Adversarial Machine Learning
Poisoning Attacks

Fereshteh Razmi
Fall 2018

Attack types

Based on time of the attack:

Evasion Attacks:

- Test time
- Model is already trained
- Goal: Misclassification at test time

Poisoning Attacks:

- Training time
- Model is training
- Goal: Misclassification at test time (dropping accuracy)

Scenarios

Online training:
Machine learning models are updated based on new incoming training data

Users may :

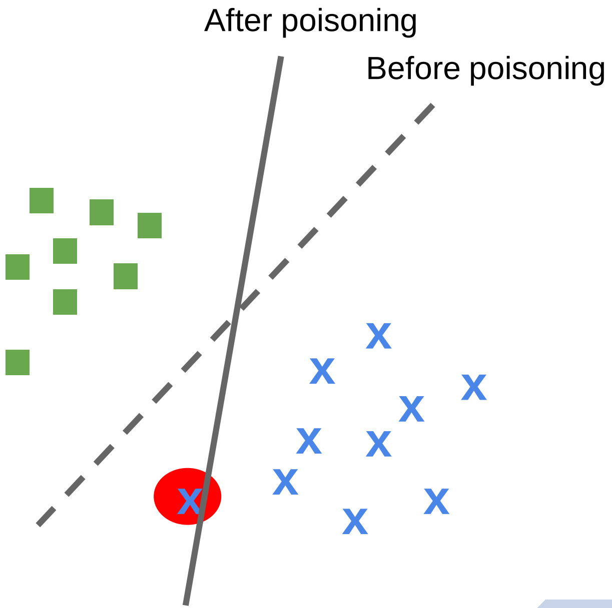
- Mark emails as spams / change spam emails to look normal
- Mark a file as malware (VirusTotal)
- Provide movie ratings / provide **untrue** comments (Recommendation systems)
- Tag photos **falsely** (Captcha)

Users Can change:

- Labels
- Feature space
- Both

How it works

1. Choose a few data points (random or from clean/normal training set)
2. Manipulate them in a way they would be very disruptive to the test-time accuracy
3. Add generated poisoning points to the training set
4. Train the model with new dataset
5. Wait for model accuracy to drop when you test with real clean data later



Poisoning Types

1. Flip labels (binary labels)

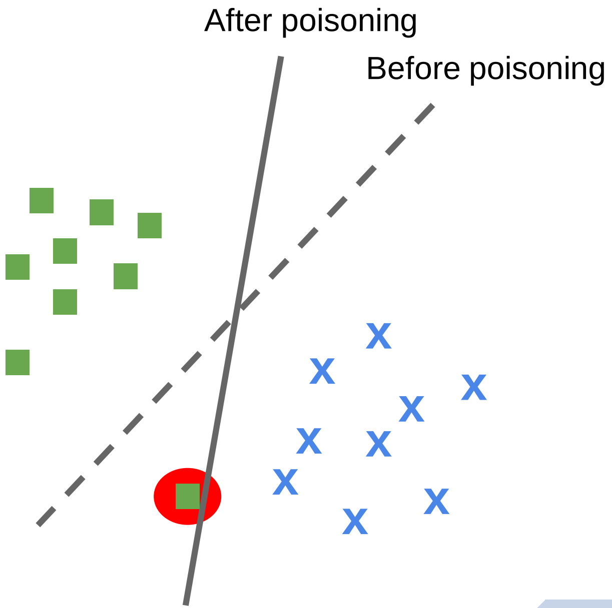
Randomly

Most influential samples

A. Representative samples (Inliers🤔)

- Statistical methods
- Common features in each class
- ...

B. Outliers



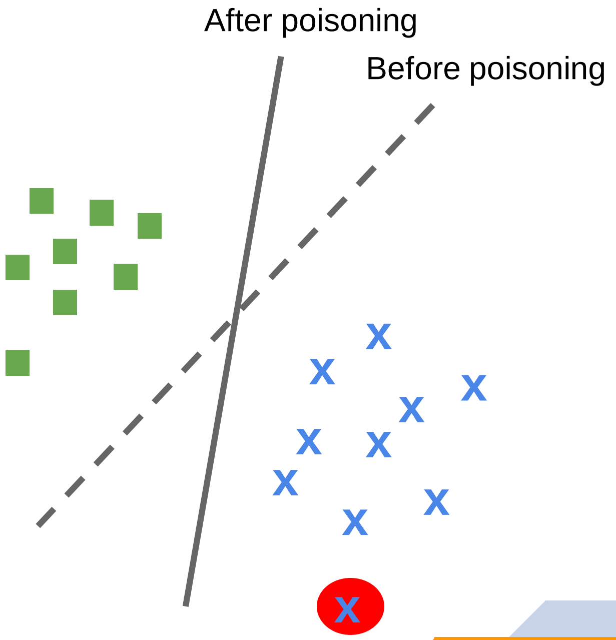
Poisoning Types

2. Bi-level optimization

Change features to get the worst accuracy

$$S_p^* \in \underset{S_p \subseteq \mathcal{Z}}{\operatorname{argmax}} \quad O_A = O_L(S_{val}, \mathbf{w}^*)$$

$$\text{s.t.} \quad \mathbf{w}^* \in \underset{\mathbf{w}}{\operatorname{argmin}}\, O_L(S_{tr} \cup S_{p'} \mathbf{w})$$



S_p: set of adversarial examples, S_{tr}: training set, S_{val}: validation set

O_L: Model Loss function, O_A: Adversary's objective/loss function

Poisoning Types

2. Bi-level optimization

Change features to get the worst accuracy

$$S_p^* \in \underset{S_p \subseteq \mathcal{Z}}{\arg\max} \quad O_A = O_L(S_{val}, \mathbf{w}^*)$$

$$\boxed{\text{s.t.} \quad \mathbf{w}^* \in \underset{\mathbf{w}}{\arg\min}\, O_L(S_{tr} \cup S_p, \mathbf{w})}$$

S_p: set of adversarial examples, S_{tr}: training set, S_{val}: validation set

O_L: Model Loss function, O_A: Adversary's objective/loss function

Inner equation: 😇

Feed the model with both poisoning and normal data for training(get the parameters that minimize Model's loss function)

Poisoning Types

2. Bi-level optimization

Change features to get the worst accuracy

$$\boxed{S_p^* \in \underset{S_p \subseteq \mathcal{Z}}{\arg\max} \quad O_A = O_L(S_{val}, \mathbf{w}^*)}$$

$$\text{s.t.} \quad \mathbf{w}^* \in \underset{\mathbf{w}}{\arg\min}\, O_L(S_{tr} \cup S_p, \mathbf{w})$$

S_p: set of adversarial examples, S_{tr}: training set, S_{val}: validation set

O_L: Model Loss function, OA: Adversary's objective/loss function

Inner equation:

Feed the model with both poisoning and normal data for training (get the parameters that minimize Model's loss function)

Outer equation:

Since we trained the model based on poisoning points, it should drop the accuracy in test time (maximize the loss function on normal validation set)

Poisoning Types

How to solve:

- Calculate the derivative of O_A w.r.t S_p
- But S_p is appeared only in the inner equation
- Chain rule (w* connects two equations)
- Need to solve inner equation
- Difficult for non-convex models (NN)
- Easy on Linear regression and SVMs

$$
\begin{aligned}
S_p^* \in \underset{S_p \subseteq \mathcal{Z}}{\operatorname{argmax}} \quad & O_A = O_L(S_{val}, \mathbf{w}^*) \\
\text{s.t.} \quad & \mathbf{w}^* \in \underset{\mathbf{w}}{\operatorname{argmin}}\, O_L(S_{tr} \cup S_p, \mathbf{w})
\end{aligned}
$$

Comparison to flipping attacks

- More powerful
- More complex
- Best result when they are used together

Settings

Goals:

- **Degrade the overall accuracy rate**
- Resulting model favors the attacker
- Plant a trapdoor

Strength:

- Add, remove, update
- Knowledge of the attacker on: (Threat Models)
    - model parameters and algorithm (White Box vs. Black Box)
    - training set

Settings

- Integrated, Collaborative (limited data)
- Targeted, untargeted

Attack: Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning

Regression problem → continuous response variable
Optimization in feature space and response variables

$$\nabla_{x_p} O_A = \nabla_w O_A * \nabla_{x_p} w(x_p)$$

$$\downarrow$$

$$\nabla_{z_p} O_A = \nabla_w O_A * \nabla_{z_p} w(z_p)$$

$$z_p = (x_p, y_p)$$

$$S_p^* \in \underset{S_p \subseteq \mathcal{Z}}{\arg\max} \quad O_A = O_L(S_{val}, \mathbf{w}^*)$$

$$\text{s.t.} \quad \mathbf{w}^* \in \underset{\mathbf{w}}{\arg\min}\, O_L(S_{tr} \cup S_p, \mathbf{w})$$

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., & Li, B. (2018). Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. *arXiv preprint arXiv:1804.00308*.

Attack: Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning (cont'd)

Parameters to tune

1. How to select initial points?
   - Flipping labels
   - i) InvFlip: $y = 1 - y$ [0.1]
   - ii) BFlip: $y = round(1-y)$ {0,1}
2. Use validation set or training set in outer loop
3. X_p or Z_p

Which combination gives us the best result (most successful attack)?

Attack: Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning (cont'd)

Parameters to tune

1. How to select initial points?
   Flipping labels
   i) InvFlip: y = 1 - y   [0.1]
   ii) BFlip: y = round(1-y)   {0,1}
2. Use validation set or training set in outer loop
3. X_p or Z_p

Which combination gives us the best result (most successful attack)?

Depends on the dataset…

Model	Dataset	Init	Argument	Objective
Ridge	Health	BFlip	(x, y)	$\mathcal{W}_{\mathrm{tr}}$
	Loan	BFlip	x	$\mathcal{W}_{\mathrm{val}}$
	House	BFlip	(x, y)	$\mathcal{W}_{\mathrm{tr}}$
LASSO	Health	BFlip	(x, y)	$\mathcal{W}_{\mathrm{tr}}$
	Loan	BFlip	(x, y)	$\mathcal{W}_{\mathrm{val}}$
	House	InvFlip	(x, y)	$\mathcal{W}_{\mathrm{val}}$

Defenses

1. Impact of each sample on the model (RONI)

- If it drops the accuracy → Retrain classifier
- Specify a threshold

Problem:

- Contribution of single points
- Coordinated set of adversarial points

Defenses

2. Detecting outliers

- Maximum impact on the learner
- Minimum injected adversary examples

- Detect directly

- Bagging methods (less weight)

…

Other works:

- Randomness: unpredictability
- Content provenance: related meta-data
- Formal models, Robustness!

Defense: Mitigating Poisoning Attacks on Machine Learning Models: A Data Provenance Based Approach

Assumption:

1. Each data point has a meta data (device identification, timestamps, ...)
2. Defender already has access to a subset of clean training data

Goal:

To remove poisoning data from the training dataset

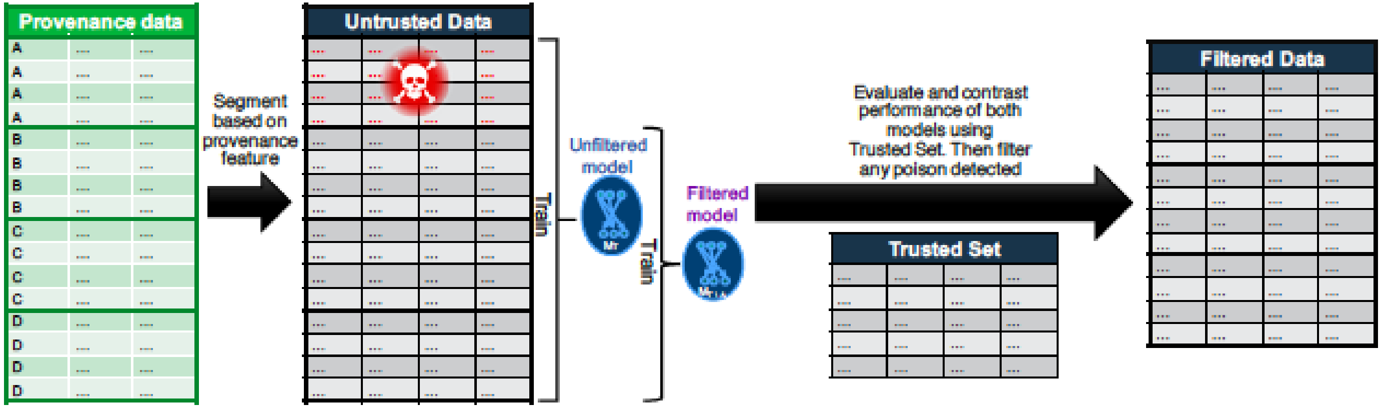
*Baracaldo, Nathalie, Bryant Chen, Heiko Ludwig, and Jaehoon Amir Safavi. "Mitigating Poisoning Attacks on Machine Learning Models: A Data Provenance Based Approach." In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 103-110. ACM, 2017.

Defense: Mitigating Poisoning Attacks on Machine Learning Models: A Data Provenance Based Approach (con't)

Train a model based on available clean training data

Segment all data based on their common metadata

Check if a data segment drops the accuracy



Defense: Mitigating Poisoning Attacks on Machine Learning Models: A Data Provenance Based Approach (con't)

Similar to RONI defense method

RONI tests each data point/record separately

Retraining with and without all the records is time-consuming

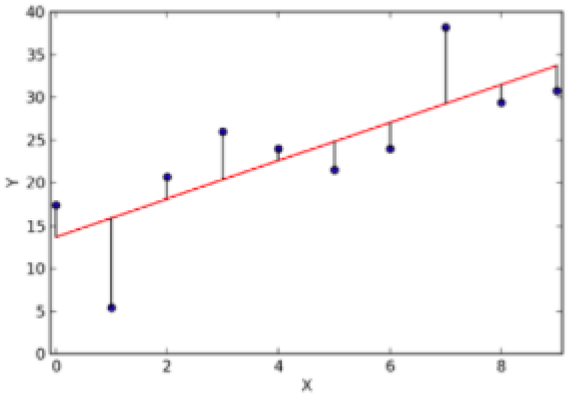
This method reduces number of retraining

How accessible are metadata for different datasets?

Defense: Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning

Uses a trimmed loss function computed on a different subset of residuals in each iteration

$$\min_{\theta,\mathcal{I}} \mathcal{L}(\mathcal{D}^{\mathcal{I}}, \theta) \quad \text{s.t. } I \subset [1, \ldots, N] \wedge |\mathcal{I}| = n$$



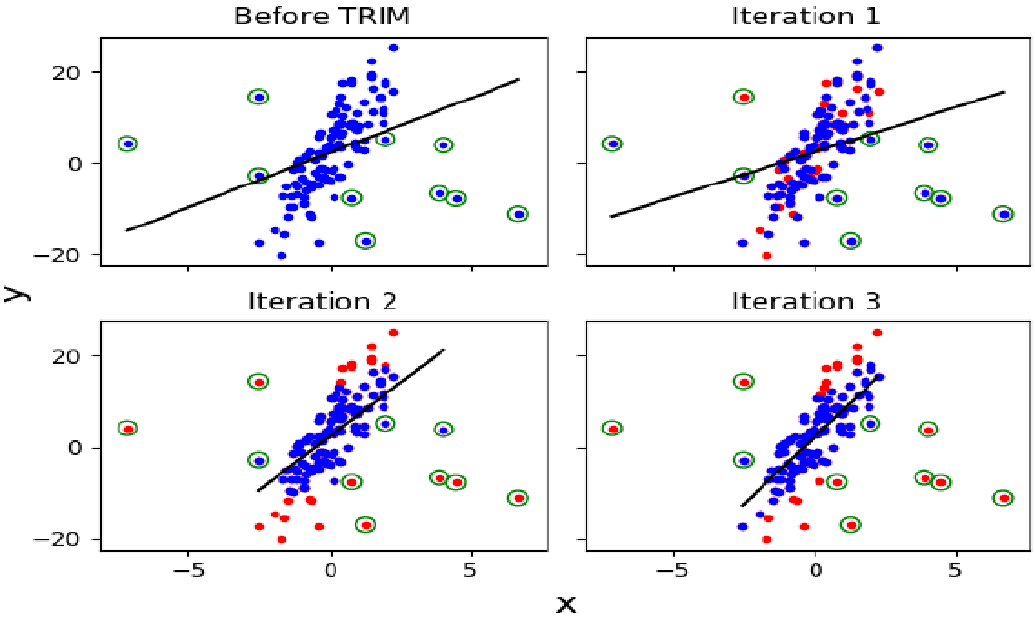
TRIM

1. Removes points with large residuals
2. Estimates the regression parameters

Does not affect inliers!

Only influential poisoning points!

Defense: Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning (cont'd)



Let’s have fun!

Attack: Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks

Assumptions:

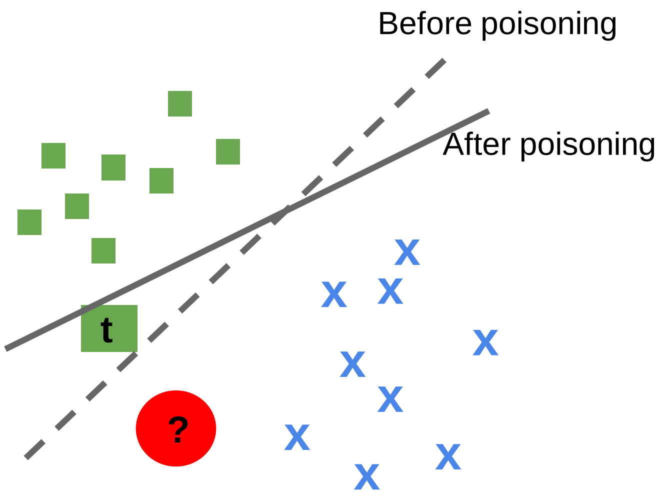
1. No access to training dataset
2. Only experts can label training dataset

Goal: Attacker causes the retrained **neural network** to misclassify a special test instance of one class as another class of her choice

t : target data point from test data (with label L_t)

base class label : L_b

Attack so that t gets label L_b



Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., & Goldstein, T. (2018). Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. *arXiv preprint arXiv:1804.00792*.

Attack: Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks (cont'd)

t : target data point from test data (with label L_t)

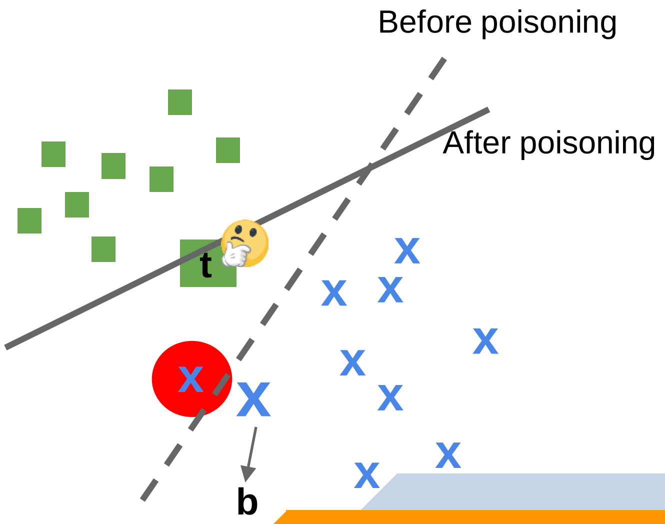
base class label : L_b

t has label L_b

b : from class b with L_b

f(.): propagates an input through the network to the penultimate layer

$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} \ \underbrace{\|f(\mathbf{x}) - f(\mathbf{t})\|_2^2}_{\text{Same output as t: NN classify it in target class}} + \beta \underbrace{\|\mathbf{x} - \mathbf{b}\|_2^2}_{\text{Same appearance as b} \rightarrow \text{Expert labels it as } L_b}$$



Same output as f(t) would be classified same as class of b
→ t would be classified in base class

Attack: Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks (cont'd)

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

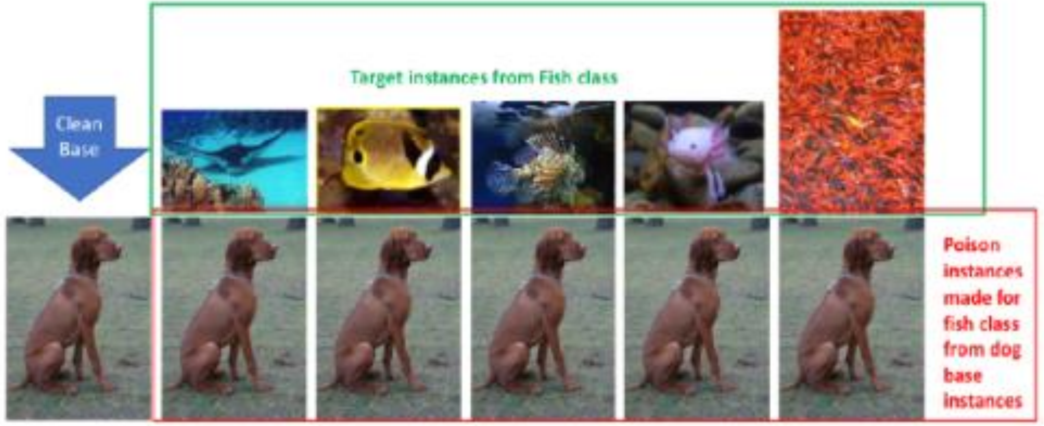
Training method:

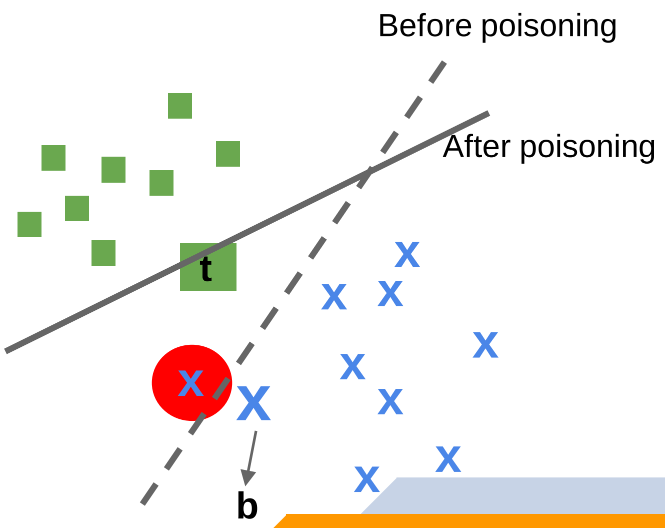
1. Transfer learning:

We already have a pre-trained model

Only softmax layer is retrained

Successful with only one poisoning point





Attack: Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks (cont'd)

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

Training method:

2. End to end:

All layers are trainable

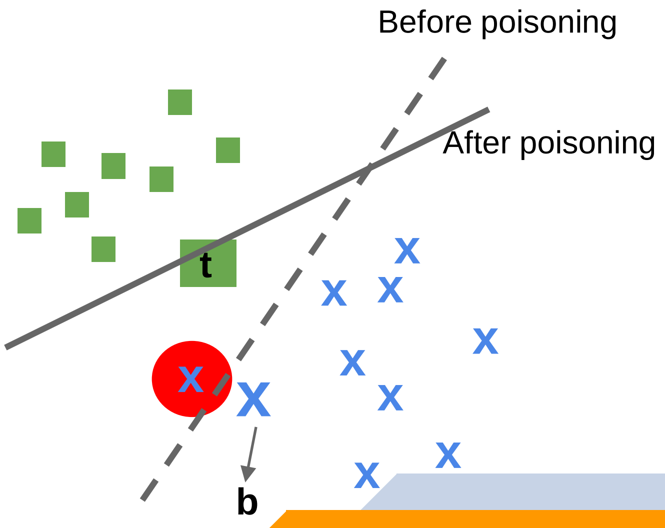
This method does not work anymore

Solution:

Using multiple poisoning points

Watermarking

$$\mathbf{b} \leftarrow \gamma \cdot \mathbf{t} + (1 - \gamma) \cdot \mathbf{b}.$$



Thank You

Any Questions?