detection by a classifier or be misclassified
  - Adversarial examples
- Data poisoning attacks
  - Happen at training time
  - Manipulate the performance of a system by inserting poison instances into the training data
  - Targeted clean-label poisoning attacks



Adversarial Example

https://medium.com/onfido-tech/adversarial-attacks-and-defences-for-convolutional-neural-networks-66915ece52e7

Shafahi et al. 2018

DNNs - Poisoning Attacks

- DNNs have been shown to fail catastrophically against data poisoning attacks
  - Test accuracy dropped 11% when attacker was allowed to modify 3% of a training set
  - Targeted backdoor attacks with few resources caused classifiers to fail for special test examples
  - Researchers trained a network using mislabeled images tagged with a special pattern
  - Success required poisons to fill up at least 12.5% of every training minibatch
- The problem with these?
  - Require test-time instances to be modified
  - Assume some degree of control over the labeling process in training
  - Unrealistic


Shafahi et al. 2018

DNNs - Targeted Clean-Label Poisoning Attacks

- Targeted attack:
  - Can control the behavior of the classifier on a specific test instance
  - Do not degrade overall classifier performance
- Clean-label attack: Do not require the attacker to have any control over the labeling of training data
- Why are these attacks significant?
  - Allows one to poison training sets easily
  - Do not need inside access to the data collection/labeling process
  - Makes attacks difficult to detect

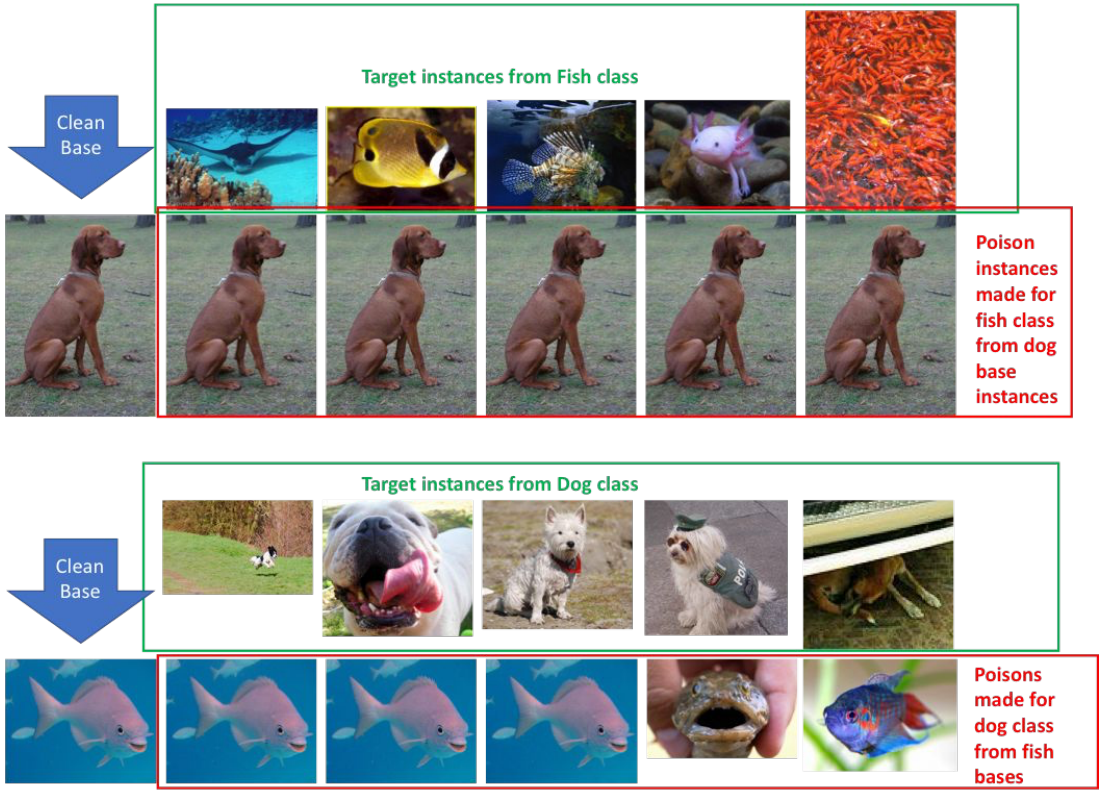

Shafahi et al. 2018

DNNs - Clean-Label Attack

- Target Instance:
  - An item in the test set (NOT the training set)
  - Aim to cause misclassification of at test time
- Base Instance:
  - An item from a class different than that of the target instance, which we intend to have the target instance misclassified as
  - Make imperceptible changes to it to craft a poison instance
- Poison Instance:
  - Must be injected into the training data
  - Aims to fool the model into labelling the target instance with the base instance label at test time

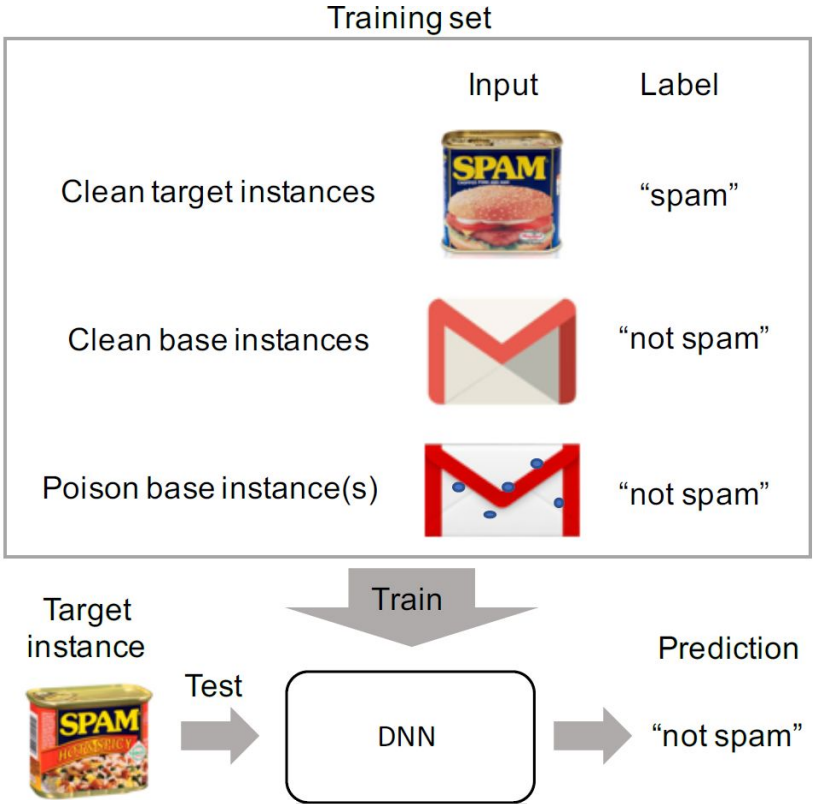

Shafahi et al. 2018

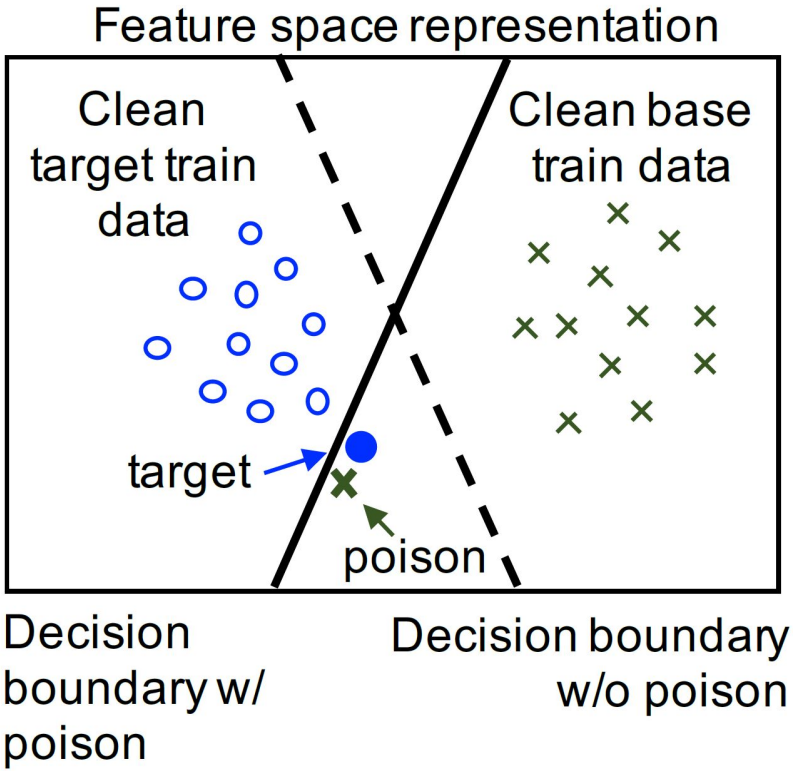
DNNs - Clean Label Attack Example



Shafahi et al. 2018

DNNs - How Clean Label Attacks Work





Shafahi et al. 2018

DNNs - How Clean Label Attacks Work

The degree to which a poison instance appears like a base class instance

Poison instance

$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} \; \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$

Feature spaces

Target

Input

Base instance

Algorithm 1 Poisoning Example Generation

Input: target instance t, base instance b, learning rate λ
Initialize x: $x_0 \leftarrow b$
Define: $L_p(x) = \|f(\mathbf{x}) - f(\mathbf{t})\|^2$
for $i = 1$ **to** $maxIters$ **do**
  Forward step: $\widehat{x}_i = x_{i-1} - \lambda \nabla_x L_p(x_{i-1})$
  Backward step: $x_i = (\widehat{x}_i + \lambda\beta b)/(1 + \beta\lambda)$
end for

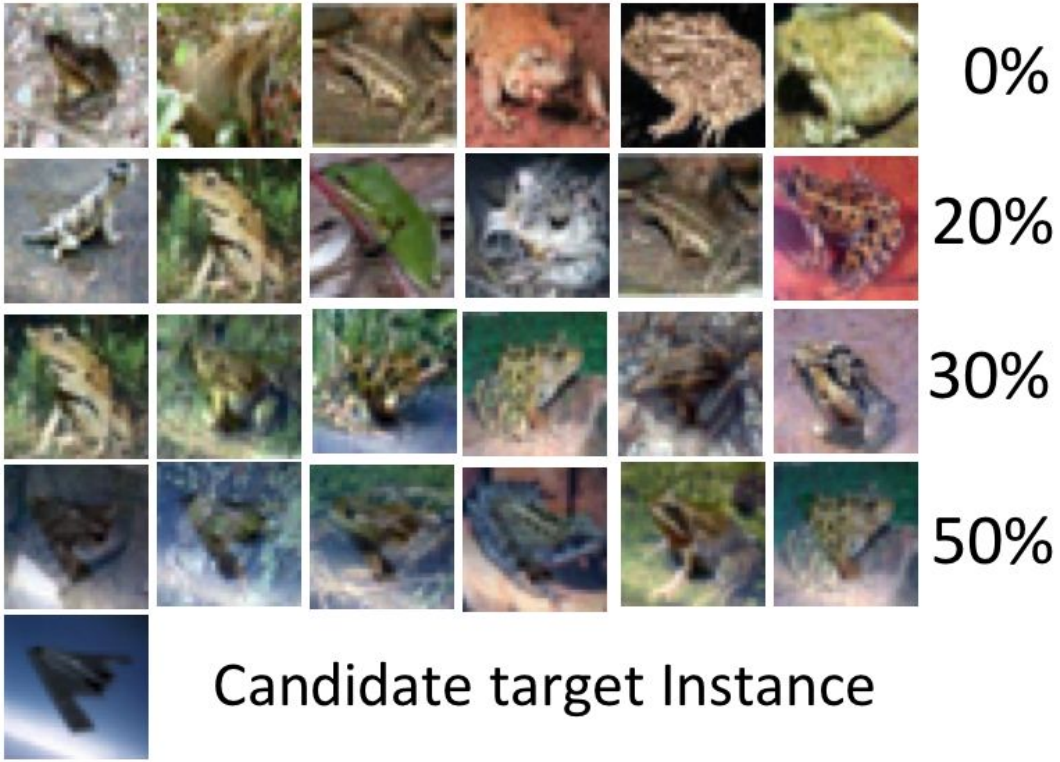
Shafahi et al. 2018

DNNs - Poisoning Attacks: Transfer Learning

- Attack a pretrained InceptionV3
- “Dog” vs “Fish” class
- 1099 test instances (698 from the “dog” class, 401 from the “fish” class)
- Results
  - Successful attack rate of 100% (Note: there were more trainable weights (2048) than training examples (1801))
  - Had a high median misclassification confidence of 99.6%
  - Overall test accuracy only dropped by an average of 0.2% (worst-case of 0.4%) from 99.5%
  - Repeated experiment with a third class, “cat”, which also achieved 100% poisoning success while maintaining a test accuracy of 96.4% (generalizes to non-binary classification)

Shafahi et al. 2018

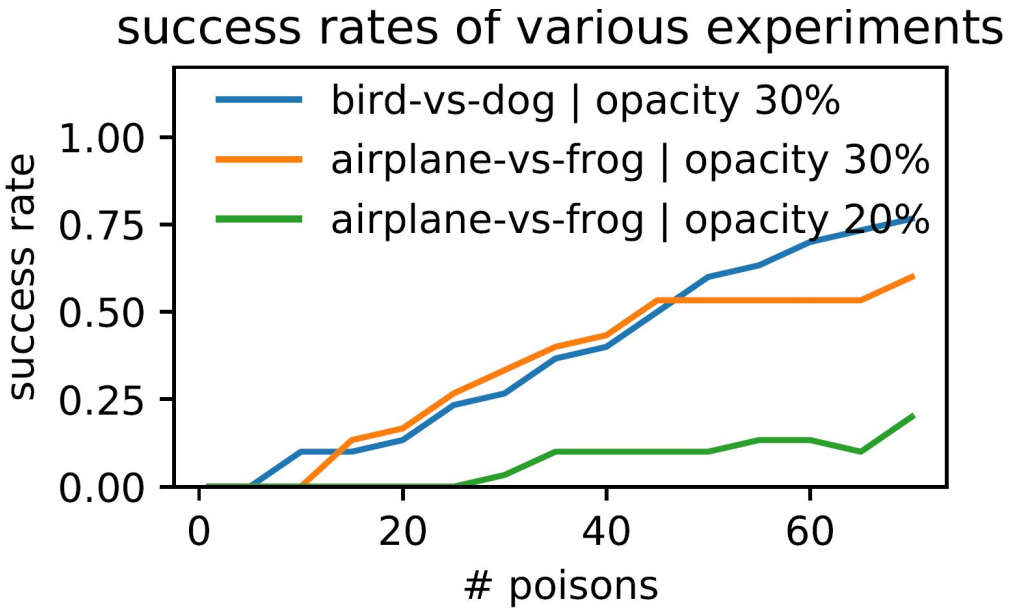
DNNs - Watermarking



Shafahi et al. 2018

DNNs - Poisoning Attacks: End-to-End Training

- Poison attacks become more difficult when all layers are trainable
- More poisons, higher opacity ⟶ Attacks are more successful



Shafahi et al. 2018

DNNs - Successful vs Unsuccessful Attacks

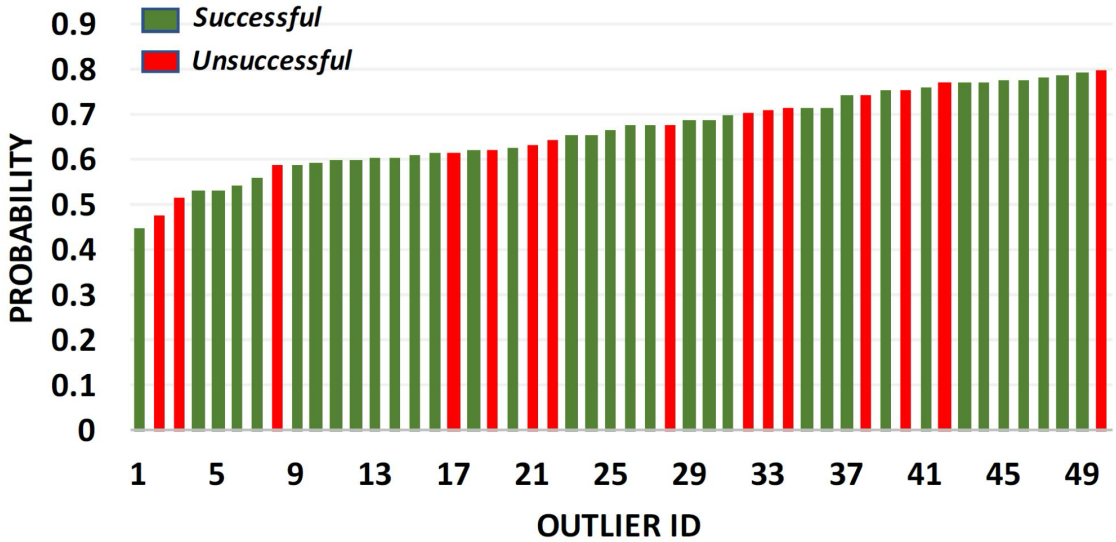
Shafahi et al. 2018

DNNs - Successful vs Unsuccessful Attacks

Shafahi et al. 2018

DNNs - Targeting Outliers

- Outliers: target class instances with lowest classification confidences
- Increases attack success (tested on airplane-vs-frog)
  - 50 poisons
  - 30% opacity
  - 70% success rate

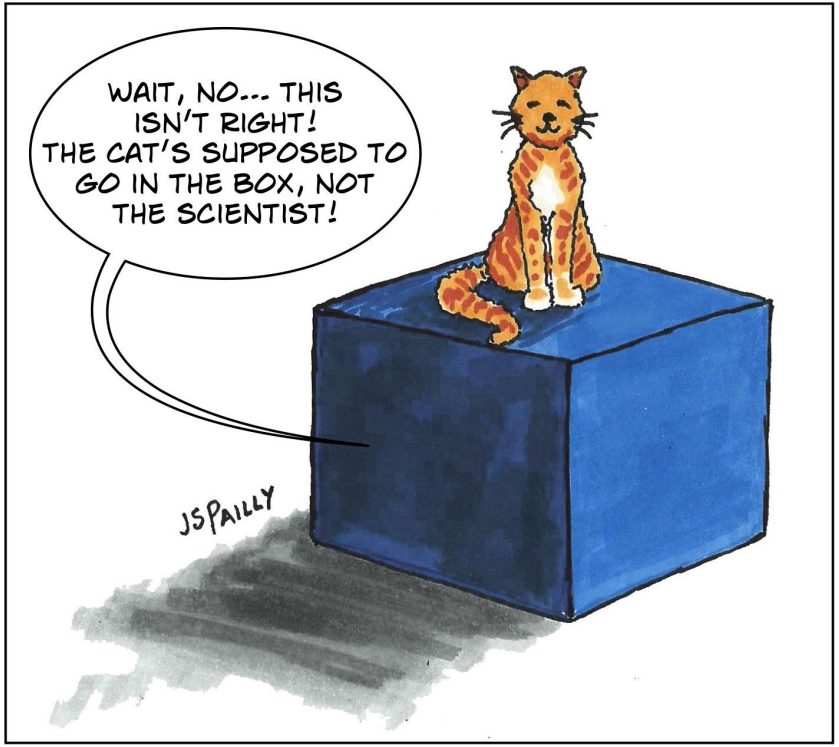


Shafahi et al. 2018

Discussion - Poisoning Attacks

- These works cover poisoning attacks from 2012-2018 on different types of machine learning models. How might poisoning attacks compromise large language models which are growing in popularity today?
    - How does that impact the fairness, trustworthiness, and reliability of generative AI systems?
- Will formalizing poisoning attacks make it easier to implement stronger attacks?
    - Should we be worried about this?

Defenses (Against the Dark Arts) 101



Fundamental Questions

- How much can an adversary degrade a model's accuracy?
- Can we compute an upper bound on test loss under attack?
- How effective are different defense mechanisms?

Approach: Formalizing the worst-case attack and evaluating defenses rigorously.

Poisoning Attack Model: Attacker vs. Defender

The Attack Setup:

- The attacker modifies the training data by injecting **εn** poisoned samples Dp.
- The defender trains a classifier on both clean and poisoned data

Attacker's Goal:

- Increase test loss L(θ) to degrade the model's performance

The test loss is defined as: $\mathbf{L}(\theta) = \mathbf{E}_{(x,y)\sim p^*}[\ell(\theta; x, y)]$

Here, ℓ(θ;x,y) is the classification loss (e.g., hinge loss for SVMs)

Steinhardt et al. 2017

Worst-Case Attack: Upper Bound on Test Loss

- The attacker wants to maximize the test loss by injecting poisoned samples Dp.
- The defender (learner) tries to minimize the test loss by adjusting θ.

$$\begin{aligned}\max_{\mathcal{D}_\mathrm{p}} \mathbf{L}(\hat{\theta}) &\overset{\text{(i)}}{\approx} \max_{\mathcal{D}_\mathrm{p}} \frac{1}{n} L(\hat{\theta}; \mathcal{D}_\mathrm{c}) \overset{\text{(ii)}}{\leq} \max_{\mathcal{D}_\mathrm{p}} \frac{1}{n} L(\hat{\theta}; \mathcal{D}_\mathrm{c} \cup (\mathcal{D}_\mathrm{p} \cap \mathcal{F})) \\ &\overset{\text{(iii)}}{\approx} \max_{\mathcal{D}_\mathrm{p}} \frac{1}{n} L(\tilde{\theta}; \mathcal{D}_\mathrm{c} \cup (\mathcal{D}_\mathrm{p} \cap \mathcal{F})) \\ &= \max_{\mathcal{D}_\mathrm{p} \subseteq \mathcal{F}} \min_{\theta \in \Theta} \frac{1}{n} L(\theta; \mathcal{D}_\mathrm{c} \cup \mathcal{D}_\mathrm{p}) \overset{\text{def}}{=} \mathbf{M}. \end{aligned} \tag{4}$$

Steinhardt et al. 2017

Online Learning for Finding Attacks

- We established that the worst-case test loss is defined by a minimax problem
- However, solving the minimax problem directly is computationally infeasible
- Instead of brute force search, this algorithm uses online learning to efficiently find the most effective poisoned samples

Algorithm 1 Online learning algorithm for generating an upper bound and candidate attack.

Input: clean data $\mathcal{D}_c$ of size n, feasible set $\mathcal{F}$, radius ρ, poisoned fraction ϵ, step size η.
Initialize $z^{(0)} \leftarrow 0$, $\lambda^{(0)} \leftarrow \frac{1}{\eta}$, $\theta^{(0)} \leftarrow 0$, $U^* \leftarrow \infty$.
for $t = 1, \ldots, \epsilon n$ **do**
  Compute $(x^{(t)}, y^{(t)}) = \operatorname{argmax}_{(x,y)\in\mathcal{F}} \ell(\theta^{(t-1)}; x, y)$.
  $U^* \leftarrow \min\left(U^*, \frac{1}{n} L(\theta^{(t-1)}; \mathcal{D}_c) + \epsilon \ell(\theta^{(t-1)}; x^{(t)}, y^{(t)})\right)$.
  $g^{(t)} \leftarrow \frac{1}{n} \nabla L(\theta^{(t-1)}; \mathcal{D}_c) + \epsilon \nabla \ell(\theta^{(t-1)}; x^{(t)}, y^{(t)})$.
  Update: $z^{(t)} \leftarrow z^{(t-1)} - g^{(t)}$, $\lambda^{(t)} \leftarrow \max(\lambda^{(t-1)}, \frac{\|z^{(t)}\|_2}{\rho})$, $\theta^{(t)} \leftarrow \frac{z^{(t)}}{\lambda^{(t)}}$.
end for
Output: upper bound U^* and candidate attack $\mathcal{D}_p = \{(x^{(t)}, y^{(t)})\}_{t=1}^{\epsilon n}$.

Steinhardt et al. 2017

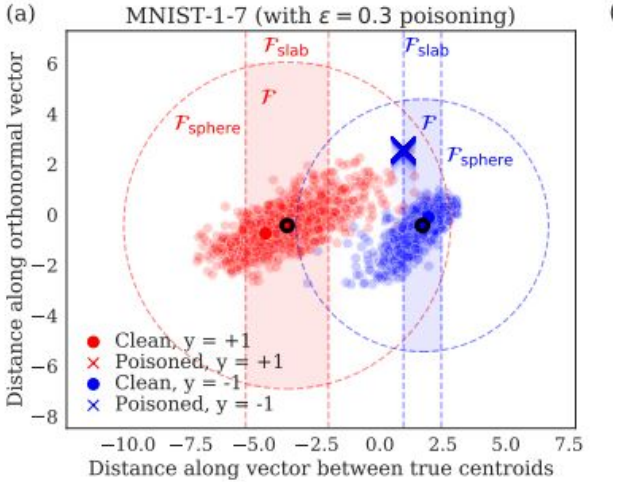
Certified Defenses

Sphere Defense: Removes extreme outliers in Euclidean space

$$F_{\text{sphere}} = \{(x, y) \mid \|x - \mu_y\|_2 \leq r_y\}$$

Slab Defense: Restricts data points that deviate too much along the decision boundary.

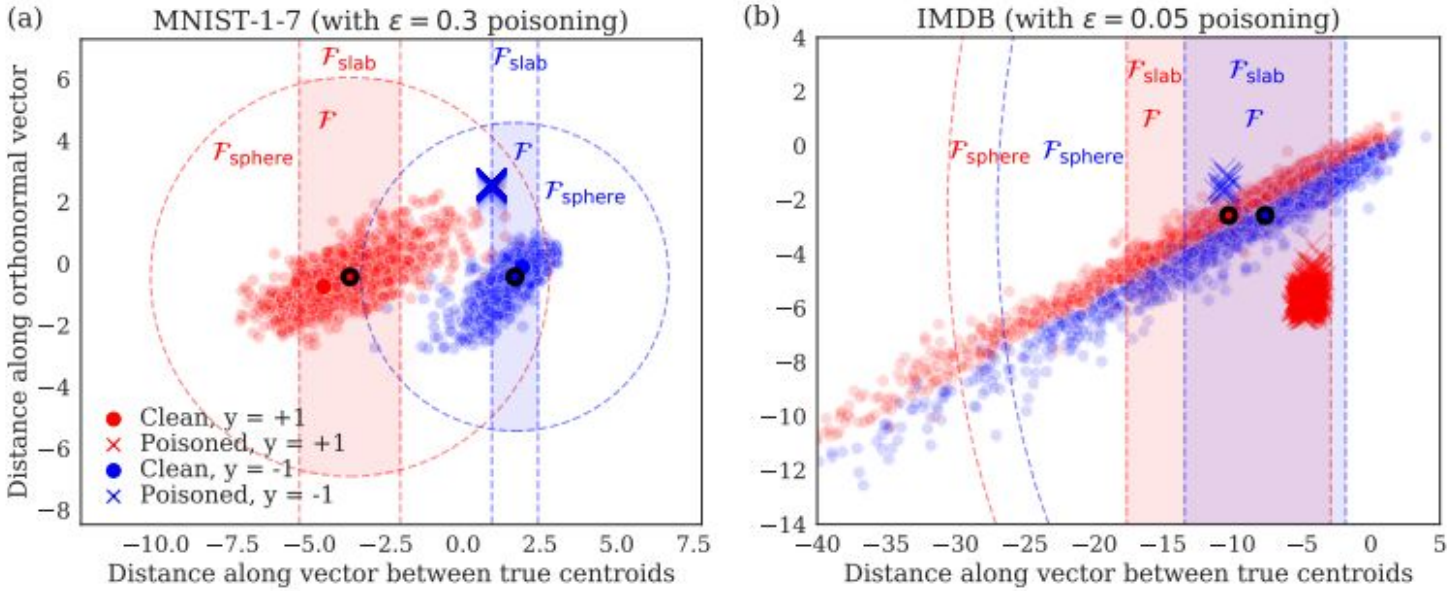
$$F_{\text{slab}} = \{(x, y) \mid |\langle x - \mu_y, \mu_y - \mu_{-y} \rangle| \leq s_y\}$$



Steinhardt et al. 2017

Impact on Decision Boundaries

- MNIST-1-7 (Left Plot): Classes are well-separated, defenses work well.
- IMDB (Right Plot): Class centroids overlap, defenses struggle.

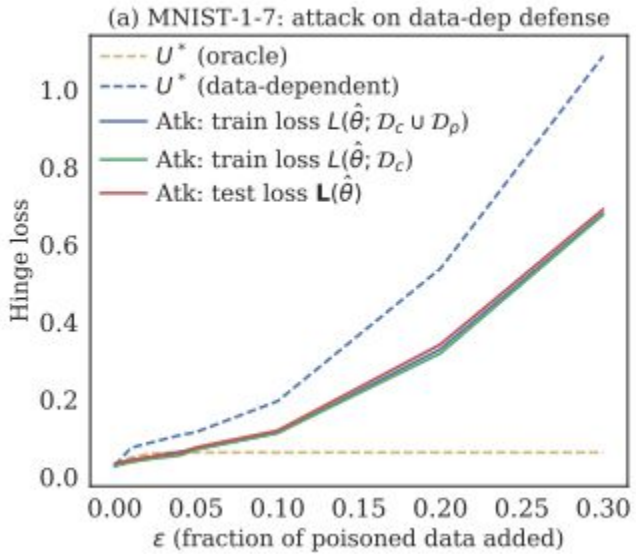


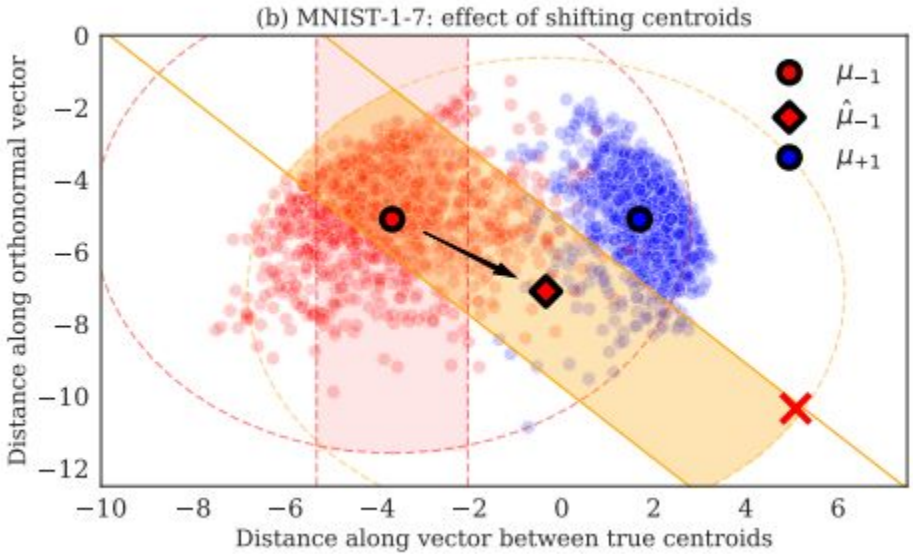
Steinhardt et al. 2017

Fixed (Oracle) vs. Data-Dependent Defenses

Two Types of Defenses:

- Oracle Defense: Uses true class means μy
- Data-Dependent Defense: Uses estimated means μ^y

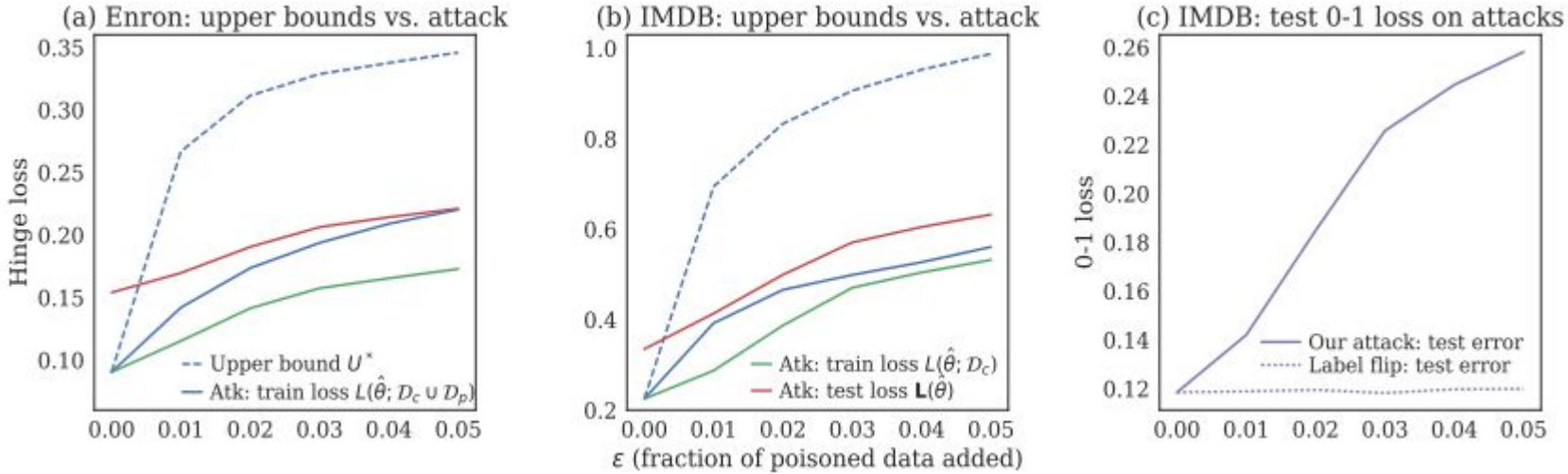




Steinhardt et al. 2017

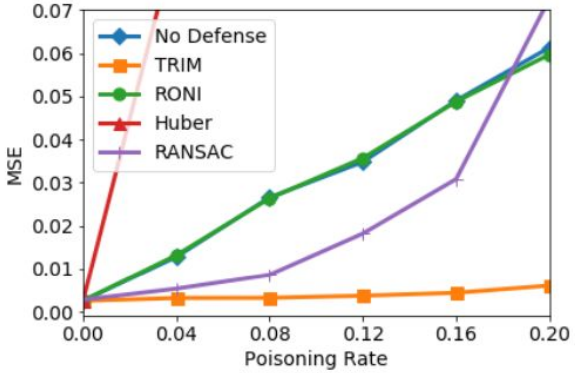
Experimental Results

- Even a small fraction of poisoned emails can significantly degrade spam filtering performance
- A well-optimized attack like ours can be much more damaging to NLP models

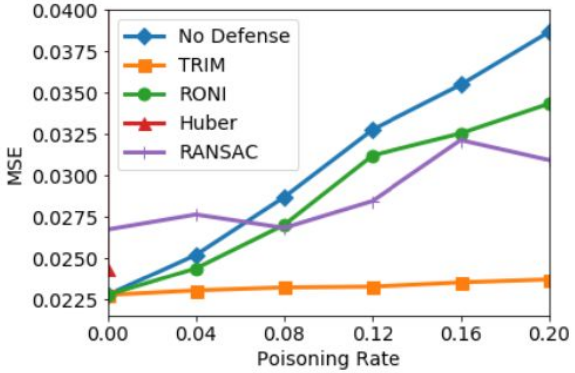


Steinhardt et al. 2017

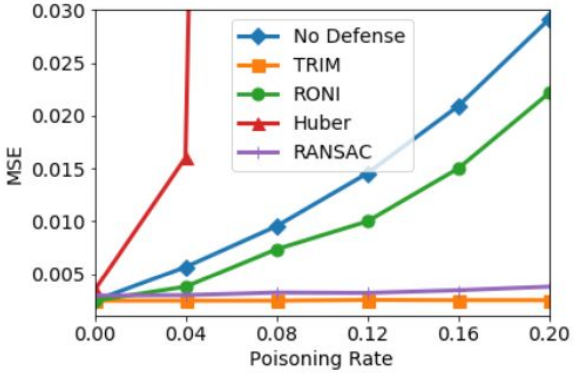
Regression - Countermeasures: TRIM



(a) Health Care Dataset



(b) Loan Dataset

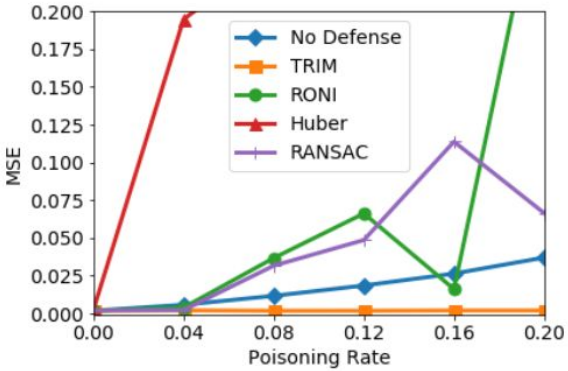


(c) House Price Dataset

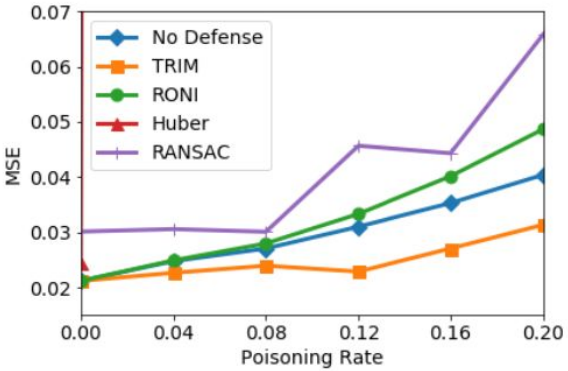
MSE of defenses on ridge on the three datasets. Defenses are evaluated against the OptP attack. The only defense that consistently performs well in these situations is the proposed TRIM defense, with RANSAC, Huber, and RONI actually performing worse than the undefended model in some cases.

Jagielski et al. 2021

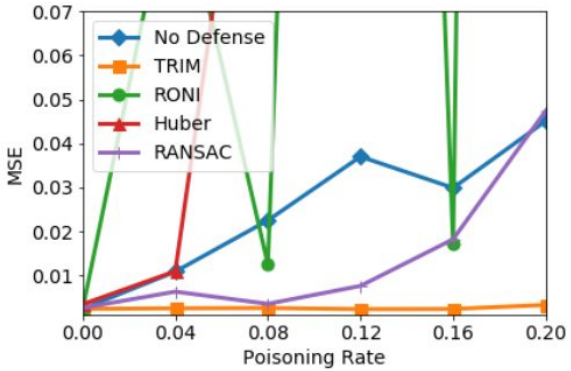
Regression - Countermeasures: TRIM



(a) Health Care Dataset



(b) Loan Dataset



(c) House Price Dataset

MSE of defenses on LASSO. Defenses are evaluated against the most effective attack OptP. TRIM performs consistently better

Jagielski et al. 2021

Conclusion

- These papers collectively underscore the vulnerabilities of ML models, including SVMs, linear regression models and neural networks to data poisoning attacks
- With some prior knowledge, bad actors can devise sophisticated attack strategies using gradient based optimization to craft malicious training data to achieve this
- While the papers by Jagielski et al., and by Steinhardt et al., outline some defenses against attacks, much scope remains in the field - more robust & efficient defense mechanisms are needed.

Discussion - Poisoning Defenses

- If you were developing a model to classify MNIST digits and learned a hacker might have manipulated some of your labels, how would you proceed?
    - Consider: what information about the original / altered dataset would you need to know?
- Should researchers be required to propose working defenses in any papers introducing new attacks?
- What are some challenges with implementing defenses against poisoning attacks in industry?

Discussion - Scenario

Suppose you are a security researcher and you've just discovered a theoretical vulnerability in LLM data privacy. Applied maliciously, this vulnerability can be developed into a poisoning attack compromising private data on millions of LLM users. Do you submit your idea for publication? Why or why not?

Consider:

- Should ML researchers be allowed to freely publish adversarial work with unclear implications in real life?
- What responsible disclosure requirements should be placed on a paper like this before it is allowed to be published?

Discussion - SVM Poisoning

- Suppose the US Postal Service was using an SVM to classify digits for sending mail to the address handwritten on the envelope
- Would the USPS system be susceptible to data poisoning?
  - How could attackers poison this model?
  - How would a poisoned SVM perform compared to a non-poisoned model?
  - What are some harmful results that data poisoning could bring out in this context?